ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA STROJNÍ

Studijní program: B0715A270013 – Strojní inženýrství
Studijní specializace: Průmyslové inženýrství a management

BAKALÁŘSKÁ PRÁCE

Možnosti virtuální kolaborace

Autor: Kryštof Kortus

Vedoucí práce: doc. Ing. Petr Hořejší, Ph.D.

Konzultant: Ing. Konstantin Novikov, MBA

Akademický rok 2021/2022

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta strojní
Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Kryštof KORTUS**
Osobní číslo:	**S19B0587P**
Studijní program:	**B0715A270013 Strojní inženýrství**
Specializace:	**Průmyslové inženýrství a management**
Téma práce:	**Možnosti virtuální kolaborace**
Zadávající katedra:	**Katedra průmyslového inženýrství a managementu**

Zásady pro vypracování

1. Úvod
2. Současná řešení virtuální kolaborace
3. Možnosti síťové komunikace ve vývojovém prostředí Unity3D
4. Výběr vhodné varianty
5. Návrh prototypu ukázkové aplikace pro virtuální realitu
6. Závěr

Rozsah bakalářské práce: **30–40 stran**
Rozsah grafických prací: **0**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

1. OKITA, A. *Learning C# Programming with Unity 3D*. Second edition, Boca Raton, FL. USA: Routledge, 2019. 690 p. ISBN-13: 978-1138336810.
2. SUNG, K., SMITH, G. *Basic Math for Game Development with Unity 3D: A Beginner's Guide to Mathematical Foundations*. Bothel, WA. USA: Apress, 2019. 424 p. ISBN 978-1484254424.
3. LINOWES, J. *Unity Virtual Reality Projects: Learn Virtual Reality by developing more than 10 engaging projects with Unity 2018*. 2nd Edition, Birmingham, UK: Packt Publishing, 2018. 492 p. ISBN 978-1788478809.
4. LaVALLE, S. M. *Virtual Reality*. Cambridge University Press, 2020. 390 p., dostupné zdarma online na http://lavalle.pl/vr/
5. *Oficiální Unity3D návody dostupné na https://learn.unity.com/*

Vedoucí bakalářské práce: **doc. Ing. Petr Hořejší, Ph.D.**
Katedra průmyslového inženýrství a managementu
Konzultant bakalářské práce: **Ing. Konstantin Novikov**
Katedra průmyslového inženýrství a managementu

Datum zadání bakalářské práce: **20. září 2021**
Termín odevzdání bakalářské práce: **27. května 2022**

L.S.

Doc. Ing. Milan Edl, Ph.D.
děkan

Doc. Ing. Michal Šimon, Ph.D.
vedoucí katedry

V Plzni dne 20. září 2021

Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.
Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

V Plzni dne: ……. …….
podpis autora

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce, panu doc. Ing. Petrovi Hořejšímu, Ph.D. a svému konzultantovi Ing. Konstantinovi Novikovu za jejich ochotu a podporu při vypracování mé bakalářské práce. Hlavně bych chtěl poděkovat za jejich konzultace, které byli vždy okamžité a postačují k dokončení mé bakalářské práce.

ANOTAČNÍ LIST BAKALÁŘSKÉ PRÁCE

AUTOR	**Příjmení** Kortus	**Jméno** Kryštof
STUDIJNÍ PROGRAM	B0715A270013 Strojní inženýrství	
VEDOUCÍ PRÁCE	**Příjmení (včetně titulů)** Doc. Ing. Hořejší, Ph.D.	**Jméno** Petr
PRACOVIŠTĚ	ZČU – FST – KPV	

DRUH PRÁCE	~~**DIPLOMOVÁ**~~	**BAKALÁŘSKÁ**	**Nehodící se škrtněte**
NÁZEV PRÁCE	Možnosti virtuální kolaborace		

FAKULTA	strojní		**KATEDRA**	KPV		**ROK ODEVZD.**	2022

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	58		**TEXTOVÁ ČÁST**	58		**GRAFICKÁ ČÁST**	0

STRUČNÝ POPIS (MAX 10 ŘÁDEK) **ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY**	Bakalářská práce se zabývá možnostmi využití virtuální reality pro virtuální kolaboraci. V této oblasti lze očekávat vysoký potenciál využitelnosti. Cílem práce je vytvoření aplikace, která s využitím brýlí pro virtuální realitu, a vybraného serveru, zprostředkuje komunikaci ve virtuálním prostředí. V rámci bakalářské práce bude vybrán vhodný software a metoda pro síťovou komunikaci. Daná aplikace bude kompatibilní s platformou Android a má veliké využití v průmyslových podnicích.
KLÍČOVÁ SLOVA **ZPRAVIDLA JEDNOSLOVNÉ POJMY, KTERÉ VYSTIHUJÍ PODSTATU PRÁCE**	VR, AR, Unity 3D, Photon, RPC

SUMMARY OF BACHELOR SHEET

AUTHOR	**Surname** Kortus	**Name** Kryštof
STUDY PROGRAMME	B0715A270013 Mechanical Engineering	
SUPERVISOR	**Surname (Inclusive of Degrees)** Doc. Ing. Hořejší, Ph.D.	**Name** Petr
INSTITUTION	ZČU – FST – KPV	

TYPE OF WORK	~~**DIPLOMA**~~	**BACHELOR**	**Delete when not applicable**
TITLE OF THE WORK	Virtual Collaboration Possibilities		

FACULTY	Mechanical Engineering	**DEPARTMENT**	KPV	**SUBMITTED IN**	2022

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	58	**TEXT PART**	58	**GRAPHICAL PART**	0

BRIEF DESCRIPTION **TOPIC, GOAL, RESULTS AND CONTRIBUTIONS**	The bachelor thesis deals with the possibilities of using virtual reality for virtual collaboration. A high potential of usability can be expected in this area. The aim of the thesis is to create an application that, using virtual reality goggles and a selected server, mediates communication in a virtual environment. Within the scope of the bachelor thesis, a suitable software and method for network communication will be selected. The application in question will be compatible with the Android platform and has great use in industrial enterprises.
KEY WORDS	VR, AR, Unity 3D, Photon, RPC

Obsah

Seznam obrázků

Seznam tabulek

Použité zkratky

VR Collaboration – kolaborace ve virtuální realitě

VR (Virtual reality) – Virtuální realita

AR (Augmented reality) – Rozšířená realita

2D (two-dimensional) – dvojrozměrný

3D (three-dimensional) – trojrozměrný

RPC (Remote Procedure Calls) - Vzdálená volání procedur

MMO (massively multiplayer online) - Masivní online aplikace

UDP (User Datagram Protocol) – nezabezpečený proti ztrátě paketů

TCP (Transmision Control Protocol) – zabezpečení proti ztrátě paketů

Úvod

Vzhledem technologickému rozvoji a prodělané pandemii se začala v nedávných letech významně rozvíjet oblast vzdálené komunikace. Po implementaci veškerých konvenčních způsobů vzdálené komunikace se pozornost soustřeďuje na kolaboraci s využitím virtuální reality (VR). V současné době však zatím existuje jen málo hotových řešení. Zavedení kolaboračních VR aplikací do firem šetří čas, peníze i práci navíc, kterou pořádání schůzek vyžaduje. Zaměstnanci mohou konat schůzky nezávazně na tom kde se právě nachází, komentovat a vpisovat své poznámky v reálném čase do projektů ve 3D podobě nebo také školit své zaměstnance po celém světě v jeden okamžik. Příkladem může být nový výrobní stroj, který je vymodelován a vložen do aplikace pro virtuální schůzky. Kolaborace ve VR nabízí velké možnosti, díky kterým může být vývoj firmy mnohem rychlejší než u ostatních, jenž nedokáží držet krok s dobou.

Lze konstatovat, že kolaborace ve VR spadá do konceptu Industry 4.0, kde ji lze použít například pro návrh produktů nebo procesů a ověřování prototypů. Inženýři mohou pomocí virtuální simulace a kolaborace kontrolovat a v reálném čase diskutovat, dosažený pokrok. Tímto způsobem lze omezit chyby (a to zejména ve fázi předprodukční) a zvýšit produktivitu.

Pro realizaci VR kolaborace jsou zapotřebí virtuální brýle s příslušným softwarem a připojení k internetu. Vzhledem k tomu, že zatím existuje jen velmi málo podobných simulátorů a zejména kvůli tomu, že na Katedře průmyslového inženýrství a managementu (KPV) chybí vlastní prototyp, který by bylo možné customizovat, bylo rozhodnuto, že se vyvine vlastní prototypová aplikace. Práce taková aplikace je primárním výstupem z této práce. Aplikace poskytne koncovým uživatelům možnost spolu komunikovat v rámci VR pomocí avatarů, kteří budou moci gestikulovat i mluvit. Účelem této aplikace je umožnit osobitější potkávání napříč různými místy. Využití přinese aplikace nejen v dobách pandemických. Aplikace je vytvořena v herním enginu Unity 3D a realizovaná přes vhodnou technologii síťové komunikace, která bude na základě požadavků vybrán.

