Univerzita Karlova
Pedagogická fakulta

BAKALÁŘSKÁ PRÁCE

2022 Stanislav Zdych

Univerzita Karlova

Pedagogická fakulta

Katedra informačních technologií a technické výchovy

BAKALÁŘSKÁ PRÁCE

Procesní model pro tvorbu aplikací VR

Process model for VR application development

Stanislav Zdych

Vedoucí práce:	PhDr. Tomáš Jeřábek, Ph.D.
Studijní program:	Specializace v pedagogice
Studijní obor:	Informační technologie se zaměřením na vzdělávání

2022

Prohlašuji, že jsem bakalářskou práci na téma Procesní model pro tvorbu aplikací VR vypracoval pod vedením vedoucího práce samostatně za použití v práci uvedených pramenů a literatury. Dále prohlašuji, že tato práce nebyla využita k získání jiného nebo stejného titulu.

V Praze 16.4. 2022

.....

podpis

Děkuji PhDr. Tomáši Jeřábkovi Ph.D. za věcné připomínky, cenné rady a odborné vedení při tvorbě této bakalářské práce. Zároveň bych chtěl také poděkovat Danielu Stringerovi za užitečné vhledy do herního vývoje.

ANOTACE

Práce pojednává o možnostech využití virtuální reality pro simulaci muzejních prostředí, popisuje principy na kterých se tato realita zakládá, historický vývoj zařízení společně se současnými trendy. Dále rozebírá vývojová prostředí využitelná pro práci s virtuální realitou, následně navazuje možnostmi tvorby a na závěr představuje autorem vytvořený procesní model, jenž zjednodušuje celý postup tvorby a ukázkové řešení tento model využívající.

KLÍČOVÁ SLOVA

VR, 3D, realita, grafika, Blender, Unity

ANNOTATION

The thesis discusses possibilities of using virtual reality for simulation of museum environments, describes principles which this reality is based on and historical development alongside current trends. It continues by talking about development environments usable for working with virtual reality then follows with creation possibilities and finally presents a process model created by the author which simplifies the entire process of creation and an example solution using this model.

KEYWORDS

VR, 3D, reality, graphics, Blender, Unity

Obsah

1 Úvod

Virtuální realita není v žádném případě nový koncept, první zmínky se objevují již ve třicátých letech dvacátého století [1]. Je to však díky moderním technologiím dneška, které činí tento koncept skutečně dosažitelným nejen pro rozsáhlé instituce, ale i pro běžné spotřebitele. Tento fakt umožňuje aktivní zapojení virtuální reality kupříkladu v datové a architektonické vizualizaci, v personálnější vzdálené komunikaci a spolupráci uživatelů na dlouhé vzdálenosti, v zábavním průmyslu a nakonec ve vzdělávacích institucích [2].

V rámci vzdělávání nejsou striktně zahrnuti pouze žáci technické výuky, právě naopak, virtuální realita může výrazně přispět ve vzdělávání v oborech směrovaných více k humanitním či přírodovědným zaměření. Příkladem muže být například výuka přírodovědy, kde mohou žáci vidět zvířata v korektním měřítku nebo studium zeměpisu, schopnost prohlédnout si volně jakékoliv místo na světě zásadně překonává standardní multimédia.

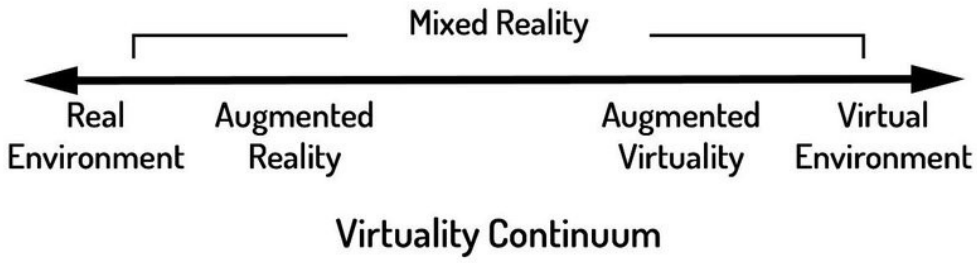
Výraznou překážkou této podpory výuky je v současnosti především absence jednoduchých nástrojů pro vytváření interaktivních prostředí a obsahu. Málokterý pedagog má dostatečné dovednosti v programování či užití herních enginů, které by mu dovolovaly vytvářet si interaktivní VR aplikace pro výuku. Práce se snaží přispět k řešení této problematiky vytvořením procesního modelu, jenž by měl usnadnit tvorbu zaměřených virtuálních prostředí (kompletní abstrakce všech možností virtuální reality by byla nadmíru obtížná). Princip modelu spočívá v práci s předpřipravenými objekty a třídami (třída je struktura v programu, která definuje vlastnosti, které jsou sdílené mezi několika objekty [3]), jenž dovolují vytvořit základní VR aplikaci bez znalostí programování. Vývojáři je poskytnuta šablona projektu obsahující základní komponenty zahrnující kamerový systém s interakční logikou, prvky umožňujícími pohyb po scéně a zobrazování informací v rámci uživatelského rozhraní. Součástí práce je také popis postupu k vytvoření tohoto modelu. Zahrnuty jsou též části vysvětlující problematiku importování 3D modelů s texturami do herních enginů.

2 Cíl a metody práce

Práce si určuje za cíl vytvořit procesní model, jenž usnadňuje tvorbu aplikací pro virtuální realitu, potencionálně zvyšující jejich zapojení do výuky pedagogům, kteří se nepohybují v problematice vývoje v rámci herních enginů. V prvních sekcích práce popisuje principy a postup historického vývoje vedoucího až k současným technickým trendům. Dále rozebírá různá vývojová prostředí k této problematice využitelná, zahrnující jak herní enginy tak i webové služby. Následně popisuje možnosti tvorby obsahu, jenž tvoří hlavní část aplikací. Práce pokračuje objasněním samotného návrhu procesního modelu společně s ukázkami kódu a popisy funkčnosti. Související kapitolou je metodika modelu, jenž prakticky popisuje užití částí modelu současně s kroky objasňující problematiku exportu aplikace. Nakonec je model demonstrován vytvořením ukázkového příkladu, jehož výstupem je aplikace virtuální reality pro vojenské muzeum představující skutečnou funkcionalitu modelu a interakci uživatele s vytvořenými 3D modely.

3 Koncepty související s VR

VR pojednává čistě o digitálním prostoru, samotný termín virtuální reality úzce souvisí s tzv. kontinuem virtuality, jenž popisuje spektrum se dvěma extrémy, reálným prostředím a virtuálním. Právě virtuální prostředí je obecně známé jako virtuální realita. Mezi extrémy reality a virtuality se nachází koncept smíšené reality (Mixed Reality tedy MR) [4]. V současné době se také mluví o termínu Extended reality, jehož cílem je zastřešovat jak koncepty smíšené reality, tak i samotnou virtuální realitu. Společnosti toto zobecnění vytvořily aby zredukovali zmatení zákazníků z nejednoznačných názvů u produktů [5].



Obrázek 1 - Grafická reprezentace kontinua virtuality [6]

Jak již nastiňuje schéma výše, smíšená realita kombinuje virtuální prostředí s reálným, dle míry využití prvků z obou prostředí se smíšená realita dělí na rozšířenou realitu a rozšířenou virtualitu [4].

3.1 Rozšířená realita (Augmented Reality tedy AR)

AR pojednává převážně o reálném prostředí, jenž je obohaceno o virtuální objekty (tedy objekty vytvořené pomocí počítače). Do této kategorie mohou dle Milgrama spadat rozhraní zobrazující reálný záběr upravený pomocí klíčovací technologie či náhlavní displeje (Head Mounted Display tedy HMD), jenž umožňují průnik obrazu reálného prostředí k uživateli [4]. Milgram se u HMD zmiňuje konkrétně o užití zrcadel avšak s rozvojem technologií se tyto přístupy mění. Pomocí odrazu světla fungují například brýle projektu Google Glass, oproti tomu sestava Holo Lens od firmy Microsoft využívá holografických displejů [7, 8]. Speciálním typem AR zařízení se díky současnému vývoji mobilních technologií stávají i chytré telefony, které mohou skrze fotoaparát zobrazovat informace v reálném světě.

3.2 Rozšířená virtualita (Augmented Virtuality tedy AV).

AV se nachází na druhé části spektra, prostředí je zde převážně virtuální s přidanými prvky z reálného světa. Oproti rozšířené realitě se v současnosti nedostává značnému užití, některé náhlavní soustavy s kamerovým sledováním prostoru podporují do jisté míry jak AV tak i AR. Praktickým příkladem může být funkce zobrazení reálné klávesnice u headsetů Oculus Quest 2. Uživatel sedí ve virtuálním prostoru a píše na reálnou klávesnici, během psaní jsou zároveň zobrazeny uživatelovi skutečné ruce [9]. Značnou nevýhodou této funkce je alespoň prozatím podpora pouze jednoho typu klávesnice.

4 Virtuální realita

Oproti rozšířené realitě a rozšířené virtualitě popisuje virtuální realita čistě počítačem generovaný prostor bez jakýchkoliv reálných prvků, občas se k VR referuje jako k virtuálnímu prostředí a mezi koncepty realit patří k nejrozšířenějším. Existuje řada definic, které se snaží virtuální realitu vymezit:

- Jedná o technologii, která vyvolává v uživateli pocit, při kterém se nachází na jiném místě než ve skutečnosti. Tohoto efektu se dosahuje nahrazením primárních smyslů daty vytvořenými počítačem [10].
- Virtuální realita je elektronická simulace prostředí, ke které se přistupuje s pomocí náhlavního displeje a oblečení se zabudovanou kabeláží umožňující konečnému uživateli interagovat v realistických třídimenzionálních situacích [10].

Obecně se tedy pojednává o čistě digitálním prostoru, jenž neobsahující žádné prvky z reálného světa. Dle Heima existují tři základní vlastnosti, také nazývané „tři I“, jenž virtuální realitu definují – imerze, interaktivita a informační intenzita. Imerze je dána mírou izolace uživatele od reálného světa. Imerzi silně ovlivňuje způsob projekce virtuálního prostředí uživateli, tyto způsoby jsou konkrétně rozvedeny v podkapitole Koncepty VR. Interaktivita vychází ze schopnosti počítačů měnit pohled scény stejně rychle jako člověk dokáže měnit svoji fyzickou pozici a perspektivu. Informační intenzita je pojem vyjadřující kvalitu virtuálního světa, který poskytuje speciální prvky jako tzv. telepřítomnost (pocit přítomnosti na místě neodpovídající skutečnosti [11]) či umělé entity vykazující známky inteligence[12].

4.1 Koncepty VR

I přestože se poslední dobou hovoří o virtuální realitě převážně jako systému s náhlavním displejem, podoby VR jsou různorodé a mnohem rozšířenější než si možná někteří lidé uvědomují. Koncepty virtuální reality je možné dělit podle druhu projekce a úrovně interakce s virtuálním prostředím:

- Systém okna je nejzákladnější typ virtuální reality pracující s monitorem počítače jako oknem do interaktivního virtuálního světa, tento systém je možné nadále vylepšit pomocí 3D brýlí [13]. Obecně pod tento systém spadají videohry či jiná interaktivní multimédia.

Obrázek 2 - Systém okna [14]

- Systém zrcadla typicky pracuje s izolací uživatelova těla od reálného prostředí, toto prostředí je následně nahrazeno virtuálním. Výsledek je nakonec promítán uživateli na obrazovku [13]. Tento systém se hojně užívá například ve zpravodajstvích, kdy se moderátor nachází na modrém či zeleném pozadí, které se nahrazuje pomocí klíčování.

Obrázek 3 - Systém zrcadla [15]

- Systém založený na vozidle typicky umisťuje uživatele do vozidla či jeho části, obehnané projekcí virtuálního prostředí, které je ovládané řídícími prvky (volant, knipl, joystick apod.) [13]. Využívá se hojně v různých typech simulátorů či arkádových hernách.

Obrázek 4 - Systém založený na vozidle [16]

- Systém jeskyně (často nazýván jako „cave“) popisuje prostor, ve kterém je uživatel téměř kompletně obklopen obrazovkami, občas se k tomuto systému přidávají 3D brýle ke zvýšení imerze [13]. Často se využívá v zábavních parcích či interaktivnějších muzeí.

Obrázek 5 - Systém jeskyně [17]

- Imerzivní systém si pod termínem virtuální reality vybaví pravděpodobně většina lidí, tvořího náhlavní stereoskopický displej (také znám jako HMD – Head Mounted Display), často i ovladače pro ruce a v některých případech celotělový oblek [13]. Využívá se v celé řadě aplikací - uměleckých, návrhových, rehabilitačních, rekreačních a dalších.

Obrázek 6 - Imerzivní systém [18]

Imerzivní systémy s HMD jsou hlavním zaměřením této práce z následujících důvodů:

- Vysoká míra imerze – Náhlavní displeje velice efektivně izolují uživatelův pohled od reálného prostředí, zároveň poskytují vysokou míru interakce s virtuálními objekty.
- Systémová mobilita - Systém jeskyně či vozidla může sice poskytovat vyšší úroveň imerze, nemůže být však volně přenášen a rychle nainstalován, většina imerzivních systémů nevyžaduje rozsáhlou úpravu místnosti pro jejich funkčnost, některé dokonce nevyžadují žádnou.
- Cena – Za posledních několik let se staly HMD systémy výrazně dostupnější pro běžné uživatele a v poměru ceny a úrovně imerze mají jasnou výhodu oproti ostatním typům.

V práci se pod pojmem VR uvažuje zejména o imerzivních systémech s HMD.

4.2 Hlavní charakteristiky VR

4.2.1 HMD systémy

Náhlavní displeje se za dobu svého vývoje dostali do poměrně standardizované podoby, zařízení, jenž si uživatel nasadí na hlavu, umožňující rozhlížení i pohyb ve virtuálním prostoru. Displeje se klasicky řeší dvěma způsoby – jeden displej rozdělen softwarově na dva anebo dva samostatné displeje. Před vykreslovaným obrazem se typicky nachází čočky, díky kterým se zvyšuje zorné pole uživatele. Vykreslováním vlastního obrazu pro každé oko se dosahuje hloubkového vnímání skrze tzv. Steroskopické zobrazení, jenž je popsané v následující kapitole.

HMD často také obsahuje zabudované reproduktory či sluchátka. V převážné většině se headsety prodávají v sadách, jejichž součástí jsou i dva ovladače pro ruce. Podstatné rozdíly se vyskytují hlavně v rámci samotné detekce pohybu v prostoru a způsobu vykreslování grafických dat.

Stereoskopické zobrazení

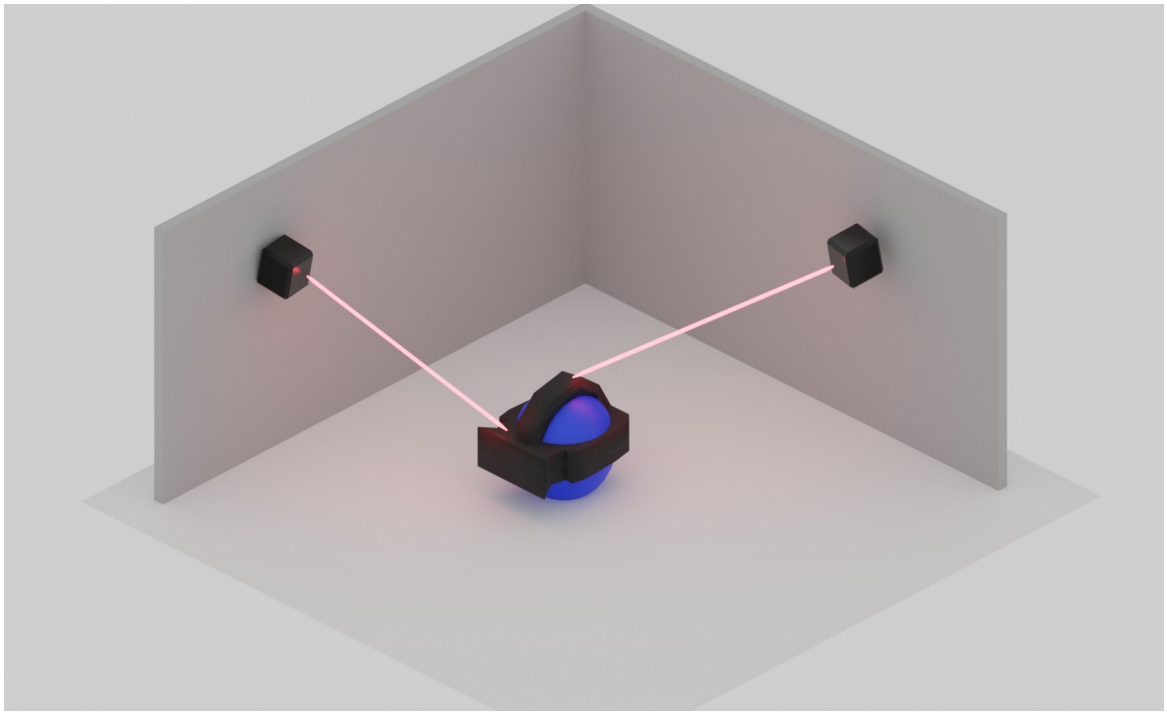
Stereoskopické zobrazení je jedním ze způsobů, který umožňuje (nejen) člověku vnímat hloubku v prostoru tím, že každému oku předkládá lehce odlišný obraz. Vedle stereoskopie se může hloubka určit například pomocí změny pozice objektu při pohybu hlavy či měřením ohniskové vzdálenosti nutné pro dosažení ostrosti objektu. Výhodou stereoskopie oproti ostatním metodám je spolehlivost hloubkových informací, které poskytuje [19]. Tyto informace jsou velice důležitá pro uživatelovu imerzi, jakmile se uživatel soustředí čistě na hloubku virtuálního prostoru, začne ignorovat vady v izolaci od reálného světa. Vnímání objektů v prostoru zároveň zásadně vyzdvihuje informační intenzitu virtuálního prostředí.

Principem tohoto zobrazení je využití dvou vstupních prvků (oko, kamera), které nahlížejí na stejnou scénu, ovšem z lehce odlišných pozic. Výstupní data, která jsou z těchto prvků získána se porovnávají pomocí rozdílů v pozicích sledovaných objektů, z těchto porovnání se následně získávají informace o hloubce [20]. Člověk si však uvědomuje pouze jeden obraz, protože mozek spojuje oba obrazy do jednoho.

Mezi první technologická řešení pracující s prostorovým zobrazením patří stereoskopy, zařízení jenž oddělují informace o obrazu optickou cestou, objevily se již ve třicátých letech 19. století. Velice známým a relativně novějším způsobem zobrazování jsou anaglyfové brýle, které oddělují obraz pomocí barevných filtrů, nejčastěji kombinací barev červené a azurové [21]. Anaglyf se dočkal vysoké obliby v kinematografii 20. století, současná kinematografie využívá principu polarizace pro dosažení prostorového zobrazení.