Cíle bakalářské práce jsou:

1. Zmínění současného řešení virtuální kolaborace
2. Porovnaní technické možnosti síťové komunikace v Unity 3D
3. Výběr vhodné varianty na základě stanovených bodů
4. Návrh prototypu ukázkové aplikace pro virtuální realitu

1. Definice základních pojmů

Vzhledem k tomu, že existují různé definice níže uvedených pojmů, budou v této kapitole definovány některé pojmy, se kterými bude dále pracováno. V mnoha případech existuje více různých definic, je tedy nutné ukázat, jak se bude s těmito definicemi pracovat v rámci této práce:

***VR* Collaboration** – se rozumí sdílení dat v reálném čase s virtuálními 3D prvky a možnosti diskuze s osobou, která se nemusí nutně nacházet na stejné pozici jako prezentující.

VR – Virtual Reality (virtuální realita) je dle [1] vyvolání cílového chování organismu pomocí umělé smyslové stimulace, přičemž organismus o tomto zásahu neví nebo ví jen velmi málo.

AR – Augmented Reality (rozšířená realita) je považována za technologii s velkou budoucností. Podle [2] (Ronald Azuma – 1997) je AR spojení dvou různých světů, reálného a virtuálního prostředí. Jedná se o interakci s objekty s v reálném čase

2D - lze v herním průmyslu lze definovat jako prostředí s pouze dvěma osami pohybu. Obvykle jde o takzvané „ploché hry“, kde se můžete pohybovat pouze doleva, doprava, nahoru a dolu.

3D - v herním průmyslu lze definovat jako prostředí, ve kterém je možné pohybovat se ve všech směrech pohybu v trojrozměrných rovinách. To znamená že se hráč může ve virtuálním prostředím posouvat a otáčet o 360° ve všech třech osách (6 stupňů volnosti) a objekty mají hloubku, délku a výšku.

RPC – Remote Procedure Calling znamená "volání vzdálené procedury". Většina počítačových programů spouští procedury nebo sady instrukcí pomocí procesoru počítače. Jinými slovy, instrukce jsou zpracovávány lokálně na stejném počítači, ze kterého je program spuštěn. Vzdálené volání procedur však spouští procedury na jiných počítačích nebo zařízeních připojených k síti. Po provedení instrukcí se výsledky procedury obvykle vracejí do místního počítače. [3]

MMO – Massively Multiplayer Online je jakákoliv online aplikace, kde hráč komunikuje s velkým počtem dalších hráčů.

UDP – User Datagram Protocol je jednodušší internetový protokol bez připojení, u kterého nejsou vyžadovány služby kontroly chyb a obnovy. U protokolu UDP nevznikají žádné režijní náklady na otevření spojení, udržování spojení nebo ukončení spojení; data jsou příjemci odesílána nepřetržitě, ať už je přijme, nebo ne. [4]

TCP – Transmission Control Protocol je orientován na spojení, což znamená, že po navázání spojení lze data přenášet oběma směry. Protokol TCP má zabudované systémy pro kontrolu chyb a zaručuje, že data budou doručena v pořadí, v jakém byla odeslána, což z něj činí ideální protokol pro přenos informací, jako jsou statické obrázky, datové soubory a webové stránky. [4]

2. Současná řešení virtuální kolaborace

V rámci této kapitoly budou ukázány některé dílčí aspekty virtuální reality a současně se podíváme na některá vybraná již existující řešení virtuální kolaborace, a sice zejména z důvodů možné inspirace pro realizace výstupu bakalářské práce (vlastního prostředí pro kolaboraci).

Aktuální vývoj na trhu, i vzhledem ke covidové situaci ve světě, klade požadavky na vzdálenou kolaboraci. Pro společnosti je tak esenciální držet krok s dobou, pro udržení konkurence schopnosti a naplňování požadavků zákazníků.

Pokud si vezmeme tradiční pojem kolaborace využitý v technologii, tak bychom mohli virtuální kolaboraci vnímat třeba jako odesílání textových zpráv na sociálních sítích. Pokud se však budeme virtuální kolaborací zabývat v rámci Industry 4.0, může nabýt tento pojem nových rozměrů. (viz. Obr. 2-1)

Virtuální kolaborace z hlediska Industry 4.0 tak může znamenat, že máme k dispozici digitální obraz prostředí určité firmy (prostory, produkt, stroje, zaměstnance), do kterého lze virtuálně vstoupit. Pomocí síťových řešení (viz dále) pak může být v této virtuální místnosti více lidí najednou. Všichni uživatelé tak mohou mezi sebou komunikovat a interagovat s objekty (např. virtuálním strojem), aniž by se fyzicky nacházeli ve stejném místě.

Obr. 2-1 Ukázka virtuální kolaborace [5]

2.1. Virtual reality (Virtuální realita)

Nežli se podíváme na dílčí příklady existujících řešení, pozastavme se nad samotnou virtuální realitou. Hlavním úkolem virtuálním reality je co nejvíce napodobit reálné prostředí a zprostředkovat tak uživateli imerzi (ponoření do virtuálního světa). (viz. Obr. 2-2) V tomto 3D prostředí se můžeme pohybovat, manipulovat s vymodelovanými předměty a provádět další typy interakcí. Technologie VR jsou využívány zejména v herním průmyslu. Postupy a přístupy, které dříve byly vlastní zmiňovanému hernímu průmyslu jsou však úspěšně adaptovány obecně pro průmysl (v rámci konceptu Industry 4.0), zdravotnictví (např. kurzy pro zdravotníky, rehabilitace), v armádě a v dalších odvětvích.

Obr. 2-2 Ukázka virtuálního prostředí [6]

Je více způsobů, jak zprostředkovat imerzní zážitek. Lze říci, že dnes jsou nejdostupnější hardwarovou komponentou pro realizaci VR zážitku brýle pro virtuální realitu, tzv. HMD – Head Mounted Disply. Hardware nám může poskytnout hned několik firem. Mezi hlavní výrobce patří HTC, Oculus, SONY, Pico a další menší výrobci.

Každá společnost má vlastní podpůrný software pro vývojáře, který se snaží co nejrychleji vyvíjet. Autor této bakalářské práce se nejčastěji setkává s produktem od firmy Oculus typu Quest 2, který již má v možnosti bezdrátové připojení přes Wi-Fi s počítačem (Oculus Link) a možnost ovládání prostředí pomocí gest (Hand Tracking).

Pro tvorbu prototypu aplikace pro vzdálenou kolaboraci jsou relevantní mobilní verze VR hardwaru. Na Obr. 2-3 je možné vidět pro porovnání produkt od značky Pico typu Neo 3, u kterých je absence hand trackingu a HMD Oculus Quest 2. Při bližším pohledu na produkty od firmy Oculus a Pico, si nemůžeme nevšimnout jisté podobnosti. Firma Oculus je se svými produkty o generaci napřed oproti firmě Pico.

Obr. 2-3 Pico Neo 2 vs Oculus Quest 2 [7]

2.2. Konvenční způsoby

Spolupráce se samozřejmě v rámci podnikového prostředí již delší dobu využívá. (viz. Obr 2-4) Mezi volně dostupné softwary by se daly zařadit např. Skype, MS Teams. Tyto softwary umožnují textovou a audiovizuální komunikaci, také ale dovolují sdílení obrazovky což umožňuje zbylým uživatelům sledovat co se na obrazovce počítače děje. Po výrobní podniky je zajímavá možnost sdílení 3D dat pomocí softwarů. Dalšími softwary, které jsou již ovšem zpoplatněny, jsou např. AutoDesk3D, Siemens Teamcenter a PLM, Dassault Systém 3DExperience.

Obr. 2-4Ukázka softwaru Teamcenter v praxi [8]

Výhodou těchto systémů je snadné spuštění, jednoduchá obsluha a nespočet výukových programů. Protože jsou tyto systémy velice rozšířené, je k dispozici kvalitní podpora jak ze softwarového hlediska, tak i z hlediska uživatelského. Nevýhodou je, že s 3D daty může pracovat většinou jen jeden člen virtuální kolaborace, zatím co ostatní jsou sledující. Samozřejmě zde také není požadovaná úroveň imerze, což může vést ke ztrátě soustředění participantů, nedostatečné kontrole v zapojení se do diskuze a s tím i související možnost „vzájemně si nerozumět“. U zpoplatněných softwarů se může jako nevýhoda zdát jejich cena.

2.3. Kolaborace ve virtuální realitě

Kolaborativní spolupráce v aplikacích je jednou z možností využití virtuální reality. (viz Obr. 2-5) Může například jít o vytváření prostorů firem, kde je zapotřebí náhled více odborníků. Tito odborníci se pomocí virtuální reality mohou připojit do aplikace, která má v sobě model firmy, a mohou mezi sebou komunikovat a vzájemně interagovat s předměty v aplikaci. Díky virtuální realitě mohou pracovat odkudkoliv s živými daty, pokud mají dostačující hardware a silné připojení k internetu. [9]

Obr. 2-5 Ukázka kolaborace ve VR [10]

Nejprve se podívejme na jeden z ranných systémů: Systém VirCa (viz. Obr. 2-6) byl jeden z počátečních systémů začátku virtuální kolaborace, který byl představen v roce 2012. Vznikl na základě překonávání vzdálenostních bariér mezi jednotlivci v týmu. Tento systém umožňuje uskutečňovat porady ve virtuálním prostředí, kde každý člen, který se nemusí nacházet ve stejné lokaci jako ostatní členové, má svého avatara. Účelem tohoto systému je především výměna zkušeností a interakce s modely, interakce mezi jednotlivými členy a sdílení vizualizace modelů. [11]

Výhodou tohoto systému je možnost jednotlivce interagovat nejen s avatary ale také s ostatními virtuálními předměty. Na druhou stranu je tento systém není tvořen jako aplikace android pro samotné brýle a je spouštěn přes počítač.