Sledování pohybu pomocí externích stanic

Metoda, která přišla s prvními moderními HMD využívá externích sledovacích stanic, které je nutné správně rozmístit do místnosti tak, aby měli co nejvyšší rozhled po prostoru. Při správném umístění poskytuje sledování pozice jak headsetu, tak i ovladačů a případných dalších prvků s vysokou přesností nehledě na jejich pozici k uživateli.



Obrázek 7 - Ukázka sledování pohybu pomocí externích stanic

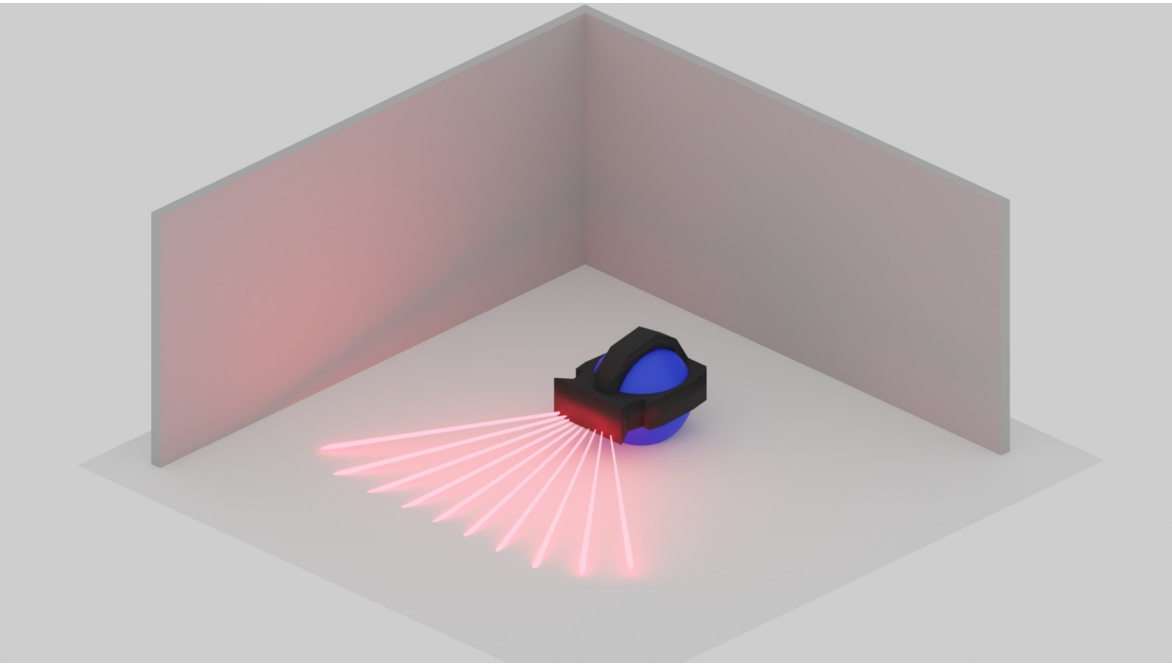
Nevýhodou je však omezení prostoru na oblast pokrytou stanicemi, kvůli specifickému umístění stanic není možné systém takového typu jednoduše přesouvat. Nakonec většina systémů s touto metodou v základu vyžaduje připojení přes množství kabelů, které se mohou uživateli plést do cesty.

Firma Oculus ve svých prvních verzí brýlí Oculus Rift užívala externích stanic, s řadou Quest a novějších používá již zabudované systémy, všechny jejich brýle jsou ale v jisté úrovni založeny na kamerovém systému „Constellations“, referující na velké množství infračervených ledek, jejichž pozice jsou triangulovány z obrazu kamer (interních i zabudovaných) [22].

Velkým průkopníkem externích sledovacích systémů je společnost Valve s poměrně odlišným řešením. Systém Lighthouse byl použit již u prvních systému HTC Vive, externí stanice v tomto případě nefungují jako senzory, ale jako laserové emitory, jenž střídají pokrytí mezi vertikálním a horizontálním směrem. Headset a ovladače vystřelené lasery zachycují, k tomuto způsobu lokace se využívá minimálně dvou stanic (lze i více) [23].

Sledování pohybu pomocí zabudovaných kamer

Druhou metodou, která přišla s modernějšími systémy je zabudování sledovacích kamer přímo do headsetu, uživatel nepotřebuje sledovací stanice a často ani externí výpočetní zařízení, jelikož tyto systémy dosahují dostatečného výkonu na vykreslování prostředí bez větších obtíží. Tato metoda se často pojí s naprostou volností pohybu, omezenou pouze reálným prostorem a absencí kabeláže pro samotný headset či ovladače.



Obrázek 8 - Ukázka sledování pohybu pomocí zabudovaných kamer

Úskalím této metody je samotná přesnost sledování, která zaostává v momentech, kdy kamery nevidí senzory na ovladačích či při neoptimálních světelných podmínkách.

Jak již bylo zmíněno, Oculus využíval u externích stanic systému Constellations, ten se stále uplatňuje u sledování pohybu ovladačů, k pohybu po místnosti se však z principu musí využívat jiného systému. Skrze zabudované kamery se vytváří 3D mapa prostoru pomocí vizuálních záchytných bodů, kupříkladu rohů nábytku, tento sledovací systém tedy funguje nejlépe v místnostech, kde se nachází velké množství různorodých objektů. V současnosti s ním pracují poslední verze Oculus Rift S a headsety Quest 1/2 [24].

Externí vykreslování grafických dat

Vykreslování prostředí vyžaduje výkonné zařízení, u tohoto typu se většinou jedná o stolní počítač či notebook, s adekvátně silnou grafickou kartou, ke kterému je samotný headset připojen. Veškeré výpočty probíhají na straně počítače, brýle samotné žádné výpočetní operace neprovádí. Výhodou tohoto způsobu je vyšší grafická kvalita prostředí (případně vyšší výdrž baterie samotného HMD, pokud je bezdrátového typu), na druhou stranu se tento typ přesouvá výrazně hůře.

Interní vykreslování grafických dat

Řada headsetů se v současnosti kompletně osamostatnila od přídavných zařízení díky výrazným pokrokům v oblasti mobilních čipů. Mobilita těchto zařízení je díky tomu na velice dobré úrovni. Navíc převážná většina z nich využívá i již zmíněných zabudovaných sledovacího systémů, které přenositelnosti dále nahrávají. Grafická fidelita je však jedním z kompromisů tohoto pohodlí, i přestože je možné tyto headsety připojit ať už drátově či bezdrátově k počítači, kvalita obrazu sepředchozímu typu nevyrovnává. Důvodem je většinou komprese videa, která musí být dostatečné vysoká aby přenesla velké množství dat s co nejnižším zpožděním přes rozhraní, které na to nebyla optimalizována (WiFi či USB-C).

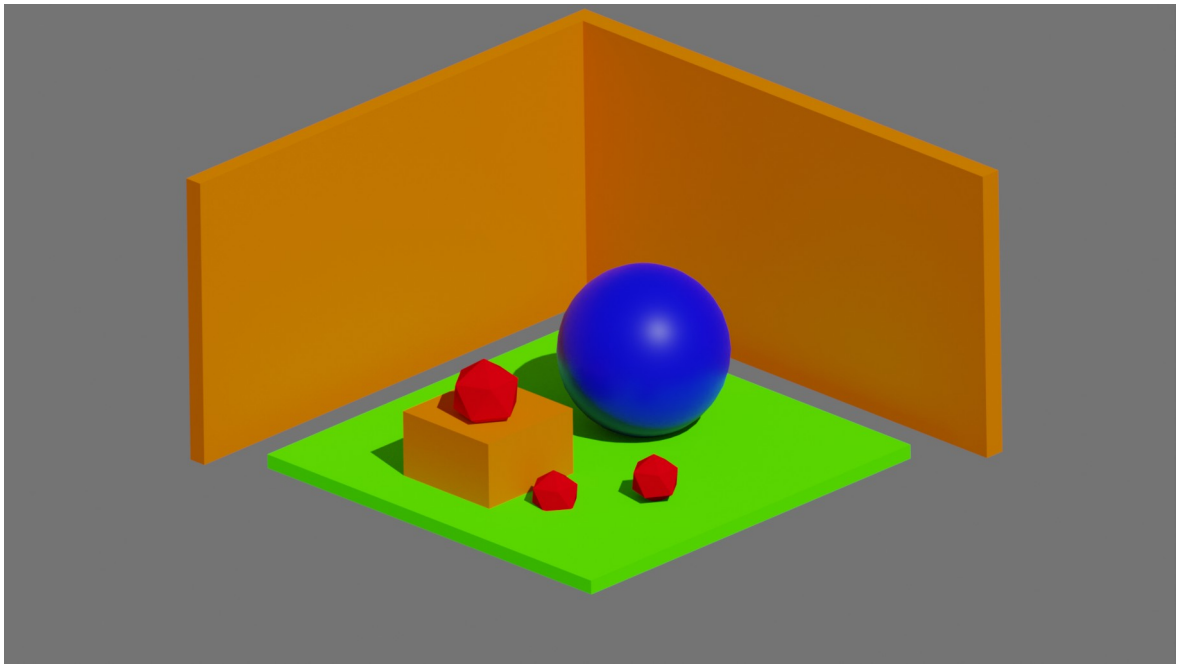
4.2.2 Prostředí virtuální reality

Existují debaty [25, 26, 27] o tom, co spadá pod pojem prostředí virtuální reality, v této práci se takto referuje k virtuálnímu prostoru zahrnující jak virtuální scénu, tak i uživatele stechnickými ovládacími prvky (HMD, ovladače pro ruce, apod.).

Základní virtuální scéna

Virtuální scéna je počítačem generovaná část prostředí, která uživatele obklopuje a se kterou může prostřednictvím zařízení pro virtuální realitu interagovat.

Každá scéna obsahuje několik elementárních prvků, které se liší svojí funkcionalitou. V následujícím obrázku jsou tyto prvky barevně označeny.



Obrázek 9 - Barevné schéma zobrazující základní prvky scény

Šedivá barva (pozadí schématu je též zahrnuto) představuje objekty se kterými uživatel nemůže interagovat, často se jedná o podlahu určenou pouze jako texturový podklad či oblohu.

Zelená barva reprezentuje oblast po které se může uživatel pohybovat ať už pomocí teleportace či jiného typu pohybového systému. Dle limitací aplikace se plochy symbolizované šedivou barvou mohou kompletně nahradit zelenou pohybovou plochou.

Oranžová barva symbolizuje statické prvky scény, která sice kolidují s uživatelem a ostatními objekty, nejsou však smýšleny jako interaktivní pro uživatelovu přímou manipulaci.

Červenou barvou jsou zvýrazněny uchopitelné (nebo jinak interagovatelné) objekty pro uživatele, oproti předchozím se liší především v tom, že uživatel dokáže mění jejich stav (pozici, rotaci, apod.).

Nakonec modrá barva představuje kamerový objekt, jenž reprezentuje pohled uživatele. Tento objekt je ve scéně vždy pouze jeden i ve chvílích, kdy se ve scéně zdánlivě nachází vícero uživatelů. Tento fakt je dán tím, jak se v praxi vytváří pouze dojem o přítomnosti dalších uživatelů.

4.2.3 Interaktivita

Samotný termín interaktivita není pevně definovaný koncept, avšak obecně se tak popisuje akce, kterou provádí člověk či objekt ve vztahu k ostatním lidem či objektům. Mezi webovými designéry se interakce může pokládat za ekvivalent navigace v rámci webu či znak dobře zpracované webové stránky z pohledu uživatelské přehlednosti. Ze softwarového pohledu se považuje interakce za události, které vyvolává uživatel - pohyb myši, stisk tlačítka myši či klávesy apod. [28].

Z praktického hlediska je ve v rámci virtuální reality interakcí brána činnost uživatele ve virtuálním prostředí – pohyb hlavou, přesun po prostoru, uchopování a manipulace s virtuálními objekty, apod.

Současné technologie nabízí rozsáhlou míru interaktivity ve virtuálním prostoru, tou nejspíše nejdůležitější je schopnost rozhlížet se v prostoru. Rané pokusy o vytvoření virtuálních brýlí umožňovali pouze simulovat prostorové zobrazení pomocí čoček, moderní headsety využívají celé řady senzorů, které jim umožňují snímání rotace.

Pokročilejší interaktivitou je pohyb v prostoru, ten se ve svých implementacích liší, záleží na preferenci uživatele a na prostorové kapacitě místnosti. Standardně je možné se s brýlemi pohybovat po prostoru jako v reálném životě, toto zajišťují sledovací prvky, jenž mohou být externě přidělané na zdech pokoje či zabudované přímo v headsetu.

Finálním typem interakce je samotná možnost interagovat s virtuálním scénou, zda-li je možné brát modely do rukou a nebo jestli lze například zatáhnout za páčku. Tento typ výrazně přispívá uvěřitelnosti virtuálního prostředí [29].

4.3 Aplikace v kontextu vzdělávání

Jedním z častých problémů ve vzdělávání je udržení pozornosti studentů a jejich aktivní zapojení ve výuce. V tradiční výuce, kde pedagog vykládá látku, občas ukáže obrázek či video, převládá problém pozornosti především kvůli nízkému zapojení žáka do hodiny. Zapojení vysoce interaktivní technologie, jako v tomto případě virtuální realitu, může výrazně přispět ke zvýšenému zájmu studentů o danou oblast lekce [30].

Virtuální realita umožňuje žákům učit se zážitkem oproti výkladu, zvýšená imerze probírané látky zlepšuje vzdělávací efektivitu celé lekce. Jednou z možných oblastí využití,

zvláště v době pandemie, je výuka zeměpisu. Aplikace Google Map je značně rozšířená ve světě webových technologií a mobilních aplikací, Google však má též tento software adaptovaný pro virtuální realitu umožňující pohybovat se téměř po jakémkoliv místě na světě. Kupříkladu vidět pyramidy v Gíze na obrázku je výrazně rozdílný zážitek oproti možnosti stát přímo před ní a vidět ve skutečném měřítku velikost stavby. Virtuální prostředí také přináší možnost dělat chyby a učit se z nich, ve výuce chemie si student standardně nemůže zaexperimentovat stylem „co by, kdyby“. Ve virtuální realitě tyto možnosti jsou a nepřináší žádné riziko pro studenta ani zbytek kolektivu (pokud ovšem student nepřekročí bezpečné hranice prostoru a nenarazí do překážky v reálném světě). Užití virtuální reality pro výuku historie může poskytnout výrazně lepší představu o událostech minulosti, kupříkladu zažít bitvu u Waterloo a vidět z blízka divize vojáků stojících v řadě, synchronizovaně pálící salvy z mušket přinese úplně jiný vhled do Napoleonských válek než úryvek textu v učebnici [30].

Virtuální muzea jsou již existující, spíše podpůrný koncept pro fyzická muzea, VR má však potenciál nahradit klasické návštěvy muzeí, které by byly pro zájemce jinak nedosažitelné. Právě čistě virtuální muzea jsou primárním zaměřením této práce.

I přestože ideální scénáře výukových aplikací virtuální reality vypadají velice pozitivně je důležité také zmínit možné problémy. Mezi nejzásadnější patřící kvalita zpracování. V případě historie jsou nepřesnosti kritické a mohou vést k chybným pojetí minulosti podobným například romantizaci japonských samurajů typické pro moderní multimédia. Dalším úskalím může být nedostatek prostředků vzdělávací instituce, zapojení této technologie aktivně do výuky vyžaduje nákup samotných zařízení a u externích typů i výkonných počítačů. Zároveň musí být učitel schopný s těmito zařízeními pracovat.

5 Historický vývoj a aktuální trendy

První představy o virtuální realitě obdobné současným technologiím se objevovaly již ve třicátých letech dvacátého století v krátkém příběhu „The Pygmalion's Spectacles". Zde postava profesora Ludwiga vynalezla brýle umožňující uživateli zažít virtuální prostor. Reálný vývoj headsetů však začal až v šedesátých letech pro armádní využití, uplatňující se především v leteckých simulátorech a bojových cvičenív rámci bezpečného prostředí. Virtuální realita se do civilního sektoru dostala teprve na začátku devadesátých let.

5.1 Virtual Boy

Obrázek 10 - Sestava Virtual Boy [31]

V roce 1995 přišla japonská herní společnost s první konzolí virtuální reality, sestava obsahovala brýle na stojánku s gamepadem. Zařízení bylo velice nepohodlné, displej zobrazoval pouze červené pixely na černém pozadí, způsobující bolesti hlavy pro mnohé uživatele. Značně omezená softwarová knihovna zařízení také neprospěla a pro Nintendo se z konzole stal komerční neúspěch [32].

5.2 Sega VR

Konkurenční Sega se také pokoušela o vytvoření zařízení pro virtuální realitu, Sega VR byly brýle velice podobné dnešním headsetům určené pro konzole Genesis. V roce 1993 bylo zařízení představeno veřejnosti na veletrhu v Chicagu. V Severní Americe mělo ještě ten rok vyjít, vývoj se však dostal do komplikací kvůli nevolnostem při testování a vydání

se odložilo na další rok. Projekt Sega VR byl nakonec zrušen a veškerý vývoj přešel na virtuální zařízení v arkádových hernách [33, 34].

Obrázek 11 - Brýle Sega VR [35]

5.3 Oculus

Úspěch virtuální reality dorazil až v roce 2012 s malou skupinou vývojářů a jejich kickstarterovou kampaní žádající 250 tisíc amerických dolarů. Finální vybraná částka činila téměř 2,5 miliónů, toto byl začátek společnosti Oculus, která vydala jejich první vývojářský headset s označením DK1 v roce 2013 [36].

5.3.1 DK1/DK2

Development Kit 1 umožňoval volnou rotaci hlavou do všech směrů s horizontálním zorným polem devadesáti stupňů, LCD displej dosahoval rozlišení 640x800 pro každé oko a obnovovací frekvenci 60 Hz. Headset měl mezi komunitou úspěch avšak vizuální zážitek byl stále daleko od pohodlného užívání a tak o rok později přišla druhá verze.

Obrázek 12 - První vývojářská verze brýlí Oculus [37]

Development Kit 2 přinesl OLED displej s rozlišením 960x1080 pro každé oko a dosahoval frekvence 75 Hz. Horizontální zorné pole se zvýšilo na devadesát tři stupňů. Výraznou novinkou však bylo sledování pohybu po místnosti pomocí externích sledovacích stanic [38, 39].

Obrázek 13 - Druhá vývojářská verze brýlí Oculus [40]

5.3.2 Rift

V roce 2014 odkoupil firmu Facebook, díky čemuž získal Oculus rozsáhlejší finanční prostředky na vývoj a o dva roky později představil první headset pro běžné uživatele označován CV1 – Consumer Version 1, později pouze jako Oculus Rift. Displej opět

postoupil na vyšší úroveň s rozlišením 1080x1200 a 90 Hz na každé oko, tentokrát se ovšem nejednalo o jeden displej rozdělen na dvě půlky ale o dva samostatné AMOLED displeje. Horizontální zorné pole se lehce snížilo na osmdesát sedm stupňů. Součástí sestavy byly vedle headsetu a sledovacích stanic i dva ovladače Oculus Touch sledující pohyb rukou, které zajišťovali interakci uživatele s virtuálním prostředí výrazně přirozenějším způsobem [41].

Obrázek 14 - Sestava Oculus Rift včetně ovladačů a sledovacích stanic [42]

5.3.3 Go

Všechny headsety na začátku moderní virtuálních reality vyžadovaly připojení k výkonnému počítači, Oculus však roku 2017 představil jeden z prvních plně samostatných zařízení s vlastním výpočetním výkonem. Oculus Go byl hardwarově podobný vlajkovým lodím mobilních telefonů své doby, upravený systém Android s procesorem Snapdragon 821 a grafickým akcelerátorem Adreno 530 umožňoval spouštět interaktivní obsah na LCD displeji s rozlišením 1280x1440 při 60 Hz na každé oko se zorným horizontálním úhlem osmdesáti devíti stupňů. Oproti Oculus Rift neumožňoval sledování v prostoru. Součástí sestavy byl pouze jeden ovladač sloužící spíše jako dálkové ovládání, díky dostupné ceně se však hojně rozšířil [43].

Obrázek 15 - Sestava Oculus Go [44]

5.3.4 Quest 1/2

Problémy verze Go měla podle plánu vyřešit náhlavní soustava Oculus Quest. Hardwarově se jednalo o hybrid mezi interaktivními možnostmi verze Rift a volností verze Go. Quest, vydaný v roce 2019, opět poháněl upravený systém Android, tentokrát ve vyšší verzi s procesorem Snapdragon 835 a grafickým akcelerátorem Adreno 540. Obsahoval rozšířenou kapacitou paměti i úložiště. Oproti Go obsahoval dva OLED displeje každý s rozlišením 1440x1600 při 72 Hz a horizontálním zorným polem devadesát tři stupňů. I přestože umožňoval pohyb v prostoru, nebyly součástí sestavy sledovací stanice. Quest byl mezi prvními headsety využívající zabudované kamery pro sledování pohybu. Součástí sestavy byly také dva ovladače Oculus Touch druhé generace, které mohly díky kapacitním tlačítkům zobrazovat pozice prstů na ovladači ve virtuálním prostoru. Softwarovou novinkou bylo zavedení renderování dle zorného úhlu, snižující nápor na mobilní hardware v komplexních scénách, přidán byl také režim Passthrough, ve kterém mohl uživatel sledovat reálný prostor skrze zabudované sledovací kamery. Quest byl komerčně značně úspěšný a stal se jedním z nejrozšířenějších headsetů.

Obrázek 16 - Sestava Oculus Quest [45]

Na úspěchu první verze zakládal Quest 2, přinášející hardwarová vylepšení ve všech směrech. Kvůli udržení nízké ceny však vytvářel kompromisy v ostatních oblastech. Díky výrazně výkonnějšímu procesoru Snapdragon XR2 s grafickým akcelerátorem Adreno 650 bylo možné zvýšit rozlišení na 1832x1920 při velmi vysoké frekvenci 120 Hz. Horizontální zorné pole bylo rozšířeno na devadesát sedm stupňů. Díky třetí generaci ovladačů Touch se výrazně zefektivnilo sledování pohybu, zároveň se snížila spotřeba baterií. Softwarovým pokrokem druhé a později i první verze se stala možnost propojit headset k počítači a využívat tak externího obsahu mimo samotné zařízení, prvním krokem bylo zavedení Oculus Link tvořící toto propojení pomocí USB-C, později vyšlo i bezdrátové řešení fungující skrze Wi-Fi zvané AirLink [46, 47].

Obrázek 17 - Druhá verze sestavy Oculus Quest [48]

5.4 Valve

Herní společnost Valve, známá především díky sériím jako Half Life či Portal, začala svůj vstup do odvětví virtuální reality spoluprací s firmou HTC. Společně pracovali naheadsetu HTC Vive, jenž vyšel v roce 2016 [49].

5.4.1 HTC Vive (Pro)

V době vydání byl headset přímou konkurencí pro Oculus Rift. Rozlišení i frekvence displeje byly pro obě zařízení totožných 1080x1200 při 90 Hz, Vive měl však výhodu ve vyšším horizontálním zorném poli čítající sto osm stupňů. Obdobně jako Rift obsahovala sestava externí sledovací stanice poskytující volný pohyb v oblastech čítající přibližně až 4,5 metrů čtverečních a dva ovladače Vive Controller. Vylepšenou verzi dvojice firem v roce 2018 pod názvem HTC Vive Pro. Nová verze přinesla vyšší rozlišení čítající 1440x1600 pixelů, frekvence zůstala stejná, novinkou pro tuto řadu headsetů byl režim passthrough skrze dvě zabudované kamery na přední straně brýlí. Vylepšeny byly i ovladače [49, 50].

Obrázek 18 - Sestava HTC Vive včetně ovladačů a sledovacích stanic [51]

5.4.2 Valve Index

V rámci řad Vive se Valve drželo spíše na straně softwarové části vývoje, budující integraci s jejich systémem SteamVR. Nový headset, který vydali roku 2019 byl však kompletně navržen a distribuován bez externí podpory. Index se rozlišením displeje rovnal HTC Vive Pro, obnovovací frekvence však byla výrazně zvýšena na 144 Hz patřící k nejvyšším v rámci virtuálních brýlí dané doby. Horizontální zorné pole čítalo sto osm stupňů totožných s první verzí Vive. Jednou z největších inovací tvořily ovladače Index Controller výrazně se lišící od konkurenčních výrobců, díky unikátní konstrukcimohl uživatel uvolnit zápěstí a pohybovat s rukama přirozeně bez nutnosti neustálého držení ovladače, kapacitní plochy na ovladačích detekovali, zda-li uživatel sevřel zápěstí a virtuální ruce následně patřičně zareagovali [52].

Obrázek 19 - Sestava Valve Index včetně ovladačů a sledovacích stanic [53]

5.4.3 Projekt Deckard

Valve se svojí příruční konzolí Steam Deck, založenou na Linuxu, spolupracoval s výrobcem čipů Advanced Micro Devices, vyvíjející efektivnější procesor a grafický akcelerátor pro zařízení mobilnější než tradiční počítače či notebooky. Přínosem tohoto vývoje byla možnost využití těchto výkonných technologií pro samostatné VR zařízení podobné konkurenčnímu Oculus Quest. V době psaní bakalářské práce nebyl tento vývoj oficiálně potvrzen, avšak v softwaru SteamVR byly objeveny reference na zařízení označované jako Deckard, nalezeny byly také žádosti o patent ke konci roku 2021 spojené s tímto projektem a samotná společnost Valve zmínila možné prospekty jejich spolupráce s AMD do budoucích projektů. Několik zdrojů zmínilo ambice ve využití zabudovaných senzorů pro sledovaní pohybu místo tradičních externích stanic typu Lighthouse používaných u Indexu [54].

6 Možnosti tvorby obsahu

Vytváření VR aplikací vyžaduje dvě zásadní části. První jsou objekty tvořící scénu, pro jejich tvorbu se využívá software pro modelování ve 3D. Druhou, logickou část, zajišťuje herní engine, který se stará o osvětlení, interakci mezi objekty a zajišťuje spustitelnost finální aplikace.

6.1 3D modelování

V rámci modelování se naskytuje řada možností, v práci se využívá svobodný otevřený program Blender, ale je možné využít dalších například Cinema 4D, 3D Studio Max, Maya a další. Alternativou může být využití prací jiných autorů, v takovém případě však mohou vznikat problémy s licencí či nekonzistentní kvalitou modelů.

6.1.1 Blender

Jak již bylo zmíněno jedná se o svobodný modelovací software s otevřeným kódem pod licencí GNU General Public License. I přestože modelování bývá nejčastějším způsobem užití tohoto softwaru není to jediné čeho je schopen, poskytuje kompletní balíček pro renderování scén, vytváření koster postav pracující s váhami v animacích, provádění fyzikálních simulací, tvorbu video kompozicí a do jisté míry i vývoj her. Silnou stránkou Blenderu je jeho modifikovatelnost pomocí skriptů tvořených v jazyce Python. I přestože se původně jednalo spíše o amatérský software, v dnešní době se stává průmyslovým standardem [55].

6.1.2 Cinema 4D

Světově známý software od firmy Maxon, hojně používán v profesionální sféře 3D modelování, fyzikálních simulací a renderování scén. Oproti softwaru jako například výše zmíněný Blender je Cinema známá svou vyšší přístupností k novým uživatelům ovšem za poměrně vysoký licenční poplatek. I přestože se nejedná o open source program, poskytuje Maxon nástroje pro vývoj rozšíření psané v jazyce C++ [56, 57].

6.1.3 3D Studio Max

Další známý komerční software dříve rozšířenýhlavně mezi herními vývojáři a studii tvořící speciální efekty pro filmový průmysl. Oproti výše zmíněným programům funguje

3ds Max pouze na platformě Windows. Firma Autodesk vyvíjející tento software v roce 2005 odkoupila konkurenční Mayu, která se postupně dostala do popředí hlavně díky lepším animačním nástrojům [58].

6.1.4 Shadery a textury

Všechen zmíněný software v kapitole 3D modelování umí pracovat s shadery, uživatelsky definovanými programy, které se slouží k vykreslování obrazových bodů a používají se k vylepšení vizuální kvality modelů. Součástí shaderů může být například jednoduchá barva na objektu anebo naopak komplexní struktura řešící přechody mezi texturami, vytvářející detaily reagující na světlo při zachování nízkého počtu polygonů v topologii modelu. Textury jsou jednou ze součástí shaderu, při práci se shaderem v Blenderu není možné využití stejných dat v enginu Unity, oba programy s nimi totiž pracují rozdílně a ne vždy podporují stejné funkce. Často je tedy nutné takzvaně „zápect“ detaily vytvořené v shaderu do jednoho grafického souboru, který lze následně v herních enginech použít.

6.2 Vývojová prostředí

Pro potřeby tvorby aplikací ve virtuální realitě se nejvíce hodí herní enginy, jsou pro interaktivní aplikace určené a ty nejrozšířenější v sobě již obsahují pomocné nástroje, které tvorbu VR usnadňují. V této práci byl využit pro vývoj multiplatformní engine Unity, který je pro běžné vývojáře zdarma. Dalšími populárními enginy pro tyto účely bývají Unreal Engine od firmy Epic či otevřený engine Godot. Způsoby vývoje se však mezi zmíněným softwarem výrazně liší a nelze tedy využít vytvořený procesní model v jiném enginu než Unity.

6.2.1 Sketchfab

Webová služba, která je v této sekci zahrnuta z důvodu jednoduchosti implementace. Nejedná se o plnohodnotnou aplikaci virtuální reality, ale archiv 3D modelů umožňující zobrazování jednotlivých modelů pomocí webového standardu WebVR. Nevýhodou tohoto způsobu je omezení na jediný model ve scéně a téměř nulová interakce s modelem samotným [59].

6.2.2 Unity

Herní engine nejvíce rozšířený mezi jednotlivci či malými týmy vývojářů, především díky rozsáhlé dokumentaci a celé řadě návodů na službách jako Youtube či Vimeo. Unity umožňuje multiplatformní vývoj na veškeré druhy zařízení, často se tedy využívá pro vytváření mobilních, desktopovýchi webových aplikací. Aplikační logika se píše v jazyce C#, která se následně kompiluje v otevřené implementaci .NET frameworku zvané Mono. Engine podporuje jak vývoj 2D tak i 3D aplikací a oproti kupříkladu Unreal Enginu je výrazně všestrannější v možnostech vývoje [60].

6.2.3 Unreal Engine

Velice známý herní engine s první verzí datující se již do roku 1998, využívaný ve hrách jako Unreal či později Unreal Tournament. UE je dnes významný především díky grafickým možnostem převyšující konkurenci a značnou řadou předpřipravených nástrojů na vývoj. V současnosti existuje pátá iterace enginu přinášející významné pokroky ve vykreslování modelů s hustou topologií pomocí technologie Nanite Virtualized Geometry a v práci s dynamickým osvětlením v reálném čase. Unreal poskytuje dvě možnosti vývoje,

pomocí vizuálního programování v rámci Blueprint Visual Scripting či přímo skrze jazyk C++. Engine poskytuje nezanedbatelné množství nástrojů pro rychlé budování aplikací, to však přináší problémy, Unreal byl od začátku zaměřen pro vytváření her v žánru střílečeк. Vývoj jiných žánrů je samozřejmě možný, ovšem vyžaduje značnou práci na tvorbu vlastních řešení, Unity je v tomto ohledu všestrannější [60].

6.2.4 Godot

Jedná se o otevřený software pod licencí MIT, jenž se v současné době začíná více rozšiřovat mezi vývojáři indie her. Godot poskytuje základní nástroje pro vývoj 2D i 3D aplikací pro desktopové, mobilní a webové platformy. Oproti výše zmíněným enginům nevyžaduje placení podílu zisku v případě vyšších prodejů, zároveň nabízí větší výběr v programovacích jazycích, vývoj může probíhat v GDSciptu, jazyku podobnému Pythonu, grafické programování pomocí Visual Scripting, C# anebo C++. Díky komunitnímu stylu vývoje softwaru se objevují rozšíření přidávající podporu pro další jazyky. Oproti jiným enginům je také výrazně méně náročný jak na úložiště vývojáře tak na výkon jeho zařízení [61].

7 Procesní model

Jak již bylo zmíněno v úvodu, primárním zaměřením této práce je vytvoření procesního modelu, jenž je v tomto případě projektová šablona herního enginu Unity. Šablona je koncipována primárně pro tvorbu virtuálních muzeí. Unity bylo zvoleno kvůli početné dokumentaci jak od oficiálních tak neoficiálních zdrojů, značné rozšířenosti tohoto enginu a bezplatné licenci*.

Ve spojení s enginem byl pro tvorbu modelů a textur zvolen modelovací software Blender, protože podporuje export do .fbx formátu používaným Unity, zároveň se jedná o open source software zdarma, díky tomu je velice oblíbený mezi nezávislými vývojáři a malými studii.

Model je psán v anglickém jazyce, aby se přizpůsobil vývojovému prostředí Unity, které nemá českou lokalizaci.

*bezplatná pro projekty, jejichž výdělek čítá maximálně 100 000$ za dobu 12 měsíců [62]

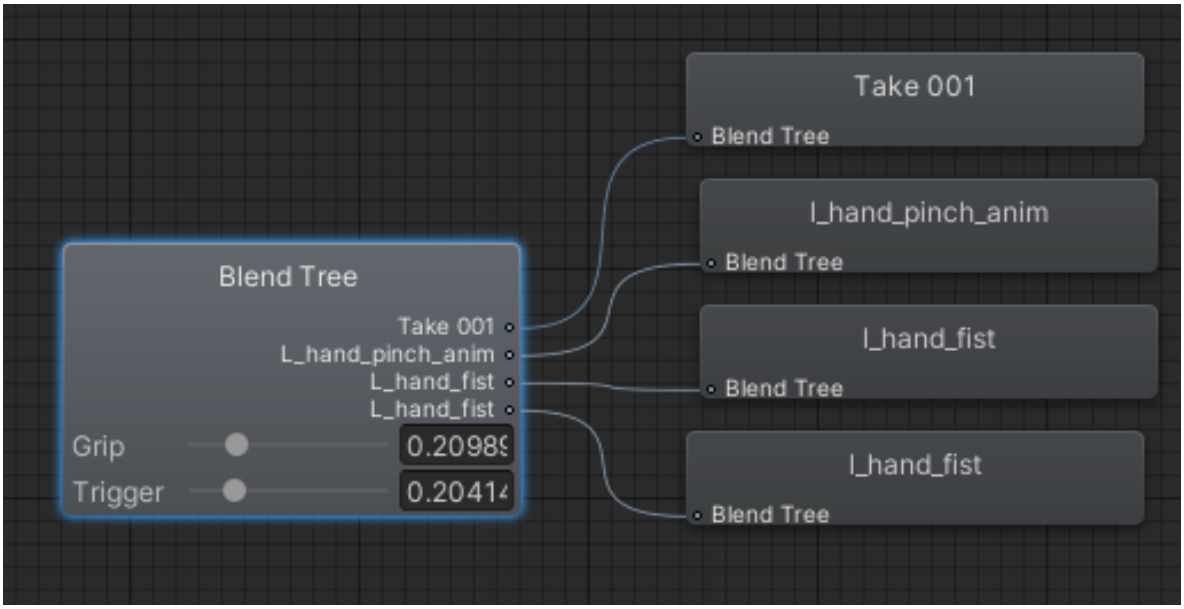
7.1 Šablona modelu

Projektová šablony je vytvořena v Unity ve verzi 2021.2.13f1, měla by však být funkční i ve vyšších verzích. Obsahuje přednastavené objekty pro kameru virtuální reality, informační prvky typické pro muzea, skripty řešící interakci a animaci rukou a nakonec nastavení projektu, zajišťující podporu ovladačů různých hardwarových výrobců.

Veřejný Github repozitář: https://github.com/Operator21/Virtual-Musem

7.1.1 Animátor stavů ruky

Pro správné zobrazení animací ruky je zapotřebí přidat ke každému ovladači představující ruku, animátor, jenž ovládá stav animace daného modelu. Klasicky se vytváří stavy, mezi kterými se přechází v rámci animátoru dle určených podmínek. Zde je však vhodné využít funkce Blend Tree, která umožňuje plynulé přechody mezi animačními stavy. Aby bylo možné kontrolovat přechody, musí být vytvořeny proměnné pro stisk ukazováčku (Trigger) a sevření pěsti (Grip). Blend Tree je také přepnut z 1D režimu na 2D Freeform Cartesian, v tomto režimu jsou přidány čtyři body ve čtvercovém rozložení, každý z nich představuje konečný stav animace.