Obr. 2-6 Ukázka scény s použitím systému VirCa [12]

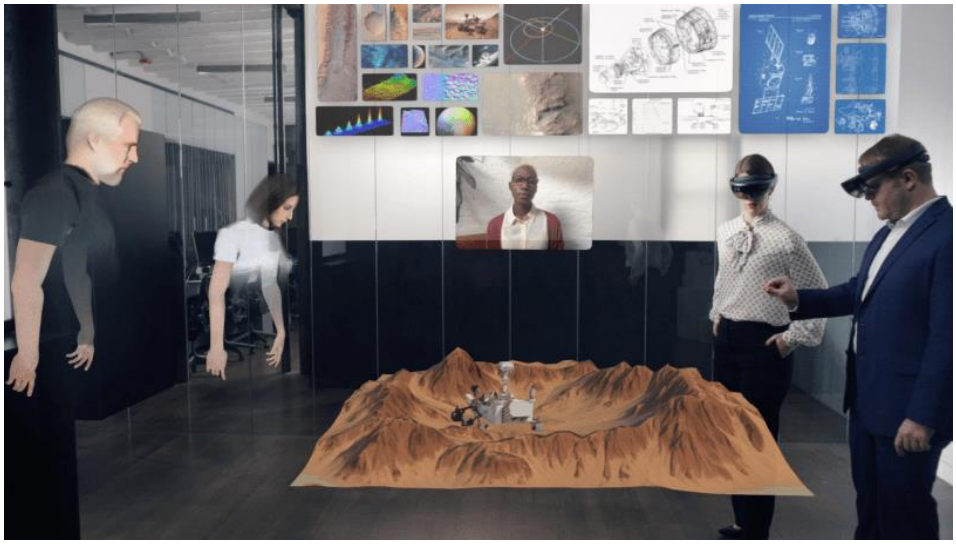
Z nynějších aplikací pro virtuální kolaboraci zmíníme aplikaci VRChat. Tato aplikace není však primárně zaměřená na konkrétní segment a zabývají se především jen virtuální kolaborací.

2.4. Virtuální kolaborace pomocí rozšířené reality

Oproti virtuální realitě se AR liší tím, že osobu nevtahuje do kompletně zpracovaného virtuálního prostředí, ale využívá i prostředí reálného světa. (viz. Obr. 2-7) Jedná se o překrývání nebo vkládání virtuálních objektů do reálného prostředí kolem nás. [13]

Z hlediska kolaborace existuje několik aplikací, které využívají AR. Dále bude uvedeno několik příkladů aplikací a jejich řešení. Roivis je aplikace rozšířené reality, která podporuje navrhování a plánování továren mezi uživateli, kteří se nacházejí na jednom místě a sdílejí společný pohled. DARCC (Hammad et al. 2009) je distribuovaná aplikace AR, která umožňuje více uživatelům simulovat společné stavební úlohy, např. koordinaci jeřábových operací. Gauglitz et al. (2014) představují systém založený na tabletu, který obsahuje rozhraní dotykové obrazovky, jehož prostřednictvím může vzdálený uživatel procházet fyzickým prostředím a vytvářet světově orientované anotace pro podporu údržby. Oda et al. (2013) představují systém pro vzdálenou údržbu zařízení. Tyto systémy ukazují, že technologie kolaborativní rozšířené reality by mohla mít významný dopad v oblasti designu a průmyslových aplikací. [2]

Z novějších aplikací můžeme zmínit software Spatial. Jde o aplikaci, která využívá AR v plné výši. Dovoluje uživateli vytvořit si vlastního avatara a proměnit svoji místnost na pracovní prostor, kde lze založit schůzku pro další uživatele této aplikace. Následně je umožněno prezentovat a všichni uživatelé mohou interagovat s prvky rozšířené reality. [14]



Obr. 2-7 Ukázka AR od společnost Spatial [15]

3. Technické možnosti síťové komunikace ve vývojovém prostředí Unity 3D

Vzhledem ke vstupním požadavkům a zkušenostem na Katedře průmyslového inženýrství a managementu bude výsledný prototyp softwarového řešení, které je výstupem z této práce, cílit na engine a současně vývojové prostředí Unity 3D. V rámci této kapitoly tedy bude v krátkosti představeno prostřední (a engine) Unity 3D a současně budou představeny některé možnosti pro síťovou komunikaci, které Unity3D přímo podporuje. Unity 3D má kvalitní podporu a zároveň kladné dispozice pro tvorbu virtuální kolaborace. Nutno podotknout, že některé aplikace zmíněné v předešlé kapitole byly vytvořeny právě v Unity 3D.

3.1. Herní engine Unity 3D

Unity 3D je multiplatformní herní engine, který byl vytvořen společností Unity 3D Technologies. V roce 2005 byla vydána první verze 1.0. Nyní je již vydaná verze 2021.x (květen 2022), která je volně k dispozici ve verzi Personal na oficiálních stránkách Unity 3D, dále existuje verze Profesional, která je placená. Personal edice je dostačující pro tvorbu menších projektů a jejich prodeji. Unity 3D slouží pro tvorbu 2D a 3D aplikací pro počítače, mobilní zařízení a konzole. Aplikace lze vytvořit v širokém seznamu softwarových platforem (viz. Tab. 3-1). [16]

Desktopové a webové aplikace	**Mobilní telefony**	**Rozšířená Realita (XR)**	**Konzole**
Windows (PC)	iOS	Arit	PS 5
Mac	Android	ARCore	PS 4
Universal Windows platform (UWP)	-	Windows Mixed Reality	Nintendo Switch
Linux StandAlone	-	Microsoft HoloLens	Google Stadia
WebGL	-	Magic Leap (Lumin)	Xbox One
-	-	Oculus	Xbox X|S
-	-	PlayStation VR	-

Tab. 3-1 Přehled softwarových platforem Unity 3D [17]

3.2. Servery

Z hlediska serverového řešení má vlastní systém Unity 3D Multiplayer and Networking. Tento systém je masivně a tvárně navržený pro vytvoření sítí (Netcode for GameObjects). V tomto systému má vývojář zodpovědnost za prvky komunikace v síti, které jsou jinak prováděny automaticky. Vývoj systému je rychlý, díky komunitě Unity 3D, která poskytuje textové i video návody a dokumentaci přímo z oficiálních stránek. [18]

Dříve byla pro Unity 3D k dispozici architektura UNet, která zastarala a de facto více než tři roky nebyl k dispozici nástupce. Výše uvedený „oficiální“ nástupce (Netcode for Gameobjects) byl vydán teprve nedávno a stále se vyvíjí. Při začátku tvorby této bakalářské práce vůbec nebyla k dispozici první verze. Z toho vyplývá i další zásadní nevýhoda: jedná se o zejména slabší dokumentaci a malé množství příkladů, což může vést k problémům zejména u větších projektů. Pro další práci (uvedenou zejména v kapitole Výběr vhodné varianty) nebudeme tento systém uvažovat.

Opusťme však nyní knihovny/architektury pro síťovou komunikaci a podívejme se obecně, jak je možné pracovat se síťovou komunikací.

3.2.1. Non-Authoritative Server

Neautoritativní server nekontroluje výsledek každého uživatelského vstupu. Klienti sami zpracovávají uživatelské vstupy a veškerou logiku lokálně a poté odesílají výsledek všech určených akcí na server. Server pak synchronizuje všechny akce se stavem světa. Z hlediska návrhu je to jednodušší implementace, protože server ve skutečnosti pouze předává zprávy mezi klienty a neprovádí žádné další zpracování nad rámec toho, co dělají klienti.

Není potřeba žádných predikčních metod, protože klienti sami zpracovávají veškerou fyziku a události a předávají serveru, co se stalo. Jsou vlastníky svých objektů a jsou jedinými agenty, kteří mohou posílat lokální modifikace těchto objektů po síti.

3.2.2. Authoritative Server

Přístup autoritativního serveru vyžaduje, aby server prováděl veškerou simulaci světa, aplikoval herní pravidla a zpracovával vstupy od hráčských klientů. Každý klient posílá své vstupy (ve formě stisků kláves nebo požadovaných akcí) na server a průběžně od něj dostává aktuální stav hry. Klient sám nikdy neprovádí žádné změny stavu hry. Místo toho sdělí serveru, co chce udělat, a server pak požadavek zpracuje a odpoví klientovi, aby vysvětlil, co se v důsledku toho stalo.

V zásadě existuje vrstva oddělení mezi tím, co chce hráč udělat, a tím, co se skutečně stane. To umožňuje serveru vyslechnout každý požadavek klienta a teprve poté se rozhodnout, jak aktualizovat stav hry.

Výhodou tohoto přístupu je, že klientům výrazně ztěžuje podvádění. Klienti například nemají možnost podvádět tím, že serveru (a tím i ostatním klientům) sdělí, že nepřítel byl zabit, protože toto rozhodnutí nemohou učinit sami. Mohou serveru pouze sdělit, že došlo ke střelbě ze zbraně, a odtud je na serveru, aby určil, zda došlo k zabití, či nikoli. [20]

3.3. Komunikace

Síťová komunikace slouží k vyměňování informací mezi počítačem a serverem nebo počítačem a počítačem. Na základě těchto informací pak může fungovat multiplayerová aplikace.

Existují dvě relevantní metody: Vzdálené volání procedur a synchronizace stavu. Není neobvyklé, že se obě metody používají na různých místech jedné simulace (hry). [20]

3.3.1. Remote Procedure Calls

Vzdálená volání procedur (RPC) se používají k vyvolání funkcí v jiných počítačích v síti, ačkoli "síť" může také znamenat kanál zpráv mezi klientem a serverem, pokud jsou oba spuštěny na stejném počítači. Klienti mohou posílat RPC na server a server může posílat RPC jednomu nebo více klientům. Nejčastěji se používají pro akce, ke kterým dochází zřídka. Například pokud klient stiskne spínač a otevře dveře, může serveru poslat RPC s informací, že

dveře byly otevřeny. Server pak může poslat další RPC všem klientům a vyvolat jejich místní funkce k otevření stejných dveří. Používají se pro správu a provádění jednotlivých událostí. [20]

3.3.2. State Synchronization

Synchronizace stavu se používá ke sdílení dat, která se neustále mění. Nejlepším příkladem je pozice hráče v akční hře. Hráč se neustále pohybuje, běhá, skáče a tak dále. Všichni ostatní hráči v síti, i ti, kteří tohoto hráče lokálně neovládají, potřebují vědět, kde se nachází a co dělá. Neustálým předáváním údajů o poloze tohoto hráče může hra přesně reprezentovat tuto polohu ostatním hráčům.

Tento druh dat je pravidelně odesílán po síti s danou periodou, většinou jsou to řádově milisekundy. Jelikož má síť omezenou šířku pásma je třeba data odesílat v co nejúspornějším formátu. Tím se zamezí přetížení sítě, nebo prodlevám ve vykreslování pohybu. [20]

4. Cíle projektu

Téma vzdálené komunikace pomocí VR je velmi zajímavé a lze i říci, že jsou některé aspekty stále neprobádané. Je podstatné si říci, že je potřeba nepodceňovat osobní komunikaci na projektech, i v době, kdy osobní setkání a přítomnost v práci není možná. K realizaci kolaborace ve VR, tak jak bude dále navrhována, jsou zapotřebí virtuální brýle. Jak již bylo zmíněno v kapitole 4 Unity 3D je pro vývoj těchto aplikací ideální jelikož přímo podporuje oba druhy typů serverů a je možné dokonce kombinovat dvě varianty komunikace. To může být realizováno nejen pomocí nativních knihoven, ale také dokonce pomocí jiných (někdy sofistikovanějších) řešení, které budou dále diskutovány.

Jak již bylo uvedeno bude výsledná prototypová aplikace zaměřena na kolaboraci ve VR vyvíjena v prostředí Unity 3D. Bude nutné nejprve vybrat vhodnou síťovou technologii/architekturu, a sice takový, který bude splňovat obecné požadavky pro realizaci VR kolaborace. Dále bude dále popsán postup nastavení projektu pro Unity 3D a implementace vybraného serveru. V aplikaci bude vytvořena místnost s 3D modelem, kde bude zprostředkována audio komunikace a důraz bude také kladen na gestikulaci. Aplikace bude primárně tvořena pro virtuální brýle značky Oculus, jelikož je má autor k dispozici. V dalších kapitolách bakalářské práce budou popsány postupy vytváření aplikace, popsání použitých balíčků a okomentované skripty.

Z uživatelského hlediska aplikace bude umět uživatele přihlásit do virtuální místnosti se stolem na kterém bude umístěn 3D model, kde se potká s ostatními uživateli. Na základě předchozí výzkumů [21] se ukazuje, že není podstatné, aby virtuální avatar byl fotorealistický. Soustředíme se tedy jen na podstatné elementy: Uživateli budou vidět virtuální ruce a místo hlavy bude mít kouli. Uživatel bude moci komunikovat se spoluúčastníky slovně a možnost gestikulace zjednoduší diskuzi nad specifickou částí 3D modelu. S modelem bude možno interagovat. Prostředí bude navrženo tak aby v něm mohla nerušeně probíhat schůze.

Navazujícím vývojem, by aplikace mohla umožňovat výměnu zobrazeného modelu. To umožní srovnání konkurenčních modelů v rámci jedné schůze. V rámci dalšího vývoje by mohla být zavedena rotace, animace objektu, popřípadě jeho rozpad na části. Předpokladem je rozvíjen vyvinuté řešení i po dokončení prací na této bakalářské práci.

5. Výběr vhodné varianty

Pro vhodný výběr varianty architektury (knihovny) síťové komunikace bude zásadní stanovit určitá kritéria, která budou dodržena při výběru. Kritéria budou stanovena na základě předchozích zkušeností, poznatků a obecných požadavků ze strany zadavatele (KPV).

Po vybrání vhodné varianty, bude u zvoleného serveru zmíněno, u jakých bodů uspěl, jaké jsou jeho výhody a nevýhody, jeho typ komunikace a možnosti programovacích jazyků.

5.1. Body pro výběr

Pro výběr vhodné varianty bylo zapotřebí stanovit body, na základě, kterých vybereme vhodnou sadu knihoven pro vytvoření zadané síťové komunikace. Nejdůležitějším vstupním limitujícím faktorem pro zvolení správné knihovny byla její kompatibilita s herním enginem Unity 3D. Druhým bodem byla možnost využívat zvolenou knihovnu s bezplatnou licencí pro nekomerční účely. Třetím bodem byla účelnost použití sady knihoven pro vytvoření masivní online aplikace pro více členů (MMO). Dále mezi hlavní body patřila stabilita a několik vedlejších bodů – např. podpora od komunity vývojářů a využití programovacího jazyka C#.

Je mnoho vhodných architektur, které odpovídají výše zmíněným bodům. Proto byly využity především komunitní fóra, která se zabývají vývojem her. Vývojáři zde zmiňují vlastní úskalí s vývojem aplikací, a i jejich nejvhodnější řešení. Následně z toho vyšly tyto sady knihoven: SmartFoxServer 2x [22], Photon Server [23], Lidgren [24] a Unity Sockets [25]:

SmartFoxServer je komplexní SDK pro rychlý vývoj multiplayerových her a aplikací pro Unity 3D, HTML5, iOS, Android, Java, Universal Windows Platform, Adobe Flash/Flex/Air, C++ a další. Zrodil se v roce 2004 a od té doby se neustále vyvíjí, dnes je předním middlewarem pro tvorbu rozsáhlých multiplayerových her, MMO a virtuálních komunit. [26]

Photon Server je multiplatformní vývojový framework, který je určen pro vývoj her (především v Unity 3D). Byl navržen německou společností ExitGames. Poskytuje knihovny jak pro klientskou taky i pro serverovou část. Podporuje širokou škálu platforem.

Lidgren.Network je síťová knihovna pro framework .NET, která využívá jediný socket UDP a poskytuje jednoduché rozhraní API pro připojení klienta k serveru, čtení a odesílání zpráv. Podporuje platformy Linux, Mac a OSX.

Unity Sockets Knihovna podporuje přijímání a vysílání událostí s nulovým nebo jedním datovým parametrem. JSON-Object-Data, která jsou přijímána ze serveru, jsou serializována do řetězce (formát JSON). Obyčejná textová data jsou předávána v nezměněné podobě. Binární užitečná zatížení nejsou podporována. ACK rovněž není podporováno.

Pro zjištění informací a následné vybrání vhodně varianty bylo nutné prohledat oficiální stránky knihoven a komunitní fóra. Na základě těchto informací byla vytvořena hodnotící tabulka:

	SmartFoxServer	Photon Server	Lidgren	Unity Sockets
Bezplatná licence	100 uživatelů	100 uživatelů	Neomezeno	Neomezeno
Účelnost pro MMO aplikace	Splňuje	Splňuje	Splňuje	Nesplňuje
Stabilita	Stabilní	Stabilní	Stabilní	Stabilní při méně náročných aplikacích
Komunita	Velká	Velká	Střední	Velká
Programovací jazyk	Java	C#	C#	C#

Tab. 5-1 Body pro zvolení vhodné sady knihoven

5.2. Vybrané řešení

Na základě výše uvedené hodnotící tabulky se jeví jako nejvhodnější sada knihoven **Photon Server**, která splňuje všechny předdefinované body zmíněné v podkapitole níže. Disponuje také dobře zpracovanými tutoriály a podporuje velké množství klientských platforem (např. Android, iOS, Linux, Unity 3D, .NET). Klient může používat různé platformy, zatím co komunikaci mezi server a klientem probíhá v reálném čase. Komunikace využívá protokoly UDP/TCP, které mají malou šířku pásma a umožňují velmi rychlou serializaci. Vývoj s Photon Serverem dovoluje využít libovolný jazyk .NET, jako je C# nebo dokonce jazyk C++. Nástroje jako Visual Studio nebo Redgate Profiler umožňují rychlý vývoj a ladění.