Obrázek 20 - Struktura Blend Tree levé ruky

7.1.2 HandInit

Skript HandInit vytváří model ruky pro specifikovaný ovladač, začátek kódu vypadá následovně. Veřejné proměnné characteristics a handPrefab umožňují nastavení hodnot přímo v inspektoru Unity. Metoda Start() se pouští vždy při vytvoření instance, která tento skript obsahuje. V rámci spuštěni jsou přiřazeny hodnoty pro privátní proměnné instance, do které se vkládá herní objekt, vytvořený pomocí zabudované funkce Instantiate() a animator, jenž pomocí funkce GetComponent<>() získává referenci na komponent Animátor. Pomocí této reference je možné následně ovládat proměnné používané v daném Animátoru. Nakonec je vytvořen privátní list devices, který je též inicializován v metodě Start().

```
public InputDeviceCharacteristics characteristics;
public GameObject handPrefab;
private GameObject instance;
private Animator animator;
private bool found = false;
private List<InputDevice> devices;

void Start() {
 Debug.Log("Hand init starts");
 instance = Instantiate(handPrefab, transform);
 animator = instance.GetComponent<Animator>();
 devices = new List<InputDevice>();
}
```

Kód 1 - Inicializace modelu ruky

Zabudovaná metoda Update() se vyvolává při každém vykresleném snímku, je tedy vhodné v této metodě neprovádět příliš obsáhlé množství operací, mohly by vést k výraznému snížení plynulosti chodu aplikace. V tomto případě se pomocí metody GetDevicesWithCharacteristics() ve statické třídě InputDevices naplní list devices dle parametrů zadaných v inspektoru pomocí proměnné characteristics. Pokud je v listu detekováno alespoň jedno zařízení, spustí se metoda UpdateHandState pro první zařízení v listu. V tomto případě se počítá pouze s jedním možným zařízením s danými parametry a tak není nutné řešit další filtrování seznamu.

```
void Update() {
 InputDevices.GetDevicesWithCharacteristics(characteristics, devices);

 if(devices.Count > 0){
  UpdateHandState(devices[0]);
 }
}
```

Kód 2 - Získání veškerých aktivních zařízení dle parametrů

Metoda UpdateHandState() provádí kontrolu dvou vstupních hodnot z ovladače. Nejprve se pokouší zjistit sílu stisku přední páčky, pokud je úspěšná, uloží tuto hodnotu do proměnné typu float s názvem triggerValue. Tato hodnota je nakonec použita pro změnu proměnné v Animátoru s názvem Trigger, podle které se upraví aktuální stav animace modelu ruky. Druhá část je ve své podstatě totožná avšak pracuje s hodnotou stisku tlačítka

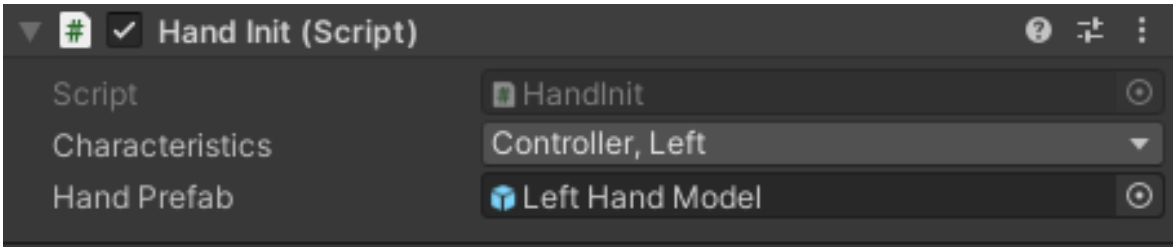
či kapacitní plochy (záleží na konkrétním druhu ovladače), ve virtuálním prostoru je to akce sevření zápěstí a uchopení objektu.

```
void UpdateHandState(InputDevice device){
 if(device.TryGetFeatureValue(CommonUsages.trigger, out float
triggerValue)){
  animator.SetFloat("Trigger", triggerValue);
 } else {
  animator.SetFloat("Trigger", 0);
 }

 if(device.TryGetFeatureValue(CommonUsages.grip, out float gripValue)){
  animator.SetFloat("Grip", gripValue);
 } else {
  animator.SetFloat("Grip", 0);
 }
}
```

Kód 3 - Úprava stavu animátoru dle stisku ovladače

Skript se využívá u dvou objektů, LeftHandSpawner a RightHandSpawner, které se následně využívají u XR soustavy. Takto vypadá přidaný skript v inspektoru, do políček se vyplňují výše definované public hodnoty. V tomto konkrétním příkladu se jedná o ovladač pro levou ruku.



Obrázek 21 - Zobrazení inspektoru levého ovladače

7.1.3 TeleportControl

Nejzásadnější skript zajišťující pohyb po virtuálním prostoru pomocí teleportace, ze začátku jsou v inspektoru zpřístupněny dva objekty baseController a teleportationObject.

BaseController a TeleportationObject

Jedná se prázdné objekty, jenž slouží výhradně pro řízení ovládání skrze komponent XR Controller zabudován v Unity. Vstupy v BaseController jsou ze začátku vždy aktivní, jakmile uživatel použije teleportační akci na ovladači, je BaseController deaktivován a na jeho místo přichází TeleportationObject.

Akční systém Unity

Novinkou v posledních verzích enginu se stal Action Based Input System, jenž odděluje logiku od vstupů umožňující jednoduchou správu ovládacích prvků. V rámci virtuální reality to znamená, že není zapotřebí řešit ovládání pro ovladač každého výrobce zvlášť.

V kódu níže jsou při inicializaci přiřazeny metody TeleportModeActivate() a TeleportModeCancel() k referenci akce teleportActivation.

```
public InputActionReference teleportActivation;
public UnityEvent onTeleportActivate;
public UnityEvent onTeleportCancel;

void Start() {
 teleportActivation.action.performed += TeleportModeActivate;
 teleportActivation.action.canceled += TeleportModeCancel;
}
```

Kód 4 - Přiřazení teleportačních akcí

Aktivační metoda vyvolává událost typu UnityEvent, která zapíná teleportační režim. Rušící metoda je rozdělena do dvou metod, jelikož je zapotřebí vytvořit krátkou prodlevu pomocí metody Invoke(), jinak by se teleportační akce nestihla provést.

```
void TeleportModeActivate(InputAction.CallbackContext obj){
 onTeleportActivate.Invoke();
}

void TeleportModeCancel(InputAction.CallbackContext obj){
 Invoke("DeactivateTeleporter", 0.1f);
}

void DeactivateTeleporter(){
 onTeleportCancel.Invoke();
}
```

Kód 5 - Vyvolání událostí při teleportaci

RayInteractorControl

Obdobným způsobem, jako skript pro ovládání teleportace, funguje kód pro zapínání a vypínání ukazovátka pro interakci s uživatelským rozhraním, informačními tabulemi, kvízy a podobnými operacemi prováděnými na dálku.

Narozdíl od teleportace se aktivace paprsku provádí v režimu přepínání pro jednoduchost užití. Kód funguje následovně, při prvním vytvoření instance s tímto skriptem je přiřazena metoda RayModeActivate() k provedené akci, v práci se jedná o stisk sekundárního tlačítka ovladače,nejedná se ovšem o standard, přepínání interakcí se liší dle preferencí daného vývojáře.

```
void OnEnable() {
 rayActivation.action.performed += RayModeActivate;
}
```

Kód 6 - Alternativa ke Start() metoda onEnable()

Samotná metoda nejprve vyvolá událost OnRayActivate vytvořenou v inspektoru a následně přenastavuje akci tak, aby při stisku provedla událost RayModeCancel().

```
void RayModeActivate(InputAction.CallbackContext obj) {
 onRayActivate.Invoke();
 rayActivation.action.performed -= RayModeActivate;
 rayActivation.action.performed += RayModeCancel;
}
```

Kód 7 - Změna akční události po jejím provedení

RayModeCancel() obsahuje naprosto totožné operace, jsou pouze invertované, což vede k neustálému přepínání metod při každém stisku tlačítka.

V inspektoru vypadají vyvolávané události obdobně jako u teleportace, při aktivaci je zapnut komponent XR Ray Interactor v objektu RayController a zároveň aktivován jeho interakční systém.

7.1.4 XR Camera Rig

Objekt zajišťující veškeré základní požadavky aplikace virtuální reality, tvoří ho struktura několika prázdných objektů typu Empty obsahující autorovy skripty společně se skripty z balíčku přímo od Unity. Struktura vypadá následovně:

- XR Camera Rig
  - Camera Offset
    - Main Camera
      - Fader
    - Right Hand
      - BaseController
      - RayController
    - Left Hand
      - BaseController
      - TeleportController

Součástí XR Camera Rig je skript XR Origin, jenž přijímá referenci na objekt tvořící hlavní strukturu, v tomto případě tedy odkazuje sám na sebe. Dále přijímá objekt určující odstup mezi kamerou a podlahou tedy objekt Camera Offset, nakonec přijímá referenci na kameru, která má být pro VR využita, přiřazena je zde základní Main Camera. Druhý aktivní skript, který se nachází v objektu, Input Action Manager, referuje pouze na obecné přednastavené akce VR ovladačů v akčním systému Unity.

V rámci Camera Offset se nachází Main Camera, jenž kromě standardních komponent obsahuje skript vstupního systému Tracked Pose Driver, který nastavuje pozici a rotaci kamery dle středu očí v headsetu, zároveň obsahuje potomka Fader, plochu určenou pro ztmavování pohledu kamery při kolizích.

Objekty pro ruce Left Hand a Right Hand si jsou velice podobné, liší se pouze referencemi na daný ovladač a svým zaměřením.

Levá (modrá) ruka slouží primárně k pohybu po scéně, obsahuje Teleport Control skript popsán v kapitole TeleportControl. Do veřejných proměnných jsou přidány podobjekty BaseController a TeleportController a reference na akci daného ovladače pro aktivaci teleportace. Nakonec jsou přidány události, které se odehrávají při aktivaci teleportace a při jejím zrušení. Během aktivace jsou zapnuty komponenty objektu TeleportController, umožňující zobrazení linie, značící místo přesunu. Zároveň jsou zablokovány vstupní akce základního ovladače, bez této blokace by mohlo dojít k nepředvídatelnému chování s interakčními objekty během teleportace. Při zrušení je BaseController vrácen do původního stavu a TeleportControl je zablokován.

Pravá (červená) ruka je definována přednostně jako interakční, operuje s uživatelským prostředím pomocí paprsku a při uchopování objektů má prioritu. Podobně jako levá obsahuje BaseController, avšak místo TeleportController obsahuje RayController, jenž vykresluje paprsek, který umožňuje interakci s popisky, kvízy a zvukovými expozicemi.

V objektech BaseController je přidán skript XR Controller pracující s akčním systémem, zde jsou definovány operace určené pro daný ovladač společně s 3D modelem jenž má ovladač symbolizovat, zde jsou využity prázdné HandSpawner objekty se skriptem HandInit. Další součásti BaseController zahrnují XR Direct Interactor, řešící interakci s objekty a Sphere Collider, jenž poskytuje detekci kolizí vyžadovanou pro funkčnost interakce.

TeleportController je v zásadě rozšířenou verzí BaseController s deaktivovanými vstupy, dokud není vyvolána teleportační operace, navíc je přidán komponent XR Ray Interactor, který je nastaven na vrstvu Teleport, tato vrstva se následně využívá u veškerých ploch určených k teleportaci. V komponentu je nastavena reference na XR Interaction Manager, základní skript, jenž je nutné mít vždy ve scéně, v opačném případě by nefungoval žádný typ interakce. XR Interaction Manager nepotřebuje žádné nastavení, jeho přidání do scény je tedy triviální. Další nastavení zahrnují spíše preference v zobrazení interakčního paprsku, pro teleportační účely využívá většina aplikací bezeriérovu křivku a rovný paprsek pro interakci s grafickým rozhraním, možnosti jsou různé a záleží na konkrétním užití. Dalším komponentem je Line Renderer, jenž může být zanechán tak jak je nastaven při přidání, jelikož hlavní nastavení jsou stejně následně nahrazena komponentem XR

Interactor Line Visual, kde se nastavuje především tloušťka interaktivní čáry, její barva při platné a neplatné interakci a případně i „zaměřovač“, ten jde ovšem nastavit i u interaktivních ploch a pro vyšší kontrolu se jedná o vhodnější přístup.

7.1.5 Locomotion System

Prázdný objekt, jenž je podobně jako XR Interaction Manager zásadní pro základní chod aplikace virtuální reality. Tento chod zajišťují tři skripty obsažené v balíčku OpenXR: Locomotion System, Teleportation Provider a Snap Turn Provider.

SkriptLocomotion System vyžaduje vyplnění dvou vstupních hodnot, první je Timeout, ovládající maximální možnou časovou délku exkluzivního připojení k danému pohybovému systému, čas se udává ve vteřinách. Druhou vstupní hodnotou je reference na XR Rig, tedy soustavu objektů, se kterou bude pohybový systém pracovat.

Teleportation Provider připravuje prostředí pro teleportační pohyb, přijímá jedinou vstupní hodnotu a to System, kde se odkazuje na výše zmíněný Locomotion System. V práci se tento skript nachází ve stejném objektu, odkazuje tedy na stejný objekt, ale na jiný komponent. Dle dokumentace je možné nechat toto políčko prázdné, skript se pokusí nalézt první validní systém, avšak rychlejším způsobem je uvést vazbu rovnou a vyhnout se tak možným problémům.

Posledním komponentem je Snap Turn Provider, jak již z názvu napovídá jedná se o skript poskytující možnost otáčení za použití ovladače místo pohybu tělem, funkce užitečná především ve chvílích, kdy není headset bezdrátový a uživatel se při excesivním otáčení zamotává do kabelů. Podobně jako u teleportace, je nutné referovat na pohybový systém, v rámci nastavení se také může upravit úhel otáčení a doba čekání mezi každou otáčkou. Dále je možné aktivovat či deaktivovat otáčení na určitou stranu a nakonec se musí určit, které akce toto otáčení vyvolávají. Standardně se jedná o páčku pravé ruky, levá ruka je většinou vynechávána jelikož je již používána k teleportaci.

7.1.6 Interaktivní objekty

V rámci interakce se rozlišují dva druhy objektů, teleportační plochy a objekty, které je možné vzíti do rukou či jinak s nimi manipulovat pomocí ovladačů.

Teleportační plocha většinou zahrnuje podlahu ve virtuálním scéně, může být dvou typů: kotva nebo oblast. Kotva označuje stacionární pozici, do které je možné se teleportovat pouze na střed a následně se hýbat v reálném prostoru dle rozsahu plochy kotevního modelu. Oblastní plocha dovoluje teleportaci po celém rozsahu modelu. Způsob užití záleží na konkrétní situaci, typicky se však kotva využívá v aplikacích, kde není obsah vytvořen pro volný pohyb, kupříkladu virtuální režim v Google mapách využívá pohyb typu kotva. Oproti tomu velká převaha her využívá oblastní systém.

Při nastavování manipulace s objekty je zásadní, aby dané instance obsahovaly kolizní pole a skript XR Grab Interactible se správně nastavenou interakční vrstvou. Vhodné je také zmínit Rigidbody systém, jenž umožňuje objektům využívat fyzikálního modelu Unity. Pro realistickou interakci (a tím i navýšení imerze) je tento komponent nutností.

7.1.7 Generování kvízů

Podstatnou součástí procesní modelu je vytváření jednoduchých kvízů, které se mohou vyplňovat přímo ve virtuální scéně. Základní bloky této funkcionality tvoří třídy QuestionClass a QuizClass.

QuestionClass

Základní objekt tvořící kvíz, zahrnuje parsovací metodu pro jednoduchý vstupní formát při vytváření, díky kterému jsou v inspektoru zadávány otázky ve formátu Otázka : Správná Odpověd, Špatná odpověd. Za první špatnou odpovědí je možné následně přidávat další špatné odpovědi oddělené čárkou. Metoda nejprve přijatý string zkontroluje, zda-li je ve správném formátu a poté ho rozdělí na pole dle znaku „**:**“. První část pole je přidělena do privátní proměnné question a probíhá další kontrola, tentokrát se ověřuje přítomnost odpovědí. Nakonec probíhá cyklus, jenž přiřadí první správnou odpověd do proměnné correctAnswer a zbytek uloží do stringového listu špatných odpovědí wrongAnswers.

```
private void setValuesFromString(string dataStructure) {

 if(!dataStructure.Contains(":"))
  return;
 string[] split = dataStructure.Split(":");
 this.question = split[0];

 if(split[1].Length < 1)
  return;
 wrongAnswers = new List<string>();
 string[] extractedAnswers = split[1].Split(",");

 for(int x = 0; x < extractedAnswers.Length; x++){
  if(x == 0)
   this.correctAnswer = extractedAnswers[0].Trim();
  else
   this.wrongAnswers.Add(extractedAnswers[x].Trim());
 }
}
```

Kód 8 - Parsovací systém otázek

Z třídy jsou přístupné pouze dvě neměnitelné get hodnoty a to Question, vracející textovou formu otázky a list stringů, jenž kombinuje správnou odpověď se špatnými a náhodně zamíchá jejich pořadí pomocí statické třídy Random.

```
public List<string> Answers {

 get {
  List<string> tmp = new List<string>();
  tmp.AddRange(this.wrongAnswers);
  tmp.Add(this.correctAnswer);
  return tmp.OrderBy(a => Random.Range(int.MinValue,
int.MaxValue)).ToList();
 }
}
```

Kód 9 - Návrat odpovědí v náhodném pořadí

QuizClass

Komponent, jenž zajišťuje funkcionalitu kvízů, používá se u objektů grafického rozhraní Canvas. Oproti ostatním skriptům se jedná o obsáhlý kód, řeší se zde generace tlačítkových odpovědí dle zadaných otázek a vyhodnocení úspěšnosti průchodu kvízem. Počet proměnných je relativně rozsáhlý, patří mezi ně herní objekt buttonPrefab, jenž určuje strukturu tlačítka, dále z inspektoru přístupný list otázek pro jednoduché zadávání přímo z rozhraní Unity, ze zadaných stringů se extrahované hodnoty předávají do listu objektů QuestionClass. Následují textová pole pro samotný obsah otázky a zobrazení skóre, několik číselných hodnot pro uchování posledních pozic tlačítek, aktuální otázky, skóre a nastavitelná výška tlačítek. Nakonec je přidána jako bonus booleanovská proměnná, určující způsob zobrazení skóre, pokud je hodnota true, zobrazí se v procentech, pokud false tak ve formátu „*počet správných otázek/celkový počet otázek*“.

```
public GameObject buttonPrefab;

public List<string> questions;
private List<QuestionClass> questionList;
private TextMeshProUGUI questionText;
private TextMeshProUGUI scoreText;
private Transform scrollContent;
private int posCurrent = 0;
private int currentQuestion = 0;
private int score = 0;
public int buttonHeight = 35;
public bool showInPrecentage = false;
```

Kód 10 - Proměnné zajišťující generování kvízu

Při spuštění je inicializováno pole otázek, uložených v objektech, zároveň se automaticky vyhledají textová pole dle jejich názvů s pomocí funkce Find(). Stejným způsobem se získává Content, objekt, do kterého se přidávají vygenerovaná tlačítka. V dalším kroku probíhá cyklus listem stringů zadaných v inspektoru a vytváří z nich objekty díky parsovací funkci v konstruktoru a rovnou je přidává do listu otázek. Nakonec je zobrazena aktuální otázka, při začátku skriptu se vždy jedná o první v listu.

```
void Start() {
 questionList = new List<QuestionClass>();

 scoreText = this.transform.Find("Score").GetComponent<TextMeshProUGUI>();

 questionText =
this.transform.Find("Question").GetComponent<TextMeshProUGUI>();

 scrollContent = this.transform.Find("Scroll View/Viewport/Content");

 foreach(string questionStructure in questions){
  questionList.Add(new QuestionClass(questionStructure));
 }
 DisplayQuestion(questionList[currentQuestion]);
}
```

Kód 11 - Nalezení textových komponentů potomků dle jejich názvů

Samotná metoda na zobrazení otázky funguje relativně triviálně, v textovém poli je zobrazena samotná otázka, pozice prvního tlačítka začíná nulou a provádí se funkce ClearContent(), jenž odstraňuje veškerá tlačítka předchozí otázky z kontejneru, zároveň také upravuje text aktuálního skóre. Generování odpovědí se provádí ve foreach cyklu, procházející veškeré textové hodnoty v get listu Answers. Při každém průběhu je vytvořen nový herní objekt – tlačítko, kterému je daný string přiřazen. Následně se tlačítku přiřazuje událost, která se provádí při jeho zakliknutí. Po stisku tlačítka se nejprve zkontroluje, zda-li zvolená odpověď odpovídá správné, pokud tomu tak je, přidá se bod do skóre. Dále se provádí funkce NextQuestion(), pokud existuje další otázka, přepne na ní a vrátí hodnotu true, jinak vrátí false. Jakmile neexistuje žádná další otázka, změní se text pole a je přidáno restartovací tlačítko, které kvíz spustí od začátku. Na úplném konci metody se upravuje rozměr kontejneru tak, aby odpovídal počtu tlačítek s určenou výškou.

```
public void DisplayQuestion(QuestionClass question) {
 questionText.text = question.Question;
 posCurrent = 0;
 ClearContent();
 foreach(string answer in question.Answers) {
  GameObject instance = CreateButton(answer);
  instance.GetComponent<Button>().onClick.AddListener(() => {
   if(instance.transform.Find("Text
(TMP)").GetComponent<TextMeshProUGUI>().text ==
questionList[currentQuestion].CorrectAnswer)
   score+=1;
   if(!NextQuestion()){
    questionText.text = "Quiz finished";
    posCurrent = 0;
    ClearContent();
    GameObject button = CreateButton("Try Again");
    button.GetComponent<Button>().onClick.AddListener(() => {
     ResetQuiz();
    });
   }
  });
 }
 scrollContent.transform.GetComponent<RectTransform>().sizeDelta = new
Vector2(0, question.Answers.Count * buttonHeight);
}
```

Kód 12 - Metoda vykreslující otázku z objektu třídy QuestionClass

Metoda NextQuestion() vrací po svém provedení booleanskou hodnotu, pokud existuje další otázka v pořadí, je zobrazena metodou DisplayQuestion() a vrací se hodnota true. V opačném případě se jednoduše vrátí false a žádné další operace nejsou provedeny.

```
public bool NextQuestion() {

 if(currentQuestion+1 < questionList.Count) {
  DisplayQuestion(questionList[++currentQuestion]);
  return true;
 }
  return false;
}
```

Kód 13 - Funkce na kontrolu zobrazení další otázky

Funkce pro vytvoření tlačítka přijímá textovou reprezentaci odpovědi, nejprve je vytvořena instance z referované předlohy tlačítka na nulté pozici bez rotačních úprav. Dále se tato instance nastavuje jako potomek kontejneru Content, toto přiřazení je nutné pro každý prvek. Bez tohoto přiřazení by se vytvořená tlačítka vytvářeli v kořenu scény, což by způsobilo chyby v pozicování ve scéně. V dalším kroku se upravuje horní odsazení instance, ze začátku je nulové a s každým přibývajícím tlačítkem se zvyšuje o nastavenou výšku tlačítka. Nakonec je textovému obsahu přiřazena odpověď z parametru a reference na instanci je vrácena pro další použití.

```
private GameObject CreateButton(string content){
 GameObject instance = Instantiate(buttonPrefab, new Vector3(0, 0, 0), new
Quaternion());
 instance.transform.SetParent(scrollContent, false);

instance.transform.GetComponent<RectTransform>().SetInsetAndSizeFromParentE
dge(RectTransform.Edge.Top, posCurrent, buttonHeight);
 instance.transform.GetComponent<RectTransform>().sizeDelta = new
Vector2(0, buttonHeight);
 posCurrent += buttonHeight;
 instance.transform.Find("Text (TMP)").GetComponent<TextMeshProUGUI>().text
= content;
 return instance;
}
```

Kód 14 - Funkce vytvářející a následně vracející tlačítko s parametrem odpovědi

Metoda pro čistění obsahu kontejneru využívá speciálního chování typického pro Unity, v klasické .NET implementaci jazyku C# by tento postup nefungoval. V enginu je možné vybrat veškeré komponenty typu Transform jen za pomoci cyklu foreach, během studia dokumentací a fór nebyl nalezen jiný způsob, jenž by toto chování umožňoval. Kromě této skutečnosti se jedná o základní cyklus, jenž maže potomky podle nalezených komponentů Transform. Ke konci je provedeno přiřazení aktuálního skóre k textovému poli, umístění dané operace do této metody je určené nutností při každé čistce obsahu aktualizovat aktuální skóre.

```
private void ClearContent(){
 foreach(Transform child in scrollContent.transform)
  Destroy(child.gameObject);
 scoreText.text = currentScore;
}
```

Kód 15 - Metoda čistící obsah kontejneru tlačítek

Metoda ResetQuiz() pouze obnovuje proměnné na původní hodnoty a volá funkce ClearContent() a DisplayQuestion(). Zajímavější je však string zobrazující skóre, nejedná se o prostou proměnnou ale o get funkci, jenž dle nastavení v inspektoru vrací dva druhy skóre. Pokud je zaškrtnuto políčko showInPercentage, zobrazí se procentuální úspěšnost namísto klasického bodového hodnocení.

```
private string currentScore {
 get {
  if(showInPrecentage){
   if(score == 0)
    return "Success: 0%";
   return "Success: " + (score/(questionList.Count/100.0)) + "%";
  }
  return score + "/" + questionList.Count;
 }
}
```

Kód 16 - Proměnná typu get, vracející aktuální skóre dle nastaveného formátu

7.1.8 Tvorba popisků

Popisky jsou ústředním prvkem muzeí, poskytují podstatné informace o objektu a převážnou většinu času se nachází poblíž nich, skript jenž tuto funkcionalitu zajišťuje se jmenuje InfoBoardClass. Proměnných se v tomto kódu nachází značné množství, textové hodnoty headerText a contentText přijímají informace, které se zobrazí uživateli v samotném popisku. Parametr umožňuje vytvořit v inspektoru Unity rozsáhlejší vstupní pole pro text.

```
public string headerText;

[TextArea(3, 10)]
public string contentText;
public GameObject mainObject;
public TextMeshProUGUI headerMesh;
public TextMeshProUGUI contentMesh;
public TextMeshProUGUI indicatorMesh;
[Range(0,10)]
public float heightFromObject;
[Space]
public UnityEvent Activate;
public UnityEvent Deactivate;
public UnityEvent Hide;
public UnityEvent Show;
private GameObject player;
```

Kód 17 - Začátek třídy InfoBoardClass s parametry nad proměnnými ovlivňující inspektor

Během prvního spuštění skriptu jsou provedena přiřazení textových hodnot k textovým objektům, zároveň se vyhledá hlavní kamera a uloží se do proměnné player pro budoucí užití. Za běhu se provádí změna vektoru určujícího pozici popisku tak, aby odpovídal pozici daného objektu

Součástí popisků je funkce skrytí, pokud je hlavní kamera příliš daleko od popisku, je skryt pod viditelným znakem otazníku symbolizující místo zájmu. Toto přepínání je řešeno v rámci metody Update().Provádí se zde porovnání vypočítané vzdálenosti kamery od objektu, ke kterému je připojen. Ve vzdálenostech menších jak 0.8 je skryt informační objekt kompletně, aby nepřekážel uživateli v prohlížení si objektu zblízka. Aby se vytvořila jistá rezerva ve vzdálenosti, je opětovné zobrazení nastaveno na 1. Dle

nastavitelné vzdálenosti se nakonec určuje, kdy se má nahradit otazník samotnou informační tabulí.

```
void Update(){
 Vector3 newPosition = mainObject.transform.position;
 newPosition.y = heightFromObject;
 transform.position = newPosition;
 distance = Vector3.Distance(gameObject.transform.position,
mainCamera.transform.position);
 if(distance > 1)
  Show.Invoke();
 if(distance > hideTextDistance)
  Deactivate.Invoke();
 if(distance <= hideTextDistance)
  Activate.Invoke();
 if(distance < 0.8f)
  Hide.Invoke();
}
```

Kód 18 - Kontrola vzdálenosti uživatele od popisku

7.1.9 Přizpůsobení rozměrů a rotace ke kameře

Aby se zabránilo nečitelnosti textu na různé vzdálenosti a pozice, byl vytvořen skript ScaleToDistance, jenž umožňuje ovládat velikost objektu dle relativní vzdálenosti k hlavní kameře, textová pole se tak mohou zvětšovat a zmenšovat a uživatel tak vidí celý obsah bez větších obtíží. Rozměry se mění při každém snímku v rámci metody Update(), nejprve se zjišťuje trojrozměrný vektor obsahující aktuální pozici kamery, dále se vypočítává vzdálenost mezi pozicí aktuálního objektu a hlavní kamery, tato vzdálenost se následně využívá ve funkci matematické knihovny Mathf.Clamp(), jenž vrátí desetinou hodnotu dle daných mezí. Je-li vzdálenost v rámci mezí, je vrácena, jinak se vrací samotná mez, která byla překročena. Nakonec se provádí rotace objektu tak, aby vždy mířil směrem na kameru.

```
void Update() {

 Vector3 playerPosition =
GameObject.FindGameObjectWithTag("MainCamera").transform.position;
 float distance = Vector3.Distance(gameObject.transform.position,
playerPosition);
 gameObject.transform.localScale = new Vector3(
Mathf.Clamp(distance/step, minScale, maxScale),
Mathf.Clamp(distance/step, minScale, maxScale),
Mathf.Clamp(distance/step, minScale, maxScale)
 );
 if(rotateTowardsPlayer)
  transform.rotation = Quaternion.LookRotation(transform.position -
playerPosition);
}
```

Kód 19 - Užití matematické knihovny pro úpravy rozměrů v rámci mezí

7.1.10 Řešení kolizí kamery

Detekce kolizí kamery v rámci virtuální reality má řadu různých řešení, společně se svými výhodami i nevýhodami. Klasický přístup u nevirtuálních aplikací, tedy zablokování možnosti pohybu skrz objekty ve virtuálním prostoru, je lehce pokořitelný pouhý pohybem v reálném prostoru. Jedním z používaných řešení v praxi je vytvoření vlastní logiky pro vrácení uživatele zpět na pozici, ve které byl před detekcí kolize kamery s objektem, tento přístup funguje, avšak často vede k dezorientaci uživatele či dokonce k vyvolání nevolnosti. Druhou prakticky používanou technikou je ztmavení obrazovky, pokud se uživatel rozhodne vejít do zdi, je mu jasně indikováno, že tento směr není platný, i přestože se samotnému procházení nezabrání, uživatel je krátkodobou ztrátou vize donucen ze zdi vyjít. Tato metoda je v práci implementována skriptem CameraFadeOut.

Z důvodu limitací Open XR kamery, není možné použít vrstvu Post Processing na ztmavení obrazu kamery. Ekvivalentního chování se však může dosáhnout vytvořením objektu plochy s černým materiálem, jenž naprosto ignoruje osvětlení scény. Plocha nastavená v hierarchii jako potomek následuje pohyb kamery a blokuje výhled, pomocí skriptu se ovládá alfa kanál materiálu.

Při načtení scény se automaticky najde stmívací plocha a uloží se na ní reference. Kamera má sférické kolizní pole společně s Rigidbody komponentem, je tedy jednoduché zachytit srážky skrze metody OnCollisionEnter() a OnCollisionExit(). Tyto metody dle počtu aktuálních kolizí vyvolávají asynchronní metody Fadein() a Fadeout().

Při detekci kolize je zjištěno, zda-li již ztmavení neproběhlo, pokud ne, provede se a tento fakt je uložen do proměnné collision.

```
private void OnCollisionEnter(Collision other) {
 if(!collision) {
  collision = true;
  Fadein();
 }
}
```

Kód 20 - Vstupní kolizní detekce před ztmavením

Jakmile objekt opustí kolizní pole kamery, probíhá kontrola, zda-li se jednalo o jediný objekt, jenž se v poli nacházel, pokud ano, zavolá se metoda, jenž postupně kompletně zprůhlední plochu.

```
private void OnCollisionExit(Collision other) {
 if(other.contactCount < 1) {
  collision = false;
  Fadeout();
 }
}
```

Kód 21 - Výstupní kolizní detekce po ztmavení

Při ztmavování se nejprve nastavuje barva na průhlednou aby se kód vyhnul případným chybám z předchozích nastavení materiálu, následně je objekt ve scéně aktivován. Cyklem se postupně přidává hodnota alfa kanálu s menší časovou prodlevou mezi každou změnou.

Po skončení cyklu a krátké pauzy je alfa kanál pro jistotu nastaven na maximální hodnotu. Inverzním způsobem funguje asynchronní metoda Fadeout().

```
async void Fadein() {
 faderMaterial.color = new Color(0, 0, 0, 0);
 fader.SetActive(true);
 for(int x = 1; x < 100; x += step) {
  faderMaterial.color = new Color(0, 0, 0, x/100f);
  await Task.Delay(delay);
 }
 await Task.Delay(delay);
 faderMaterial.color = new Color(0, 0, 0, 1);
}
```

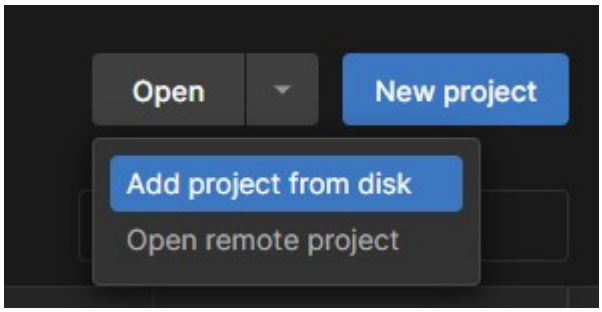
Kód 22 - Metoda pro stmívání kamery

7.2 Metodika modelu

Model obsahuje několik předpřipravených objektů, které zajišťují základní funkcionalitu scény virtuální reality, v některých případech, především v kapitole interaktivních objektů,je nutné do inspektoru dodat několik referencí. Všechny tyto povinnosti jsou zde rozepsány.

7.2.1 Otevření šablony

Jak již bylo zmíněno v kapitole Šablona modelu, procesní model funguje na verzi Unity2021.2.13f1 (případně vyšší). Pro použití je nutné mít nainstalovanou aplikaci Unity Hub a exportovaný procesní model na disku. Pro přidání projektu se v Hubu nachází tlačítko Open → Add project from disk, vybraný adresář musí obsahovat minimálně složky ProjectSettings, Packages a Assets.

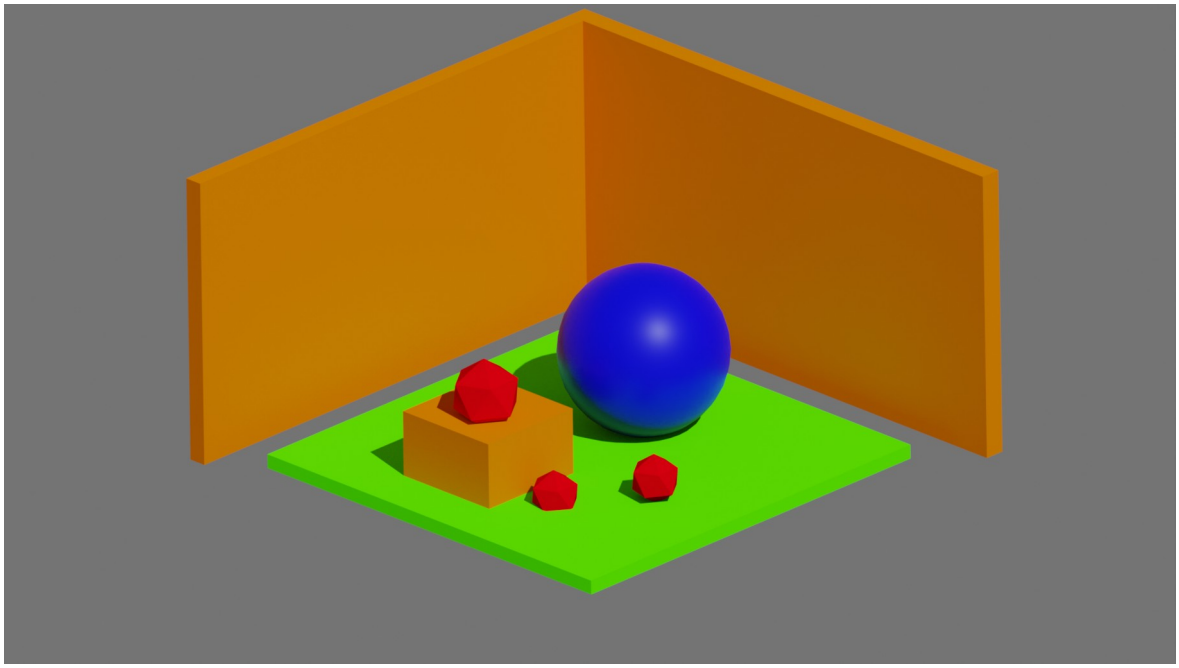


Obrázek 22 - Přidání projektu z disku

Pro otevření projektu stačí kliknout na jeho název. Procesní model obsahuje pouze zdrojové soubory, zbylé pomocné soubory si Unity generuje samo a proto může prvotní načtení projektu trvat několik minut.

7.2.2 Základní VR scéna

V kapitole Prostředí virtuální reality již byla zmíněna základní struktura scény virtuální reality společně s tímto schématem.



Obrázek 23 - Schéma základní virtuální scény

Procesní model poskytuje čtyři části zmíněné struktury:

- **Teleportační plochy**
  - Basic Floor
  - Basic Anchor
- **Interaktivní objekty**
  - Floating Reset Button
  - Info Board
  - Quiz
  - Sound Player
- **Uživatelské prvky**
  - XR Camera Rig
  - Locomotion System
- **Statické objekty**

- Table – Steel
- Table – Wood
- Wall – Bricks
- Wall - Plaster

Kvůli způsobu strukturizace objektů v enginu Unity, není možné v procesním modelu předpřipravit abstraktní interaktivní objekt (oranžová), v navazujících kapitolách je ale popsána metoda jejich tvorby s předpřipravenými komponenty.

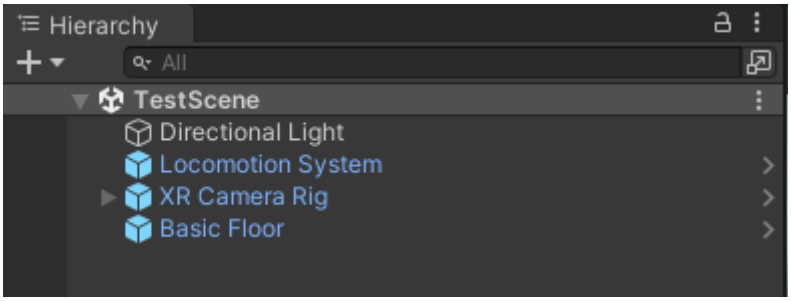
Podloží (šedá) také nelze logicky předpřipravit, jelikož se jedná o 3D model (skupinu modelů), který je závislý na konkrétní tématice aplikace.