Vývojové softwary a klientské platformy, na kterých je možné využít Photon framework jsou zmíněny v tabulce níže. [23]

Podporované Platformy	Frameworky pro vývoj v určitých platformách
.NET	Windows 8 RT a Phone, .NET
Unity 3D	Při vyexportování z vývojového prostředí jsou všechny podporované.
Xamarin	Při vyexportování z vývojového prostředí jsou podporované iOS, Android, Mac OS
Marmalade	Při vyexportování z vývojového prostředí jsou všechny podporované.
Unreal Engine	Windows, Mac OS X, iOS, Android NDK
Corona	Při vyexportování z vývojového prostředí jsou podporované Kindle, Windows Phone, Android, iOS
Cocos2d-x	Marmelade, Windows, Android NDK, iOS
Samostatné Platformy	Flash, iOS, Linux, JavaScript, Android, Android NDK, Mac OS X, Windows, PlayStation Mobile

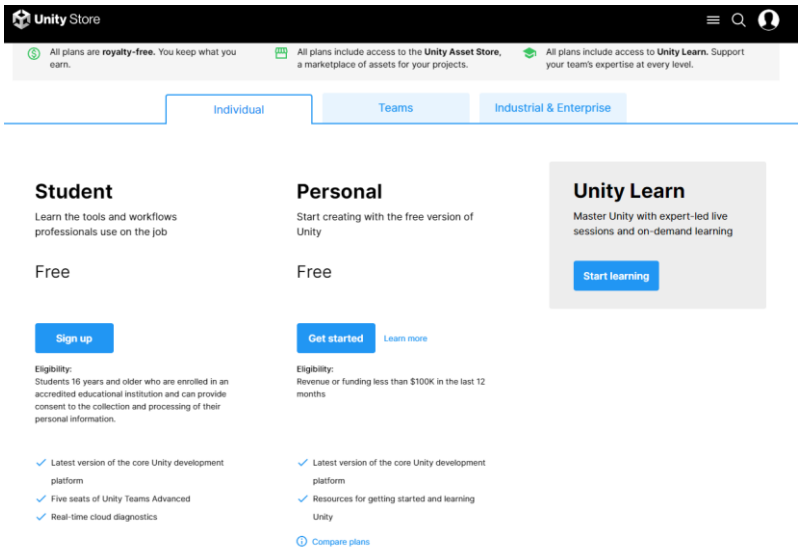
Tab. 5-2 Vývojové prostředí pro Photon klienty a podporované platformy

6. Návrh prototypu ukázkové aplikace pro virtuální realitu

K vytvoření návrhu ukázkové aplikace místnosti pro schůzky je zapotřebí stáhnout program Unity 3D a správně ho nakonfigurovat pro efektivní práci.

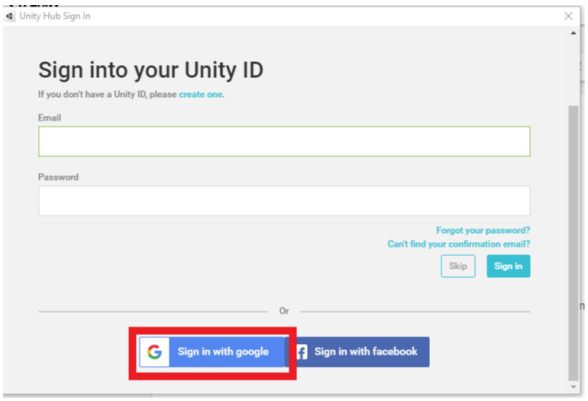
6.1. Instalace softwaru Unity 3D a vytvoření projektu

K instalaci softwaru můžeme použít libovolný internetový prohlížeč. V prohlížeči přejdeme na stránky https://store.unity.com/#plans-individual, kde si vyhledáme individuální verzi a následně zvolíme verzi „Personal“. (viz. Obr. 6-1) Tato verze licence je postačující pro tvorbu projektu a zdarma pro uživatele, kteří na projektech vytvořené v tomto programu nevydělávají přes 100 000 dolarů hrubého ročně. Pokud by tato částka byla přesažena, uživatel si musí zakoupit licenci Unity Pro, která stojí momentálně 1 800 dolarů za rok.



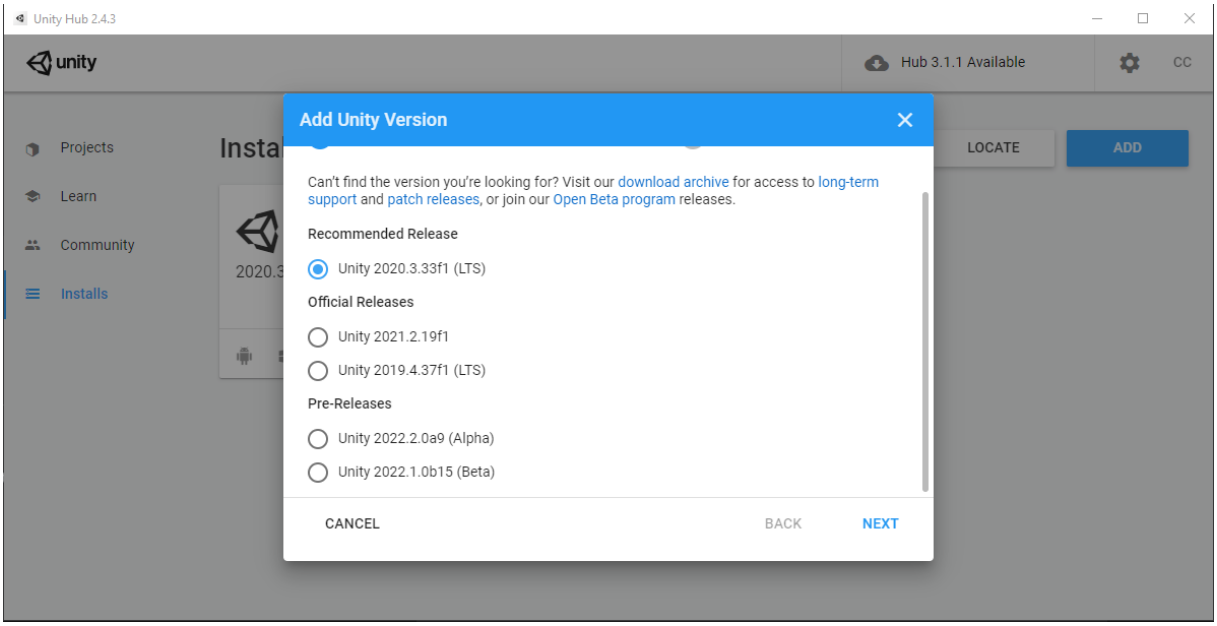
Obr. 6-1 Výběr Personal licence [27]

Po instalaci ze stránek Unity se nám do počítače nainstaluje Unity Hub, což je platforma pro přihlášení účtu a instalaci verzí Unity 3D. Pro přihlášení máme možnost zvolit přihlášení pomocí Google účtu, Facebookového účtu nebo účet vytvořit přímo u společnosti Unity 3D. Osobně jsem zvolil přihlášení přes Google účet, z důvodu přehlednosti upozornění na emailu. (viz. Obr. 6-2)



Obr. 6-2 Přihlášení do Unity Hub

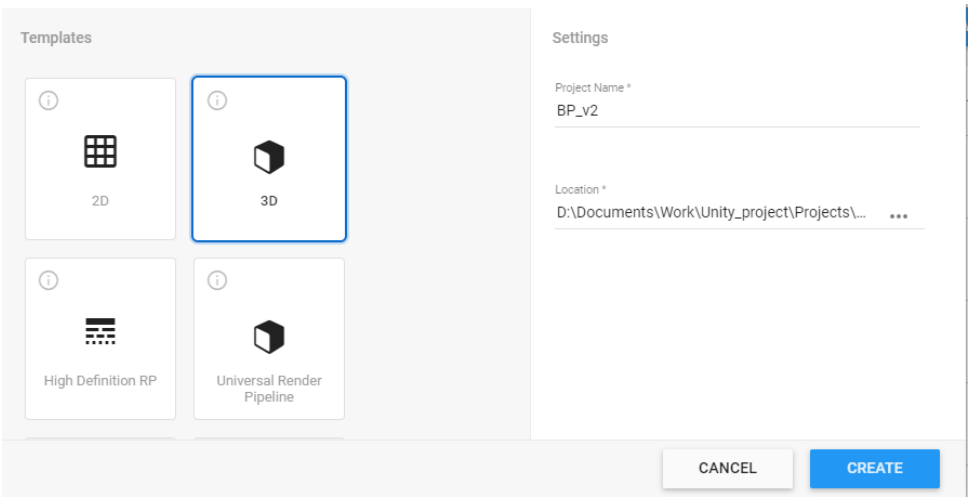
Dalším krokem je výběr vhodné verze Unity 3D, ve které budeme pracovat. Na výběr máme hned z 5 různých variant. Pro náš projekt bude nejlepší použít co nejnovější verzi ale zároveň verzi s označením LTS. Označení LTS nám dává informaci o zaručené funkčnosti verze. Instalována bude verze Unity 2020.3.33f1 (LTS). V následném kroku vybereme moduly MS Visual Studio Community a podporu Android Support, která je zapotřebí k vytvoření aplikace pro mobilní zařízení a virtuální brýle. (viz. Obr. 6-3)



Obr. 6-3 Výběr verze Unity

6.2. Vytvoření a nastavení projektu v Unity 3D

Projekt vytvoříme v Unity Hub ve verzi, kterou jsme si v předešlém kroku nainstalovaly. Z důvodu, že aplikace je zaměřena spíše na programovací část, a ne na grafickou, bude postačující vybrat šablonu 3D. Kdybychom potřebovali dělat náročnější efekty, poté by se vybrala možnost URP (Universal Render Pipeline). (viz. Obr. 6-4)



Obr. 6-4 Výběr šablony pro projekt

Nyní se seznámíme s prostředím v Unity 3D v našem projektu. (viz. Obr. 6-5)

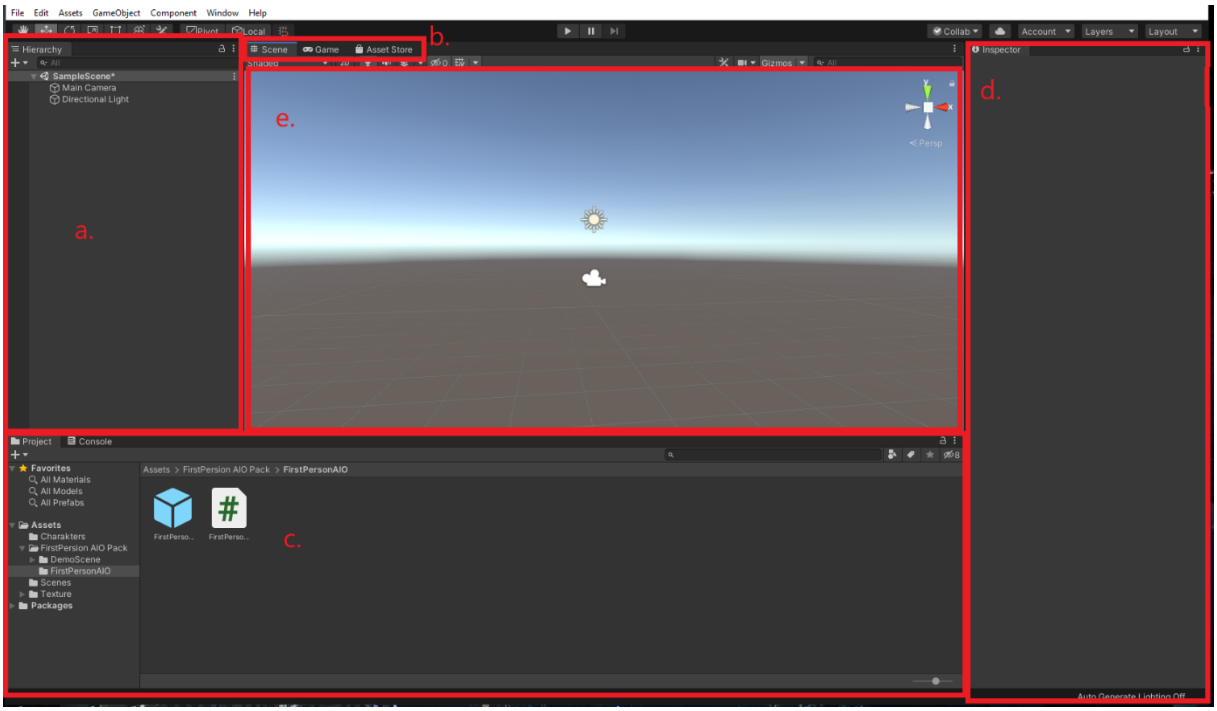
V poli „a.“ nalezneme hierarchii subjektů, které jsou již použity ve hře.

V poli „b.“ máme hned tři záložky. První záložka „Scene“ slouží k tvorbě hry. Druhá „Game“ dovoluje naši hru spustit a odzkoušet. V poslední záložce „Asset Store“ (tzv. Obchod s aktivy) se nám zobrazuje nabídka věcí, které si můžeme doinstalovat do našich Assetů.

V poli „c.“ se nám zobrazují námi přidané assety, z kterých můžeme vybírat a přidávat je do naší hry. Například charaktery, textury, objekty, …

V poli „d.“ si po označení konkrétního subjektu můžeme nastavit jeho vlastnosti.

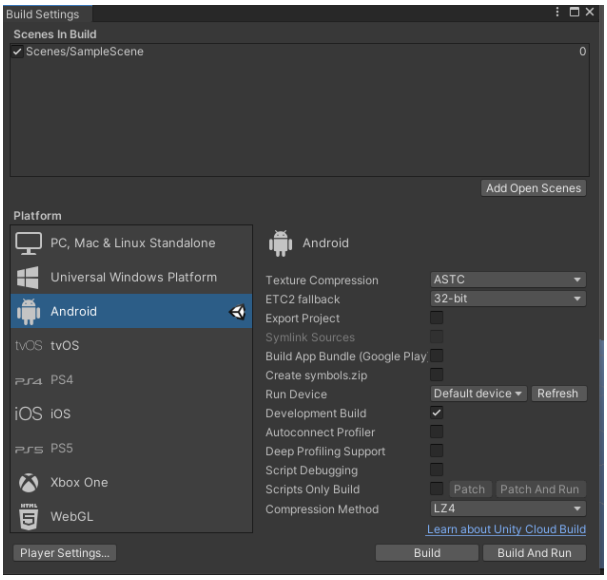
V poli „e.“ je aktivní plocha celého dění, kde se můžeme pohybovat.



Obr. 6-5 Prostředí v Unity 3D

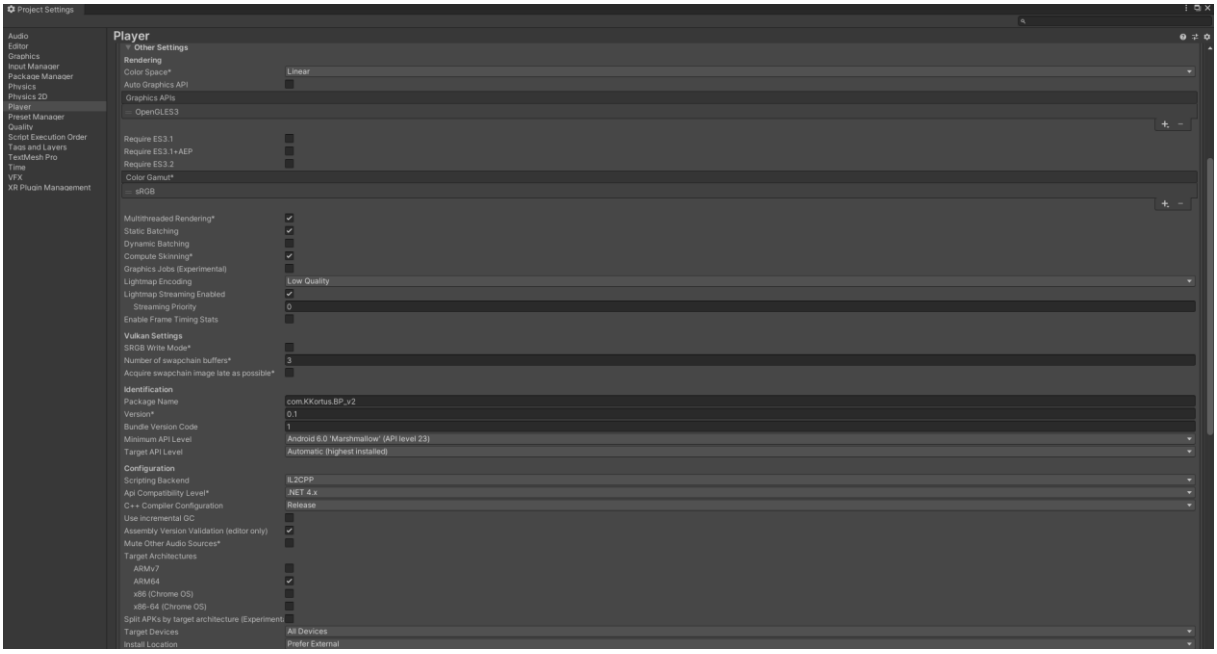
Pro nastavení našeho projektu pro správnou funkcionalitu na brýlích Oculus Quest 2 provedeme několik bodů.