Poskytnuté statické objekty v kombinaci s přiloženými materiály pro podlahu a primitivy vytvořitelnými přímo v enginu by měli být více než dostačující pro tvorbu jednoduché scény virtuálního muzea.

Tvorba základní scény

Vytvořit jednoduchou funkční scénu v rámci modelu je velice snadné, pokud se pracuje se základní scénou Unity, musí být nejprve odstraněny všechny objekty, kromě Directional Light, z hierarchie. Hlavní kamera, jenž se v každé scéně nachází, totiž koliduje s předpřipravenou VR kamerou a engine neví kterou z „hlavních kamer“ použít. Bezpečnější je tedy zbavit se všeho kromě osvětlení.

Je doporučeno přetáhnout jako první do scény objekt Basic Floor a nastavit jeho pozici na všech osách (x, y, z) na 0, dále přetáhnout XR Camera Rig na danou plochu a nakonec Locomotion System. Hierarchie scény by měla vypadat následovně:



Obrázek 24 - Základní topologie VR scény

Tyto objekty by měli automaticky detektovat veškeré potřebné reference a scéna by měla být v tomto složení funkční, pokud však ne, musí být reference nastaveny manuálně. Tento proces je popsán v navazující části.

Manuální nastavení referencí v případě chyb

V případě nefunkčnosti VR scény je možné, že některé z komponent nenalezly potřebné objekty nutné pro chod scény, pokud nefunguje pohyb teleportací, či otáčení do stran, je možné že Locomotion systém nerozpoznal kamerový set, v tom případě stačí přejít do objektu Locomotion System a do stejnojmenného skriptu přetáhnout objekt XR Camera Rig do políčka XR Origin. V případě, že nefunguje pouze teleportace, pak pravděpodobně nerozpoznal objekt teleportační oblasti interakční systém, k nápravě je nutné vybrat objekt podlahy (Basic Floor) a do políčka Interaction Manager pod komponentem Teleportation Area přetáhnout objekt Locomotion System.

7.2.3 Interaktivní VR objekt

Dle schématu scény výše jsou interaktivní objekty dvou typů, teleportační plochy a objekty uchopitelné virtuálníma rukama. Teleportační plocha se dále dělí na oblastní a kotevní, jejich využití se liší dle situace.

Teleportační plocha

I přestože přednastavená plocha je funkční, mohou nastat situace, kdy je zapotřebí použít jiní tvar či vlastní 3D model, pokud má například uživatel vytvořenou mapu v modelovacím softwaru a chce ji využít jako virtuální prostředí, měl by nejdříve oddělit plochy, na které by mělo být možné se teleportovat, od zbylých objektů v rámci modelu. Toto rozdělení zamezuje teleportaci po zdech, jenž dle autorových znalostí není

standardem v žádné VR aplikaci. Jak již bylo zmíněno, jsou dva typy teleportačních skriptů, pokud se má uživatel aplikace volně pohybovat po dané ploše, tak výběr míří na Teleportation Area skript. Pokud se jedná o menší objekt, jenž slouží spíše jako podstava nebo se uživatel nemá teleportovat po celé ploše, ale „uchytit“ na prostředek objektu, pak se volí Teleportation Anchor. Nastavení těchto ploch je následně totožné, v políčku Interaction Layer Mask musí být zaškrtnuta pouze vrstva Teleport, dále Custom Reticle určuje podobu zaměřovače teleportace, toto může být zanecháno prázdné anebo obsahovat 3D model, práce pro tento případ obsahuje jednoduchý zaměřovač zvaný TeleportTarget ve složce Objects, jenž je možné volně využít. Zbylá nastavení se mohou nechat tak jak byla při přidání skriptu.

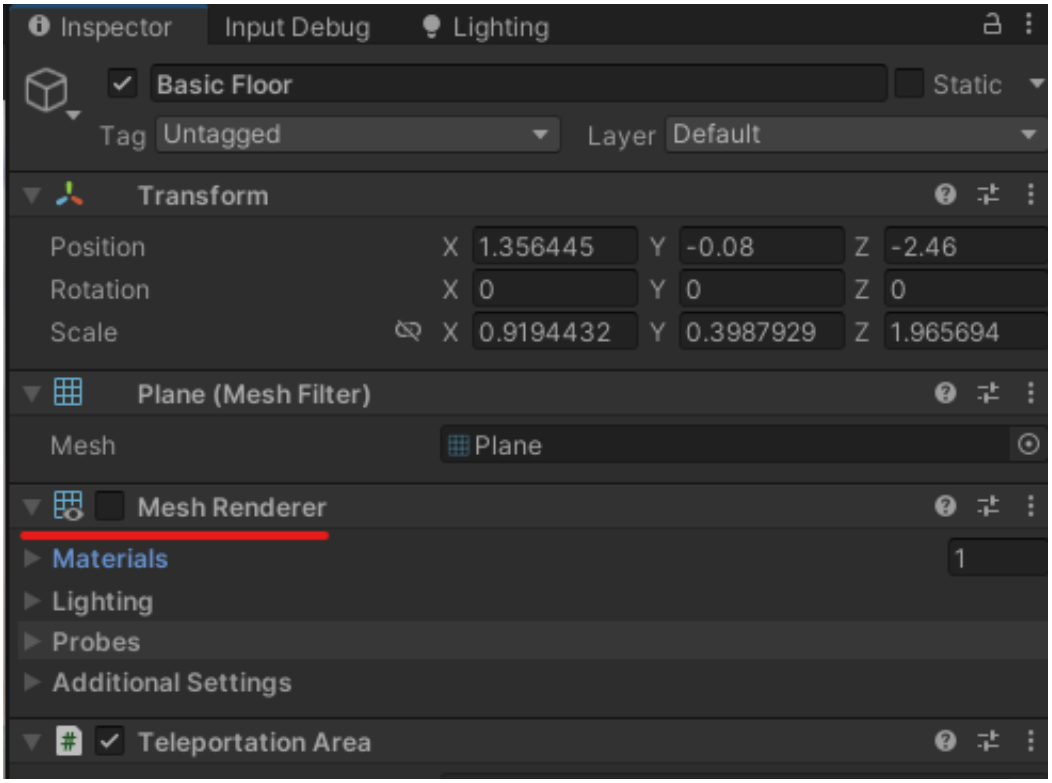
Kolizní pole

Jakmile je vlastní objekt nastaven na teleportaci, přichází otázka tvaru objektu, závisí na tom totiž tvar kolizního pole. Pokud se jedná o jednoduchý tvar, kupříkladu kostka, muže se využít jedné či kombinace kolizních komponent jednoduchých typů – Box Collider, Capsule Collider a Sphere Collider. S těmito typy se dá jednoduše operovat a nezpůsobují problémy s výkonem či nepředvídatelným chováním. Alternativou jsou typy Mesh Collider a Terrain Collider, tyto typy se využívají u topologií, kde jsou detaily příliš rozsáhlé na to, aby se vytvářeli kolizní pole skládáním jednoduchých tvarů, Mesh Collider není dobré využívat příliš často, jelikož dle složitosti topologie může být výrazně náročný na výkon aplikace a dle tvaru modelu způsobovat problémy s fyzikou aplikace. Terrain Collider se využívá zásadně u speciálního objektu Terrain, jenž Unity nabízí. Umožňuje vytvářet terén pomocí štětců a je podle toho optimalizován pro toto konkrétní užití.

Při tvorbě ukázkového řešení se Terrain Collider ukázal jako silně nespolehlivý a není tedy doporučeno ho využívat. Mesh Collider obsahuje v nových Verzi Unity optimalizační funkci, která sice zamezuje ztrátě výkonu, je však prakticky nepoužitelný u objektů, ve kterých se vyskytují mezery.

Vytváření neviditelných teleportačních ploch

Není nutností aby byla teleportační plocha viditelná ve scéně, zásadní je pouze kolizní pole, v ukázkovém řešení jsou hojně využívány přednastavené teleportační plochy s menší úpravou, jsou u nich vypnuty Mesh Renderer komponenty, které vykreslují modely ploch.



Obrázek 25 - Deaktivace zobrazení modelu u teleportačních ploch

Díky této technice se mohou rychle vytvářet teleportační území na modelu terénu, u kterého není žádaný kompletně volný pohyb a kotevní typ je až příliš limitující.

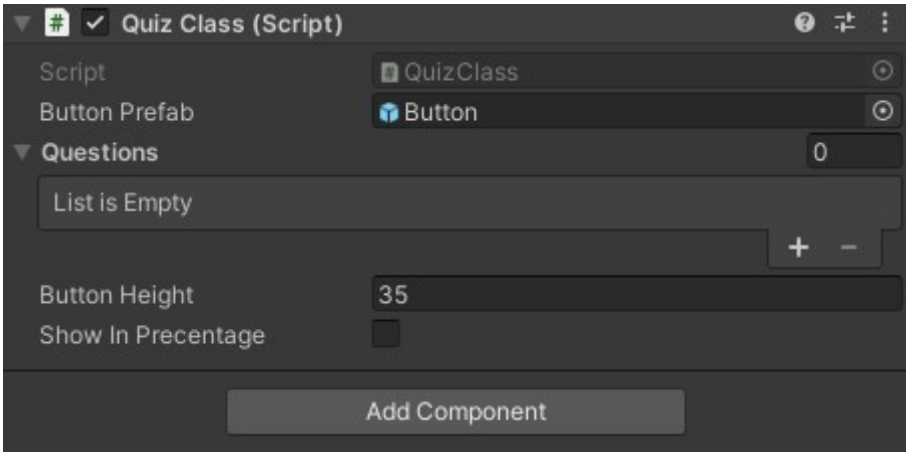
Uchopitelný objekt

Tento typ objektů je ze strany modelu obtížnější předpřipravit, jelikož se každý typ liší dle tvaru meshe a místa kde má být objekt uchopen. Nutné je tedy vlastní model přesunout do scény a přidat mu kolizní pole, zmíněné výše, s co možná nejjednodušším tvarem, aby se aplikace vyhnula případným problémům s fyzikou. Ta se aktivuje přidáním komponentu Rigidbody do objektu, díky tomu začne reagovat na gravitaci a patřičně interagovat s objekty, které mají nastavené kolizní pole. V rámci Rigidbody je vhodné nastavit políčko Mass, jedná se o objemní veličinu objektu která ovlivňuje rychlost dopadu tělesa v gravitaci, nejedná se však přímo o hmotnost, jelikož tato jednotka je nezávislá na síle gravitace. Je také možné zabránit změně pozice či rotace v nastavených osách pomocí constraint. Tato funkce se ale využívá spíše u dvoudimenzionálních platformních her. Dalším komponentem, jenž už zajišťuje samotnou interakci s ovladači je XR Grab Interactible. V tomto skriptu je vhodné věnovat pozornost zaškrtávacím políčkům Smooth

Position a Smooth Rotation, člověk se při svém pohybu značně třese i když si to neuvědomuje, zmíněné funkce tento problém řeší zahlazením pohybu. Posledním důležitým nastavením je Attach Transform, v základním nastavení je toto políčko prázdné a kvůli tomu se jakýkoliv objekt vzatý do ruky přesune svým středem na střed ruky daného ovladače. Pokud je přidán jiný Transform objekt do již zmíněného políčka, bude využit jeho střed. Dobrým způsobem jak tohoto využít je vytvořit objekt typu Empty, nastavit ho jako potomka interaktivního objektu, změnit jeho pozici na místo, kde je žádané aby se za objekt chytalo a nakonec nastavit referenci ve skriptu na Transform vytvořeného potomka. Nevýhodou tohoto nastavení je fakt, že lze určit pouze jediný objekt na tento typ zachytávání. Objekt se vždy přesune dle potomka, nezávisle na úhlu ve kterém byl objekt vzat.

7.2.4 Vytvoření kvízu

Pro vytvoření kvízu ve scéně byl pčipraven objekt Quiz ve složce Prefab. Tento objekt stačí přetáhnout do scény, upravit jeho pozici aby byl viditelný pro uživatele a v inspektoru nastavit komponent QuizClass dle preferencí. V základě vypadá nastavení skriptu takto:



Obrázek 26 - Možnosti nastavení kvízu

V tomto nastavení se sice kvízové okno zobrazuje, není však plně funkční, jelikož neobsahuje žádné otázky. Pro přidání otázek se může použít tlačítko plus či změnit nulu na číslo určující množství žádaných otázek. Tím se vytvoří textová pole do, kterých se již zapisují otázky.

Zadávání otázek

Pro co nejkompaktnější způsob zadávání byl vytvořen parsovací systém, jenž přijímá otázky v následujícím formátu:

Otázka : Správná Odpověd, Špatná odpověd, Špatná odpověd, ...

Správná odpověď je vždy po samotné otázce a odděluje se dvojtečkou, další odpovědi nejsou podmínkou pro fungování, stačí tedy pouze správná odpověd. Pro potřeby kvízu je však vhodné neukazovat vždy jen správnou odpověď, další možné odpovědi se oddělují čárkou a může jich být libovolné množství (omezeno samozřejmě kapacitou paměti zařízení). Porovnání správné odpovědi probíhá znakově, shodují-li se správná odpověď i špatná, je vyhodnocena jako správná, příklad:

1 + 1 = 2 : Ano, Ano, Ne

Jakékoliv „Ano“ je v tomto případě platné. Mezery mezi oddělovači nejsou podmínkou, systém je při zpracování odstraňuje, proto není problém si vytvářet větší mezery pro přehlednost mezi částmi vstupu.

Příklady platných zápisů

Jaké je hlavní město ČR? : Praha, Brno, Ostrava

17 < 5:Nepravda,Pravda

Tato práce si zaslouží za 1 : Ano , Ano , Ano , Ano

Příklady neplatných zápisů (nezpůsobí pád aplikace ale kvíz nebude funkční)

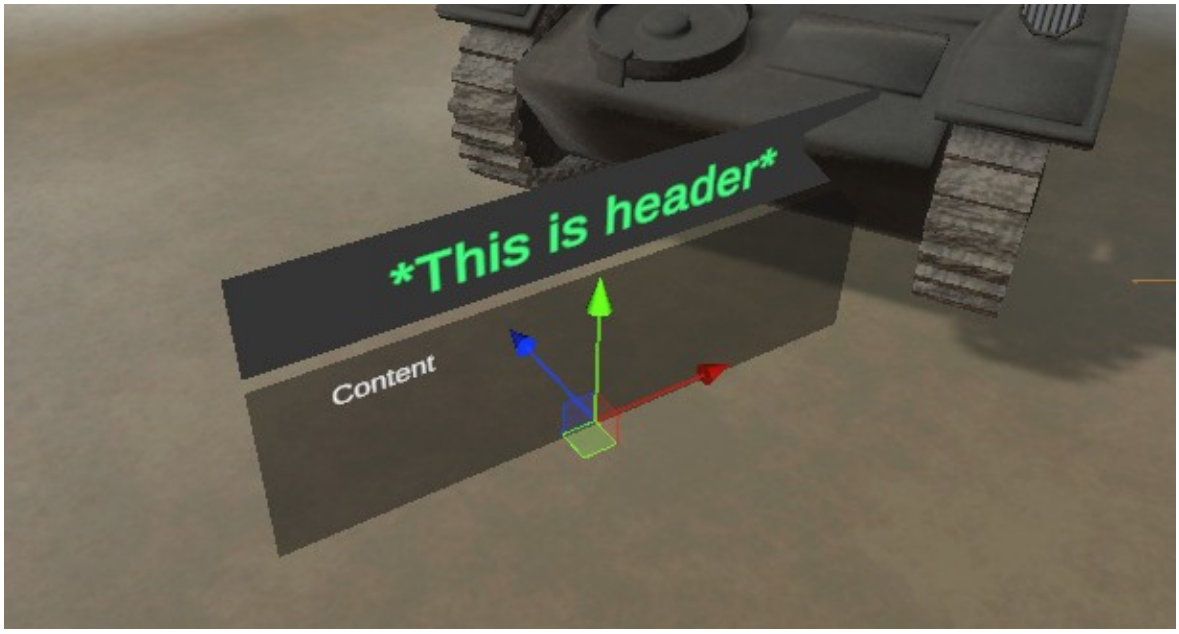
Jaké je hlavní město ČR? Praha, Brno, Ostrava

15 : 5 : 3, 7, 9

Tato práce si zaslouží za 1 : ,

7.2.5 Nastavení popisku objektu

Mezi přednastavenými objekty se nachází struktura InfoBoard, jenž slouží pro vložení popisků předmětů, které jsou u muzeí typické. Oproti klasickým muzejním popiskům všaklevitují ve vzduchu a otáčí se relativně k aktuální pozici kamery, díky tomu nemůže nastat situace, během které by uživatel nemohl přečíst text kvůli jeho natočení. Popisek také následuje objekt pro který je určen, pokud je s ním tedy pohnuto, nemůže dojít k záměně textů. Aby nebyla virtuální scéna příliš zahlcena velkým množství textu jsou tyto popisky skryty pod znakem otazníku, dokud se k nim nepřiblíží VR Kamera. Otazníku je možné nastavit jinou barvu dle preferencí, v základu se využívá zelená barva. Text může při uchopení objektu překážet, proto se všechny popisky vždy dočasně skryjí a objeví se opět ve chvíli kdy je objekt dostatečně daleko od kamery.



Obrázek 27 - Popisek před spuštěním aplikace

Použití těchto popisků je snadné, stačí přetáhnout objekt InfoBoard kamkoliv do scény, změna pozice je zbytečná jelikož se nastavuje sama ve skriptu dle referencovaného objektu, ten je také důležité nastavit v inspektoru pod políčkem Main Object, jinak skript vrátí chybu. Po tomto nastavení již popisek funguje tak, jak má. Nakonec se nastavují textové části popisku, Header Text určuje obsah nadpisu, Content Text samotný popis

objektu. Poslední, spíše volitelnou možností je posuvník pro nastavení výšky popisku, pokud je například objekt vyšší a popisek kvůli tomu není vidět. Tento posuvník ovlivňuje výšku neustále, může být tedy použit i při běhu aplikace pro jednodušší nastavení.

Obrázek 28 - Popisek po spuštění aplikace

Změna otazníku na text je podmíněna vzdáleností od uživatele, tato vzdálenost se dá upravit posuvníkem Hide Text Distance v inspektoru.

7.2.6 Export 3D modelu z Blenderu do Unity

V dřívějších verzí enginu Unity bylo možné přímo vložit soubor typu .blend, takže nebylo nutné model nijak exportovat. V době, kdy se tato práce vytvářela, je však tato funkce pět let zastaralá a nativní importování modelů se již nepoužívá. Pro moderní verze Unity je standardem formát .fbx. Export však není kompletně bezproblémový jelikož Blender a Unity obsahují rozdíly mezi osami -x, -y a -z. Pokud je tedy model vložen bez správného nastavení exportu, budou jeho osy pootočené, tento problém lze opravit přímo v Unity, ale je vhodnější ho řešit před exportem v Blenderu.