Za 1. v „File => Build Settings“ přepneme náš projekt do platformy Android. Tento krok je důležité udělat hned ze začátku, protože Unity 3D bude přizpůsobovat celý projekt i assety pro tuto platformu. Pokud tak učiníme až po zkonstruování aplikace, mohla by tato část trvat nepříznivě dlouho. (viz. Obr. 6-6)



Obr. 6-6 Nastavení Build Settings

Za 2. nastavíme projekt tak aby byl co nejlépe optimalizovaný. V „File => Build Settings => Player Settings“ se dostaneme do okna „Project Settings“ na záložku „Player“, kde v nadpisu „Other Setting“ změníme „Color Space“ na lineární z důvodu jednodušeji vykreslovaných barev. Dále v „Graphics API“ necháme pouze možnost „OpenGLES3“ pro správný chod na virtuálních brýlí. „Minimum API Level“ přepneme na „Android 6.0 ‘Marshmallow‘“ pro podporu i na starších zařízení. V konfiguraci skriptů změníme „Scripting Backend“ na „IL2CPP“ a v „Api Compatibility Level“ na „.Net 4.x“ pro lepší optimalizaci skriptů v projektu. Zbytek nastavení necháme tak jak je v základu nastaveno. (viz. Obr. 6-7)



Obr. 6-7 Nastavení Player => Other Settings

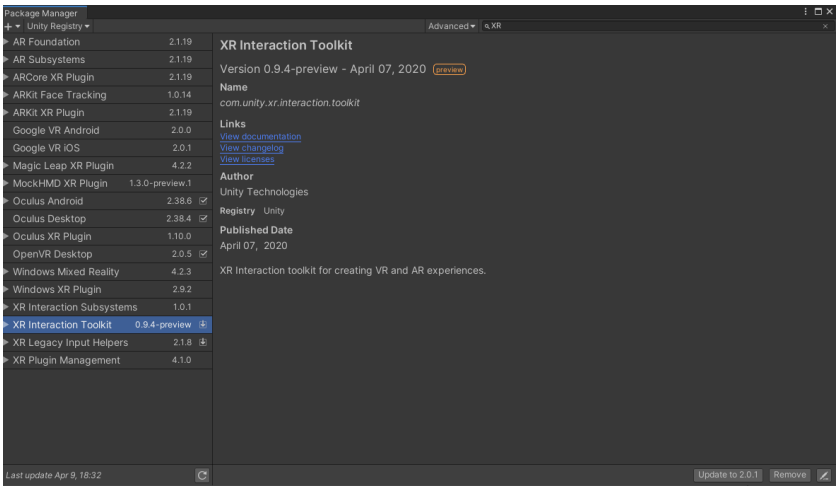
6.3. Aplikace Virtuální kolaborace

V rámci aplikace je vytvořena jedna scéna „BP_v2“. V kapitolách níže se budeme zabývat implementací balíčků, modelů, vytváření skriptů, samotných prvků ve scéně a na závěr vytvoření samotné aplikace.

6.3.1. Implementace a nastavení balíčku XR

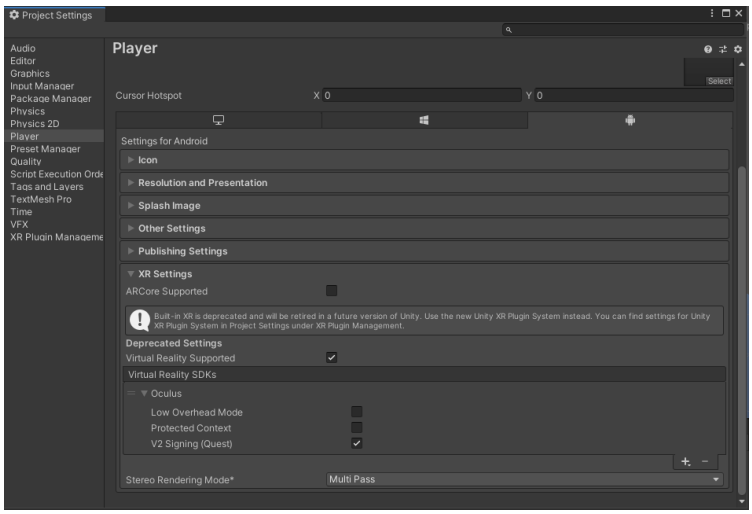
Pro možnost se v aplikaci vyskytovat pomocí virtuálních brýlí je nutno přidat asset, který nám virtuální brýle do aplikace zahrne. V takovém případě je zvolen balíček s XR kamerou, která má již přednastavené prvky pro pohyb v aplikaci a promítnutí ovladačů.

Pro přidání XR assetu musíme vstoupit do „Window => Package Manager“ zde vybereme možnost „Unity Registry“ a následně pomocí vyhledávání najdeme balíček „XR Interaction Toolkit“, který pomocí projektového manažeru nainstalujeme. (viz. Obr. 6-8)



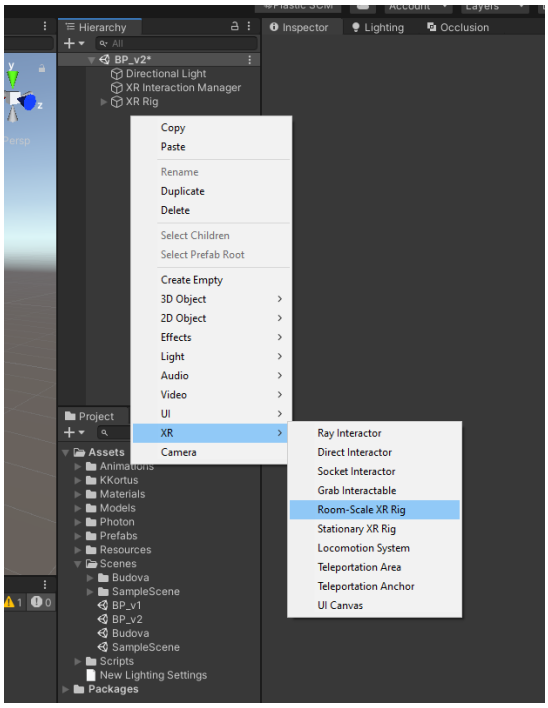
Obr. 6-8 Instalace XR kamery

Po instalaci XR balíčku ho přizpůsobíme pro práci s virtuální realitou. Přejdeme do „Edit => Project Settings => Player“ kde v záložce „XR Settings“ zapneme možnost „Virtual Reality Supported“. Protože hardwarem, který použijeme pro vytvořenou aplikaci jsou Oculus Quest 2, tak nastavíme možnost „Virtual Reality SDKs“ na „Oculus“. (viz. Obr. 6-9)



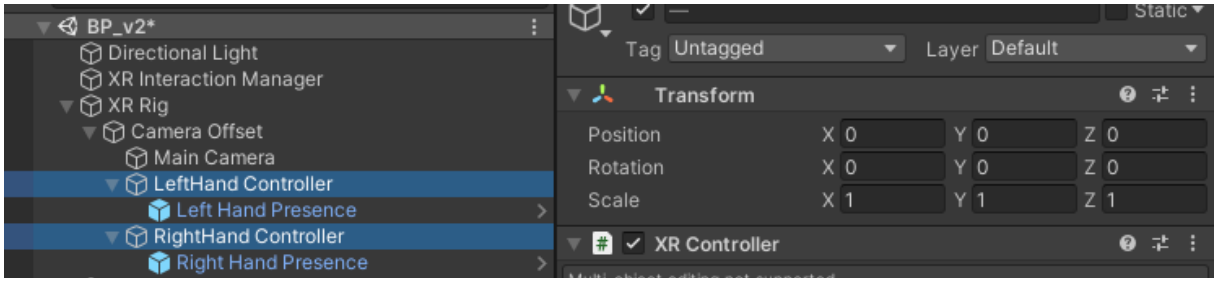
Obr. 6-9 XR pro VR

Ve scéně musí existovat vždy jen jedna hlavní kamera, která se bude přiřazovat virtuálním brýlím. Protože budeme využívat XR kameru, musí být původní kamera ve scéně odstraněna a nahrazena XR kamerou. Pravým tlačítkem v okně „Hierarchy“ vložíme XR kameru „XR => Room – Scale XR Rig“. V rámci této akce nám přibyde ke kameře ještě „XR Interaction Manager“, který pomáhá kameře ke správnému nastavení interakce s ovladači. (viz. Obr. 6-10)