Pro opravení nesrovnalostí os má naštěstí Blender experimentální funkci, která dokáže tuto chybu odstranit sama, jsou zde ovšem limity. Před samotným exportem do .fbx je doporučeno vybrat objekt, který chce uživatel použít a teprve potom otevřít záložku pro výstup do fbx souboru. Díky předchozímu výběru se mohou pomocí políčka Limit to Selected Objects odfiltrovat ostatní nepotřebné objekty, které nejsou použitelné pro Unity jako osvětlení, podlaha, emitory částic a podobně. V kategorii Transform se vyskytuje ona experimentální funkce, Apply Transform, pokud je políčko zaškrtnuto, provedou se přepočty os a model se v Unity zobrazí se správnou rotací. Úskalím této funkce jsou možné problémy při práci s animacemi a kostrami postav, v takovém případě je nutné natočit samotnou kostru a manuálně aplikovat rotační transformaci.

Import do Unity

Exportovaný soubor stačí přetáhnout do správce souborů v Unity a odtamtud do scény, nastává však další problém a to nefunkční textury modelu. Unity nedokáže získat informace ze shaderů, které používá Blender (kromě jednoduché barvy z Principled BSDF shaderu), je proto nutné převést materiály z Blenderu do několika obrazových textur.

Texturové mapy

Pomocí těchto map je možné exportovat informace ze shaderu modelovacího softwaru a uložit je do obrázku klasického formátu – jpeg, png, apod. Dále zmíněné mapy nezahrnují veškeré existující ale pouze ty nejzákladnější, které se využívají nejčastěji.

Diffuse/Albedo

V Blenderu se využívá tzv. Diffuse mapa, jenž zjednodušeně obsahuje data o barvě a osvětlení. Oproti tomu Unity využívá modernější Albedo mapu, která obsahuje data čistě o barvě. V současné době se Albedo využívá ve větší míře, jelikož je mnohem častější praktikou poskytnout enginu barevnou složku textury a osvětlení nechat zapéct až při kompilaci scény.

Normal Map

Normálová mapa je velice mocný nástroj v boji s hustou topologií, umožňuje vytváření většího množství detailů, které reagují na světlo bez přidání dalších polygonů. Standardní praktikou v herním průmyslu je vytvoření modelu s nízkou hustotou topologie a modelu s

vysokou hustotou, model vysoké hustoty je následně exportován do normálové mapy, která se aplikuje na model s nízkou hustotou, detaily jsou zachovány ale nárok na výkon rapidně snížen.

Roughness/Smoothness

Blender pracuje s mapou Roughness, určující ostrost odrazu v předmětu, objekty s nízkou úrovní Roughness odráží ostřejší obraz s vysokou zakalenější. Unity ovšem používá Smoothness jenž je přímou inverzí Roughness. Aby byly Blender Rougness mapy správně zobrazené v Unity je nutné nejdříve převrátit jejich barvy.

Export mapy v Blenderu

Blender je velice specifický v procesu exportu map a nedodržení jednoho z kroků celého postupu vytváří zbytečné komplikace. Daný proces se skládá ze dvou základních částí: vytvoření UV mapy a zapečení textur do vytvořené mapy.

UV mapa

Předtím než je možné exportovat konkrétní typy textur, musí se vytvořit UV mapa, ta určuje, jak bude následně vyexportovaný obrázek na model „přilepen“. Nejlepším způsobem je vytvořit si mapu vlastnoručně, přidělováním tzv. „seams“ na kraje objektu, tento proces zabírá výrazné množství času a úsilí aby byl proveden správně. V Blenderu se UV mapa vytváří nejrychleji funkcí Smart UV Project, stačí mít celý model vybrán v editačním režimu a zakliknout Smart UV Project v záložce UV. Výsledná mapa není perfektní, ale je funkční.

Baking

Exportu map se také říká „baking“ nebo-li zapékání, v Blenderu je nutné mít jako aktuální renderovací engine Cycles, Eeve tuto funkci v době psaní práce nepodporuje. V materiálu daného objektu se také musí nacházet node Image texture, nemusí být k ničemu připojen ale musí existovat. Kliknutím na New se zobrazí možnost vytvoření nového obrázku, na názvu nezáleží ale výška a šířka jsou podstatné, standardem je využití mocniny dvojky, typicky se používají hodnoty 1024x1024, 2048x2048, 4096x4096, apod. Větší rozlišení se rovná většímu množství detailů které se mohou pojmout. Důležité je si ale uvědomit, že s

velikostí textur se zvyšuje i využití paměti. Po vytvoření obrázku může začít samotné zapékání.

V kategorii Render properties se nachází záložka Bake, zde je zásadní nastavit jaký typ mapy má být vytvořen. Pro základní barevnou texturu se volí možnost Diffuse, kde se v záložce Influence zaškrtne pouze Color, políčka Direct a Indirect musí byt vypnuta, jinak budou v textuře světelná data. Pro normálovou a Rougness mapu je postup stejný, pouze se nenastavuje hodnota Influence, veškerá ostatní nastavení mohou být nechána ve svých počátečních hodnotách. Není doporučeno exportovat shadery s Metallic hodnotou vyšší jak nula, vhodnější je tuto hodnotu měnit přímo v Unity pro lepší grafické výsledky odrazu. Po každém zapečení se přepíše obsah obrázku v nodu Image Texture výstupní mapou. S každou operací je tedy nutné uložit exportovaná data jako nový soubor, tato možnost se nachází v pohledech Image Editor či UV Editor → Image → Save as.

Import textur

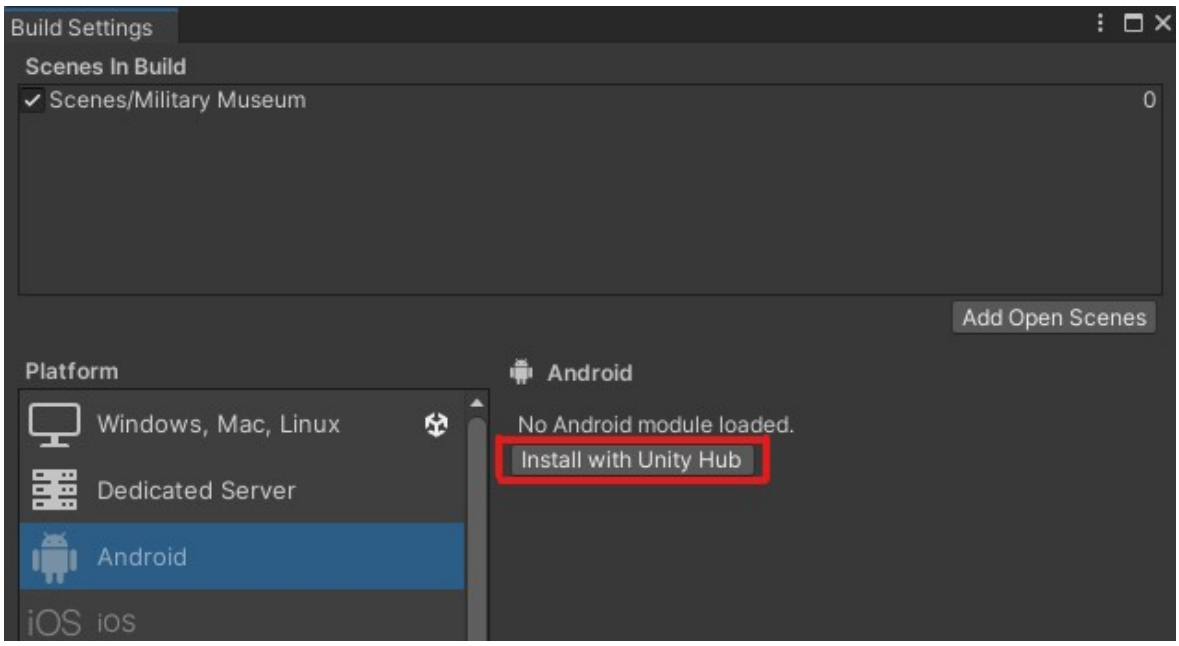
Pro užití exportovaných map musí být vytvořen materiál v herním enginu, lze ho vytvořit od znova anebo využít již existujícího materiálu v exportovaném modelu, ten sice obsahuje jen informaci o základní barvě, je však už připojen k modelu, jedná se tedy o jednoduchou úpravu. Aby byla úprava přístupná, musí být materiál extrahován z modelu kliknutím na něj ve správci souborů Unity a v záložce Materials na Extract Materials, zobrazené okno následně požádá o složku na uložení materiálu. V modelu se tato složka jmenuje Materials, nejedná se však o nutnost, důležité je pouze aby se materiál nacházel pod hlavní složkou Assets, která je přístupná v rozhraní Unity projektu. Po vytvoření materiálu přichází na řadu samotné mapy, po jejich vložení do Unity (v práci jsou tyto mapy vkládány do samotných podsložek v Materials pro jednodušší orientaci) je stačí přetáhnout do prázdných políček materiálu, barevná mapa v Blenderu odpovídá Albedo mapě v Unity, normálová mapa je v obou programech totožná. Pokud byla exportována Roughness mapa s inverzí, předává se do Metallic, pokud nebyla vytvořena, může být hodnota jednoduše nastavena přímo v Inspektoru. Pokud se model zdá průhledný anebo vykazuje jiné chování než v Blenderu, je špatně nastaven Rendering Mode v materiálu a musí být změněn na Opaque.

7.2.7 Export aplikace z prostředí Unity

Aby bylo možné exportovat samostatnou spustitelnou aplikaci je nutné mít vytvořenou alespoň jednu scénu a mít ji přidanou v záložce File → Build Settings, pokud není žádná přidána, stačí ji přetáhnout ze správce projektu a nebo kliknout na tlačítko Add Open Scenes za předpokladu, že je žádaná scéna v aktuální chvíli otevřena.

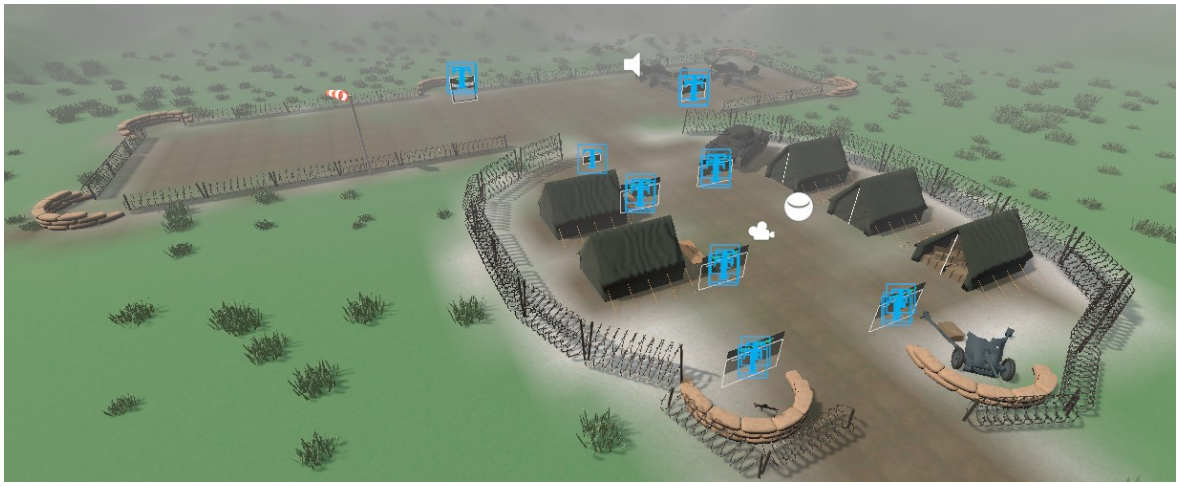
V základu se s Unity instalují pouze nástroje na budování aplikací pro platformu, na které je vyvíjeno, pokud je aplikace dělaná například pro samostatné headsety Oculus, musí být nejprve nainstalovány vývojářské nástroje pro Android. Unity hub se o instalaci těchto nástrojů postará sám.

Pomocí tlačítka Build & Run se export potvrzuje, po zadání cesty k libovolné složce začne proces, vytvářející spustitelný soubor pro žádanou platformu společně s dalšími pomocnými soubory.



Obrázek 29 - Ukázka instalace modulů pro export

8 Ukázkové řešení



Obrázek 30 - Ukázka vojenského muzea

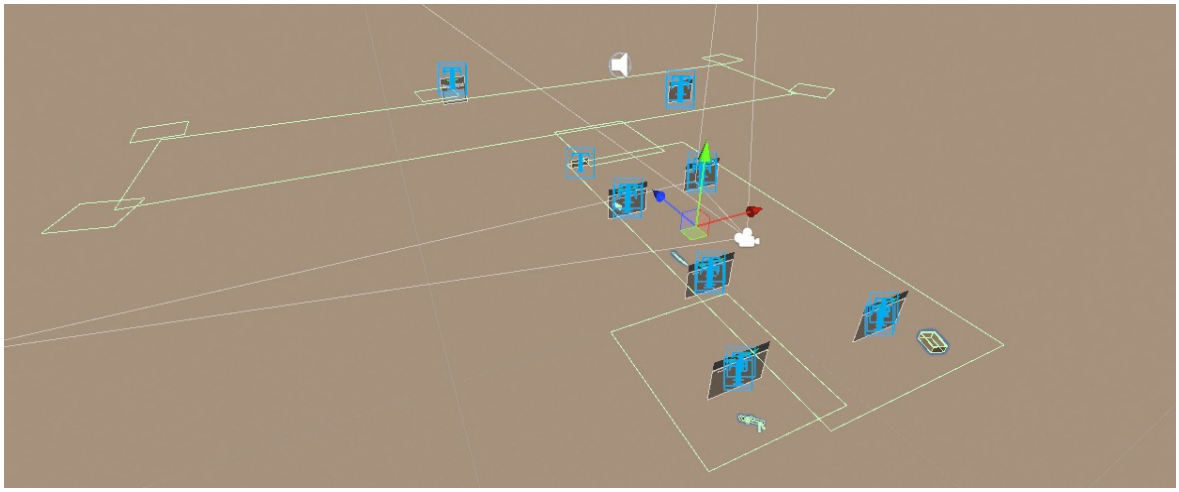
Pro ukázkové řešení této práce bylo zvoleno vojenské muzeum, důvodem je především zájem autora o tuto tematiku. Vymodelované prostředí neodpovídá skutečné lokaci, ale je inspirované předsunutými vojenskými tábory. Technika je v tomto řešení výhradně německého typu z doby druhé světové války, cílem tohoto zaměření je navození pocitu přítomnosti ve skutečném táboře za válečného stavu, této skutečnosti odpovídá i ambientní ozvučení.

8.1 Struktura ukázkové scény

Ukázková scéna je pro přehlednost rozdělena do tří sekcí – Scene Basics, XR Basics a Enviroment.

8.1.1 Interaktivní objekty

Zastřešující objekt XR Basics obsahuje veškerou interakční logiku tvořící virtuální realitu – teleportační oblasti, informační a kvízové tabule, ukázkové zvukové expozice, interaktivní modely zbraní a hlavně kamerovou soustavu.



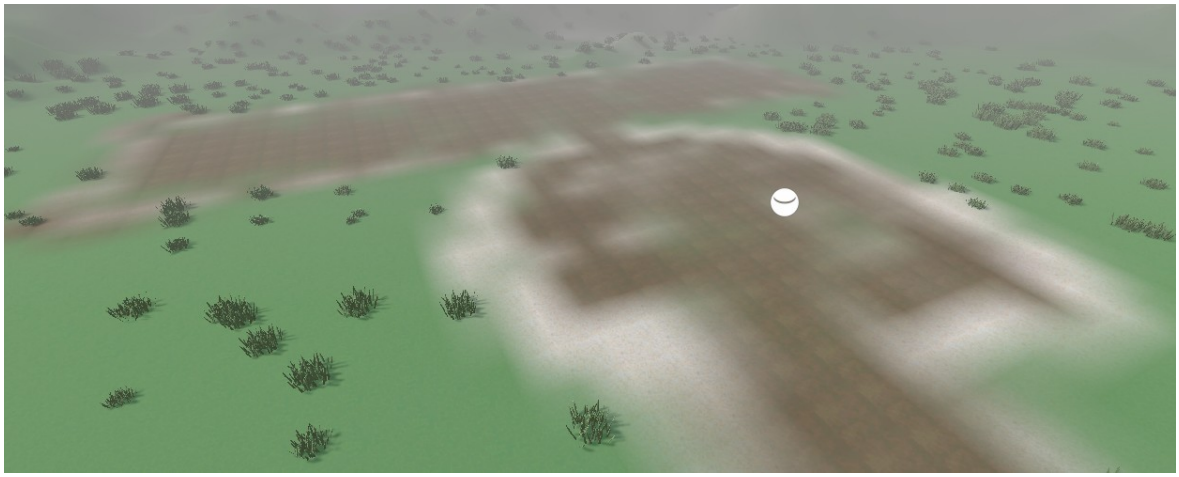
Obrázek 31 - Ukázkainteraktivních objektů vojenského muzea

Teleportační oblasti (zelené obdélníkové obrysy viz. obrázek výše) jsou totožné předpřipravené objekty, pouze je u nich vypnuto renderování (Mesh Renderer komponent) a je jim upravena velikost. Informační tabule jsou též přednastavené objekty se změněným textovým obsahem, výškou tabule a především referencí na herní objekt o kterém informují. U zvukových expozicí je pouze nastaven zvukový soubor, jenž se má po interakci přehrávat.

U objektů, které jsou určené k uchopení, se nachází, dle popsaného postupu v modelu, komponenty, umožňující interakci a užití fyziky.

8.1.2 Přídavné objekty

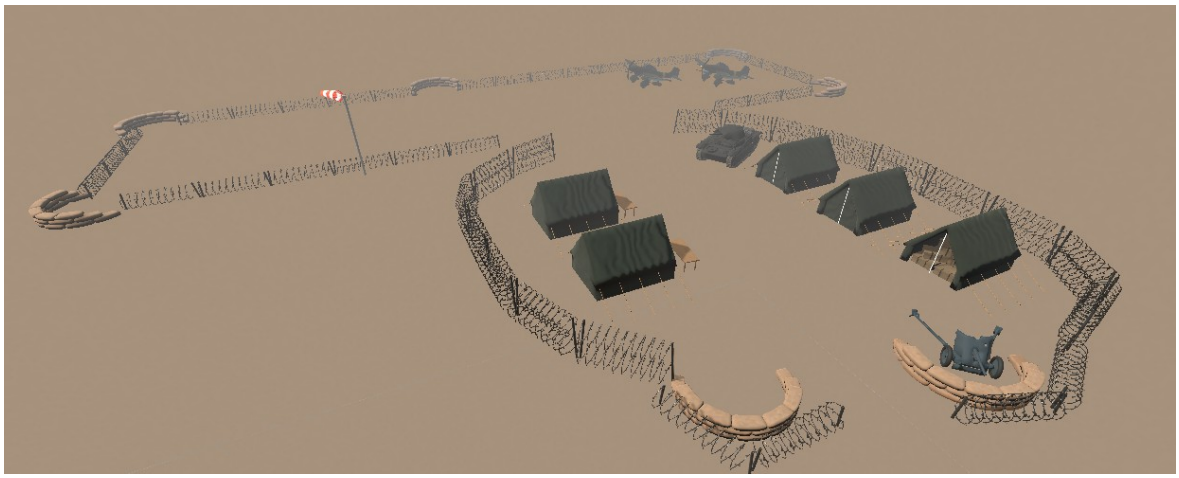
Tyto objekty nejsou nutností pro fungování scény, pouze napomáhají ke zvýšení imerze, je však doporučeno mít minimálně jedno světlo ve scéně, scéna by při absenci světla nebyla vidět.



Obrázek 32 - Ukázka nekolizních objektů tvořících scenérii vojenského muzea

Objekt Scene basics obsahuje nejzákladnější objekty tvořící scénu - terén oblasti, osvětlení, ambientní zvukový zdroj, odrazovou sondu a kolizní pole, které zajišťuje aby se hozené objekty nepropadali do země.

Terén obsahuje tři druhy textur tvořených v Blenderu, modelová část je vybudována pomocí editoru přímo v Unity. Zvuková nahrávka je zapůjčena od autora AMBIENCE CHANNEL [63].



Obrázek 33 - Ukázka kolizních objektů vojenského muzea

Finální část, zařazena pod objekt Enviroment, je složená s různých objektů tvořící prostředí – neuchopitelné modely zbraní, vozidel, ostnaté dráty, pytel s pískem, stany, apod.

8.2 Obsah

Veškeré modely i textury byly dělány čistě v programu Blender, velice nápomocnými se při modelování staly nákresy vojenské techniky z řady webových stránek, které obsahovali podoby konkrétních objektů ve třech osách. V materiálech jsou pro vytváření dodatečných detailů hojně využívány generátory šumu a vln v kombinaci s normálovou mapou.

8.2.1 Seznam exponátů

V této kapitole se nachází soupis exponátů, jenž byly pro ukázku procesního modelu vytvořeny. Pěchotní výzbroj tvoří interaktivní modely zbraní, s nastavenými místy uchopení, těžká technika je pouze statického typu, nelze ji uchopovat, či s ní jinak pohybovat.

Pěchotní výzbroj

Puška Kar98k

Obrázek 34 - Model pušky Kar98k

Samopal MP40

Obrázek 35 - Model samopalu MP40

Lehký kulomet MG42

Obrázek 36 - Kulomet MG42

Těžká technika

Pěchotní dělo Pak36

Obrázek 37 - Pěchotní dělo Pak36

Lehký tank Panzer 2

Obrázek 38 - Lehký tank Panzer 2

Střemhlavý bombardér Ju87

Obrázek 39 - Bombardér Ju87 Stuka

9 Závěr

Za cíl této práce bylo vytvoření procesního modelu, který by umožnil tvorbu aplikací virtuální reality vývojářům neobeznámeným s touto problematikou, potencionálně vedoucí k většímu rozšíření virtuální reality. Díky předpřipraveným objektům a ukázkovým materiálům je možné, při použití modelu, vytvořit funkční aplikaci za výrazně kratší čas a s menším úsilím. V modelu jsou zahrnuty základní logické objekty pro pohyb, zobrazování informací, pouštění zvukových nahrávek a také jednoduché testování znalostí pomocí kvízů. Zároveň jsou obsaženy elementární modely pro tvorbu prostředí společně s kompletní knihovnou objektů vytvořenou pro ukázkové řešení.

Tvorba obsahu je jednou z problematik, jenž zde také figuruje, získání kvalitních a konzistentních 3D modelů připravených pro herní enginy není jednoduché, občas až nemožné. Ani u placeného obsahu není práce bezstarostná, nutné je mít alespoň základní znalosti v oblasti exportu a práce s UV mapami. Základní techniky exportu jsou v práci popsány a měli by tak poskytnou dostatek informací pro jejich praktické využití s existujícími modely. Samotná tvorba modelů není v práci řešena, neboť to otevírá kapitolu pro úplně novou problematiku, na kterou se práce nezaměřuje.

V rámci práce na modelu se vyskytovala řada problémů, které neměli vždy přímočaré řešení, občas dokonce žádné. Určité typy exportovaných map z modelovacího softwaru nebyly vůbec použitelné v herním enginu a museli v něm být zreplikovány. Řada přídavných grafických vylepšení standardní pipeliny použitého enginu nebyla v době vývoje podporována v režimu virtuální reality, podobně jako řada logických prvků. Kamera virtuální reality kompletně ignorovala jakákoliv kolizní pole, přinášející problémy s volným procházením kamery skrz objekty. Celkově, i přes vzniklé problémy, dosáhla práce svých vytyčených cílů.

10 Seznam použitých informačních zdrojů

[1] *The Project Gutenberg eBook of Pygmalion's Spectacles, by Stanley G. Weinbaum* [online]. [vid. 2022-03-09]. Dostupné z: https://www.gutenberg.org/files/22893/22893-h/22893-h.htm

[2] MAZURYK, Tomasz a Michael GERVAUTZ. *Virtual Reality - History, Applications, Technology and Future* [online]. B.m.: Institute of Computer Graphics, Vienna University of Technology, Austria. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.42.7849&rep=rep1&type=pdf

[3] *Základní pojmy OOP. Třída, objekt, jeho vlastnosti. Metody, proměnné. Konstruktory.* [online]. [vid. 2022-03-09]. Dostupné z: https://is.muni.cz/el/1433/podzim2007/PB162/um/02/printable.html#minimodule33

[4] MILGRAM, Paul a Fumio KISHINO. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems* [online]. nedatováno, **1994**(12), 77. Dostupné z: https://www.researchgate.net/profile/Paul-Milgram/publication/231514051_A_Taxonomy_of_Mixed_Reality_Visual_Displays/links/02e7e52ade5e1713ea000000/A-Taxonomy-of-Mixed-Reality-Visual-Displays.pdf

[5] GOODE, Lauren. Get Ready to Hear a Lot More About „XR". *Wired* [online]. nedatováno [vid. 2022-04-09]. ISSN 1059-1028. Dostupné z: https://www.wired.com/story/what-is-xr/

[6] *Milgrams Virtuality Continuum* [online]. [vid. 2022-03-17]. Dostupné z: https://www.researchgate.net/profile/Zeynep-Burcu-Kaya-Alpan/publication/344556882/figure/fig2/AS:944658297397249@1602235497153/Milgrams-Virtuality-Continuum.jpg

[7] MICROSOFT HOLOLENS. *Introducing Microsoft HoloLens 2* [online]. 2019 [vid. 2022-03-12]. Dostupné z: https://www.youtube.com/watch?v=eqFqtAJMtYE

[8] Tech Specs. *Glass* [online]. [vid. 2022-03-12]. Dostupné z: https://www.google.com/glass/tech-specs/

[9] VIRTUAL REALITY OASIS. *Track A Keyboard In VR Using The Oculus Quest 2* [online]. 2021 [vid. 2022-03-13]. Dostupné z: https://www.youtube.com/watch?v=QrQhxol7rPw

[10] YOH, Myeung-Sook. The reality of virtual reality. In: *Proceedings Seventh International Conference on Virtual Systems and Multimedia*: *Proceedings Seventh International Conference on Virtual Systems and Multimedia* [online]. 2001, s. 666–674. Dostupné z: doi:10.1109/VSMM.2001.969726

[11] *telepresence noun - Definition, pictures, pronunciation and usage notes | Oxford Advanced Learner's Dictionary at OxfordLearnersDictionaries.com* [online]. [vid. 2022-03-14]. Dostupné z: https://www.oxfordlearnersdictionaries.com/definition/english/telepresence

[12] HEIM, Michael. *Virtual Realism*. B.m.: Oxford University Press, 2000. ISBN 978-0-19-535009-8.

[13] BIOCCA, Frank a Mark R. LEVY. *Communication in the Age of Virtual Reality*. B.m.: Routledge, 1995. ISBN 978-0-8058-1549-8.

[14] Acer's Predator Z35 raises the bar, but shoots itself in the foot. *Digital Trends* [online]. 5. červenec 2016 [vid. 2022-04-08]. Dostupné z: https://www.digitaltrends.com/monitor-reviews/acer-predator-z35-review/

[15] The weather forecast uses a Green Screen and So Can You. *Teleprompter for Video* [online]. 9. listopad 2020 [vid. 2022-04-08]. Dostupné z: https://scripted.video/green-screen-virtual-background-video/

[16] VI-grade launches Unreal environment for simulators. *Vehicle Dynamics International* [online]. 11. listopad 2020 [vid. 2022-04-08]. Dostupné z: https://www.vehicledynamicsinternational.com/news/simulation/vi-grade-launches-unreal-environment-for-simulators.html

[17] Disney says virtual reality "cave" is the future of immersion technology | attractionsmanagement.com news. *http://www.attractionsmanagement.com/* [online]. [vid. 2022-04-08]. Dostupné z: http://www.attractionsmanagement.com/news.cfm?codeID=314449

[18] JANUARY 2020, James Pearce8. VR market saw 5% growth in 2019 as consumers shift to next gen devices. *IBC* [online]. [vid. 2022-04-08]. Dostupné z: https://www.ibc.org/trends/vr-market-saw-5-growth-in-2019-as-consumers-shift-to-next-gen-devices/5326.article

[19] READ, Jenny C. A. What is stereoscopic vision good for? In: Nicolas S. HOLLIMAN, Andrew J. WOODS, Gregg E. FAVALORA a Takashi KAWAI, ed. *SPIE/IS&T Electronic Imaging* [online]. 2015, s. 93910N [vid. 2022-03-09]. Dostupné z: doi:10.1117/12.2184988

[20] BORANGIU, Theodor a Alexandru DUMITRACHE. Robot Arms with 3D Vision Capabilities. In: [online]. 2010. ISBN 978-953-307-070-4. Dostupné z: doi:10.5772/9668

[21] PROKÝŠEK, Miloš. Didaktické aspekty využití prostorového zobrazování [online]. 2012 [vid. 2022-03-09]. Dostupné z: https://dspace.cuni.cz/handle/20.500.11956/45552

[22] MELIM, Andrew. *Increasing Fidelity with Constellation-Tracked Controllers* [online]. [vid. 2022-04-09]. Dostupné z: https://developer.oculus.com/blog/increasing-fidelity-with-constellation-tracked-controllers

[23] ADAM SAVAGE'S TESTED. *SteamVR's „Lighthouse" for Virtual Reality and Beyond* [online]. 2015 [vid. 2022-04-09]. Dostupné z: https://www.youtube.com/watch?v=xrsUMEbLtOs

[24] UPLOADVR. *Oculus Insight: How Facebook's Oculus Quest & Rift S Track Your Head And Hand Movements* [online]. 2019 [vid. 2022-04-09]. Dostupné z: https://www.youtube.com/watch?v=nrj3JE-NHMw

[25] BELL, Mark W. Toward a Definition of "Virtual Worlds". *Journal For Virtual Worlds Research* [online]. 1970, **1**(1) [vid. 2022-04-15]. ISSN 1941-8477. Dostupné z: doi:10.4101/jvwr.v1i1.283

[26] SØRAKER, Johnny Hartz. Virtual Entities, Environments, Worlds and Reality. nedatováno, 28.

[27] SCHROEDER, Ralph. Defining Virtual Worlds and Virtual Environments. *Journal For Virtual Worlds Research* [online]. 1970, **1**(1) [vid. 2022-04-15]. ISSN 1941-8477. Dostupné z: doi:10.4101/jvwr.v1i1.294

[28] HEETER, Carrie. Interactivity in the Context of Designed Experiences. *Journal of Interactive Advertising* [online]. 2000, **1**(1), 3–14. ISSN 1525-2019. Dostupné z: doi:10.1080/15252019.2000.10722040

[29] NAIMARK, Michael. VR Interactivity. *Medium* [online]. 22. říjen 2016 [vid. 2022-03-09]. Dostupné z: https://michaelnaimark.medium.com/vr-interactivity-59cd87ef9b6c

[30] HU AU, Elliot a Joey J. LEE. Virtual reality in education: a tool for learning in the experience age. *International Journal of Innovation in Education* [online]. 2017, **4**(4), 215. ISSN 1755-151X, 1755-1528. Dostupné z: doi:10.1504/IJIIE.2017.10012691

[31] *Virtual Boy Set* [online]. [vid. 2022-03-17]. Dostupné z: https://imagenesgamers.canalrcn.com/ImgTodoGamers/vb_0.jpg

[32] EDWARDS, Benj. Unraveling The Enigma Of Nintendo's Virtual Boy, 20 Years Later. *Fast Company* [online]. 21. srpen 2015 [vid. 2022-03-17]. Dostupné z: https://www.fastcompany.com/3050016/unraveling-the-enigma-of-nintendos-virtual-boy-20-years-later

[33] VG LEGACY. *Sega VR - Press Conference Reveal* [online]. 2020 [vid. 2022-03-17]. Dostupné z: https://www.youtube.com/watch?v=RFJyEz8B2uo

[34] Sega VR - VG Legacy | Genesis / Mega Drive Virtaul Reality Add-on. *VG Legacy* [online]. [vid. 2022-03-09]. Dostupné z: https://vglegacy.com/hardware/sega-vr/

[35] *Sega VR* [online]. [vid. 2022-03-17]. Dostupné z: https://edge.alluremedia.com.au/m/k/2017/11/2017-11-02_094128.jpg

[36] A Brief History Of Oculus. *TechCrunch* [online]. [vid. 2022-03-17]. Dostupné z: https://social.techcrunch.com/2014/03/26/a-brief-history-of-oculus/

[37] *Oculus Development Kit 1* [online]. [vid. 2022-03-17]. Dostupné z: https://static.wixstatic.com/media/0a0847_d30ce368b88c40dd8b3066021b2a7700~mv2_d_1600_1200_s_2.jpg/v1/fill/

w_996,h_746,al_c,q_85,usm_0.66_1.00_0.01/0a0847_d30ce368b88c40dd8b30660
21b2a7700~mv2_d_1600_1200_s_2.jpg

[38] Oculus Rift DK1: Full Specification. *VRcompare* [online]. [vid. 2022-03-09]. Dostupné z: https://vr-compare.com/headset/oculusriftdk1

[39] Oculus Rift DK2: Full Specification. *VRcompare* [online]. [vid. 2022-03-09]. Dostupné z: https://vr-compare.com/headset/oculusriftdk2

[40] *Oculus Development Kit 2* [online]. [vid. 2022-03-17]. Dostupné z: https://www.vrbites.com/wp-content/uploads/2014/10/nexusae0_oculus-rift-dev-kit-2.jpg

[41] *Oculus Rift: Full Specification - VRcompare* [online]. [vid. 2022-03-17]. Dostupné z: https://www.vr-compare.com/headset/oculusrift

[42] *Oculus Rift* [online]. [vid. 2022-03-17]. Dostupné z: https://vr-geeks.com/wp-content/uploads/2019/08/Oculus-Rift-Review.jpg

[43] Oculus Go: Full Specification. *VRcompare* [online]. [vid. 2022-03-09]. Dostupné z: https://vr-compare.com/headset/oculusgo

[44] *Oculus Go* [online]. [vid. 2022-03-17]. Dostupné z: https://roadtovrlive-5ea0.kxcdn.com/wp-content/uploads/2018/05/oculus-go-lrg.jpg

[45] *Oculus Quest* [online]. [vid. 2022-03-17]. Dostupné z: https://assets.website-files.com/5c8048f9f674b521a0e84c30/5df2adcdc2725fc6b08e781d_oculus%20(3).png

[46] *Oculus Quest: Full Specification - VRcompare* [online]. [vid. 2022-03-17]. Dostupné z: https://www.vr-compare.com/headset/oculusquest

[47] Oculus Quest 2: Full Specification. *VRcompare* [online]. [vid. 2022-03-09]. Dostupné z: https://vr-compare.com/headset/oculusquest2

[48] *Oculus Quest 2* [online]. [vid. 2022-04-10]. Dostupné z: https://www.mall.cz/i/59661103

[49] D'ORAZIO, Dante. Valve's VR headset is called the Vive and it's made by HTC. *The Verge* [online]. 1. březen 2015 [vid. 2022-03-09]. Dostupné z: https://www.theverge.com/2015/3/1/8127445/htc-vive-valve-vr-headset

[50] *HTC Vive: Full Specification - VRcompare* [online]. [vid. 2022-03-17]. Dostupné z: https://www.vr-compare.com/headset/htcvive

[51] *HTC Vive* [online]. [vid. 2022-03-17]. Dostupné z: http://image.en.yibada.com/data/images/full/106462/htc-vive.jpg

[52] *Valve Index: Full Specification - VRcompare* [online]. [vid. 2022-03-17]. Dostupné z: https://www.vr-compare.com/headset/valveindex

[53] *Valve Index* [online]. [vid. 2022-03-17]. Dostupné z: https://vr-room.ch/wp-content/uploads/2019/08/Index9.jpg

[54] PORTER, Jon. Valve reportedly developing standalone VR headset codenamed 'Deckard'. *The Verge* [online]. 29. září 2021 [vid. 2022-03-09]. Dostupné z: https://www.theverge.com/2021/9/29/22699914/valve-deckard-standalone-vr-headset-prototype-development

[55] About. *blender.org* [online]. [vid. 2022-03-09]. Dostupné z: https://www.blender.org/about/

[56] 3D Modeling, Texturing, Lighting, Animation and Simulation Software |…. *Maxon* [online]. [vid. 2022-03-09]. Dostupné z: https://www.maxon.net/en/cinema-4d

[57] *Plugin Development : Cinema 4D C++ SDK* [online]. [vid. 2022-03-09]. Dostupné z: https://developers.maxon.net/docs/Cinema4DCPPSDK/html/page_maxonapi_plugin_dev.html

[58] *3ds Max Software | Get Prices & Buy Official 3ds Max 2022* [online]. [vid. 2022-03-17]. Dostupné z: https://www.autodesk.com/products/3ds-max/overview

[59] Sketchfab - The best 3D viewer on the web. *Sketchfab* [online]. [vid. 2022-03-09]. Dostupné z: https://sketchfab.com

[60] Unity vs. Unreal: Which One is Best For You? *Udemy Blog* [online]. 29. září 2021 [vid. 2022-03-09]. Dostupné z: https://blog.udemy.com/unity-vs-unreal-which-game-engine-is-best-for-you/

[61] *Introduction — Godot Engine (3.1) documentation in English* [online]. [vid. 2022-03-09]. Dostupné z: https://docs.godotengine.org/en/3.1/about/introduction.html#doc-about-intro

[62] TECHNOLOGIES, Unity. *Free 2D, 3D, VR, & AR software for cross-platform development of games and mobile apps. - Unity Store* [online]. [vid. 2022-04-09]. Dostupné z: https://store.unity.com/products/unity-personal

[63] AMBIENCE CHANNEL. *RELAXING RAIN WITH DISTANT ARTILLERY w/BATTLE AMBIENCE | AMBIENCE CHANNEL* [online]. 2020 [vid. 2022-03-29]. Dostupné z: https://www.youtube.com/watch?v=wKduUdwXLt0

11 Seznam přiložených zdrojových kódů

12 Seznam obrázků