Obr. 6-10 Vložení XR kamery

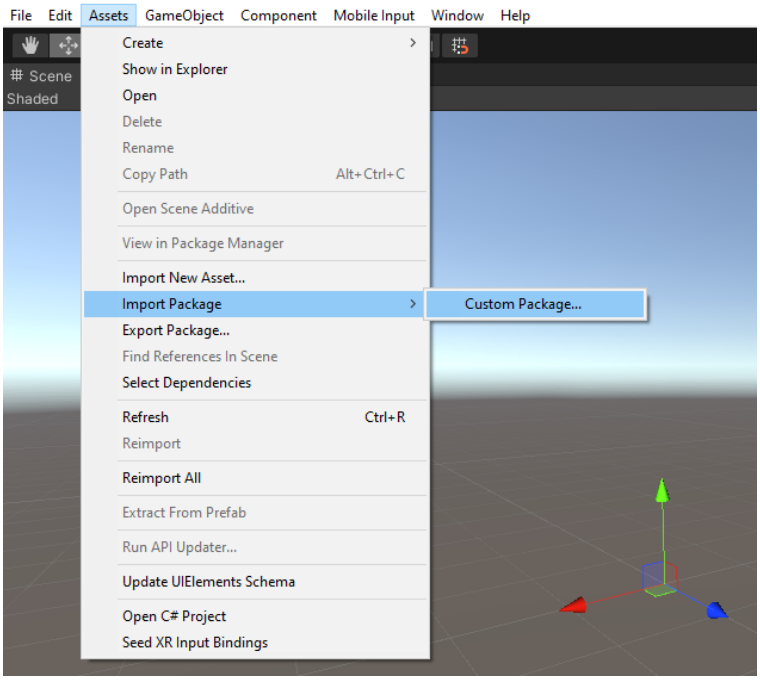
Následně odebereme z kamery skripty, které v naší aplikaci nebudou potřeba z důvodu zlepšení výkonu aplikace. Přejdeme do „XR Rig => Camera Offset“ a označíme si oba objekty pro ovladače. Poté z nich odstraníme všechny skripty kromě „XR Controller“. Následně pod ovladače přidáme modely rukou, které jsem použil již v dřívějších projektech. (viz. Obr. 6-11)



Obr. 6-11 Nastavení ovladačů XR

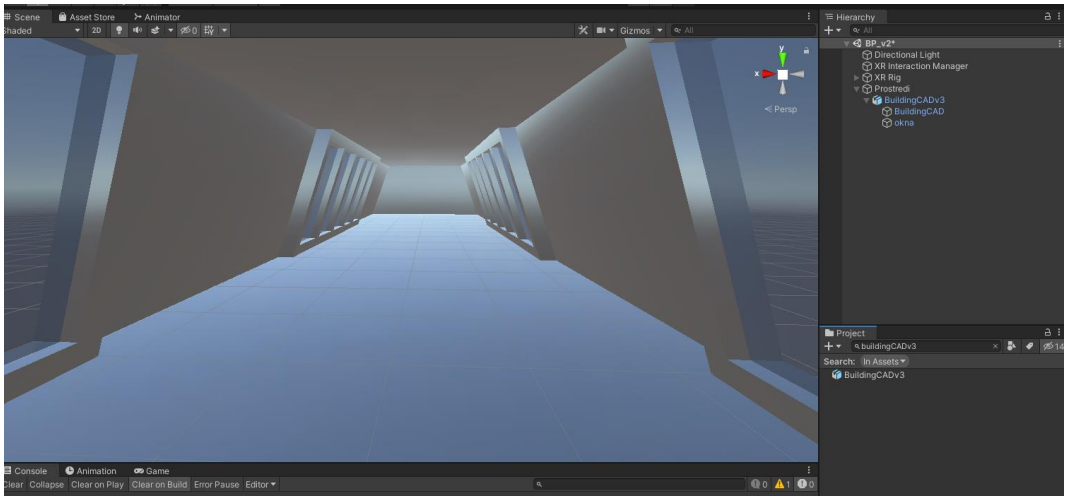
6.3.2. Tvorba vizuálního prostředí aplikace

Aby aplikace působila přirozeně na osoby, které se v ní budou nacházet, je důležité vytvořit kvalitní a uživatelský přívětivé prostředí. Z toho důvodů byl využit model budovy, která byla vytvořena již dříve pro jiné projekty. Pro vložení modelu přejdeme na „Assets => Import Package => Custom Package…“. (viz. Obr. 6-12)



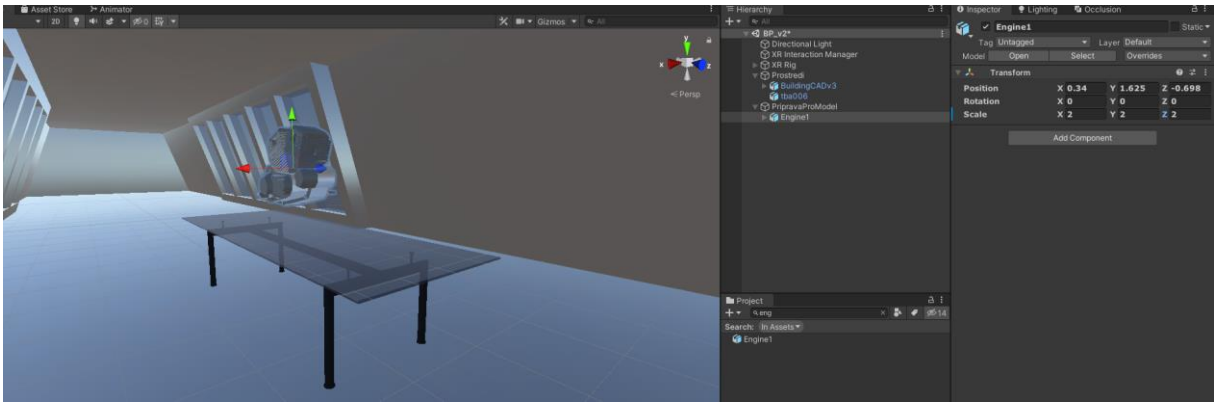
Obr. 6-12 Vložení modelu do prostředí Unity 3D

Hierarchie je v projektu velice důležitá část. Nachází se zde veškeré objekty scény. Při tvorbě větších aplikací, mohou být v hierarchii až stovky objektů, a proto je důležité v nich pro přehlednost udržovat pořádek. Na základě této úvahy byl vytvořen prázdný objekt s názvem „Prostredi“ kam budou vkládány všechny prvky pro tvorbu prostředí, s kterými nebude nijak interagováno v rámci aplikace. Následně do tohoto objektu z okna „Project“ byl přetažen myší model budovy. Budova byla v projektu přesunuta tak aby její podlaha souhlasila s nulovou y-ovou souřadnicí. Usnadní nám to nastavení kamery na výšku od podlahy. (viz. Obr. 6-13)



Obr. 6-13 Vložení modelu budovy

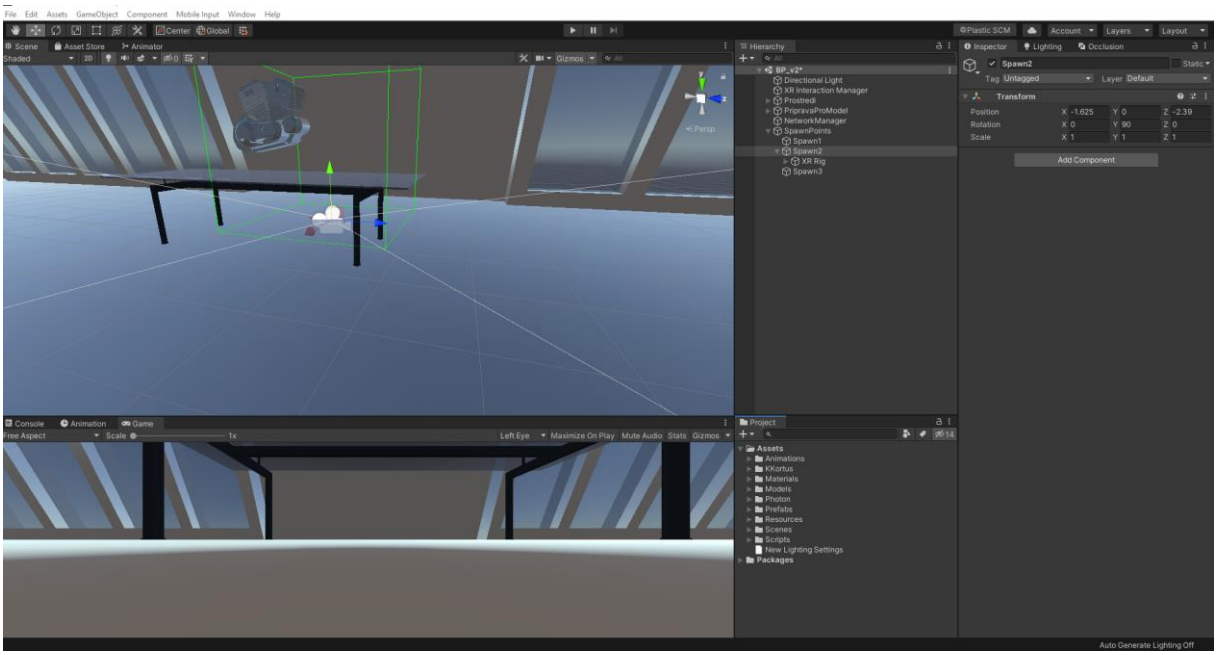
Na základě tvorby projektu kolaborace vložíme do scény také stůl a motor, který bude předmětem pro diskuzi v této místnosti. Vkládání objektů provedeme stejně jako to bylo u modelu místnosti. Protože budeme s modelem motoru chtít provádět v následujících krocích interakci, neumístíme ho do objektu „Prostredi“, ale vytvoříme si nový objekt „PripravaProModel“. (viz. Obr. 6-14)



Obr. 6-14 Vložení zbylých modelů

Dále si předpřipravíme body kolem stolu, kde budeme chtít hráče postupně po zapnutí aplikace umístit. Pokud bychom tak neučinili všichni hráči budou na jednom místě.

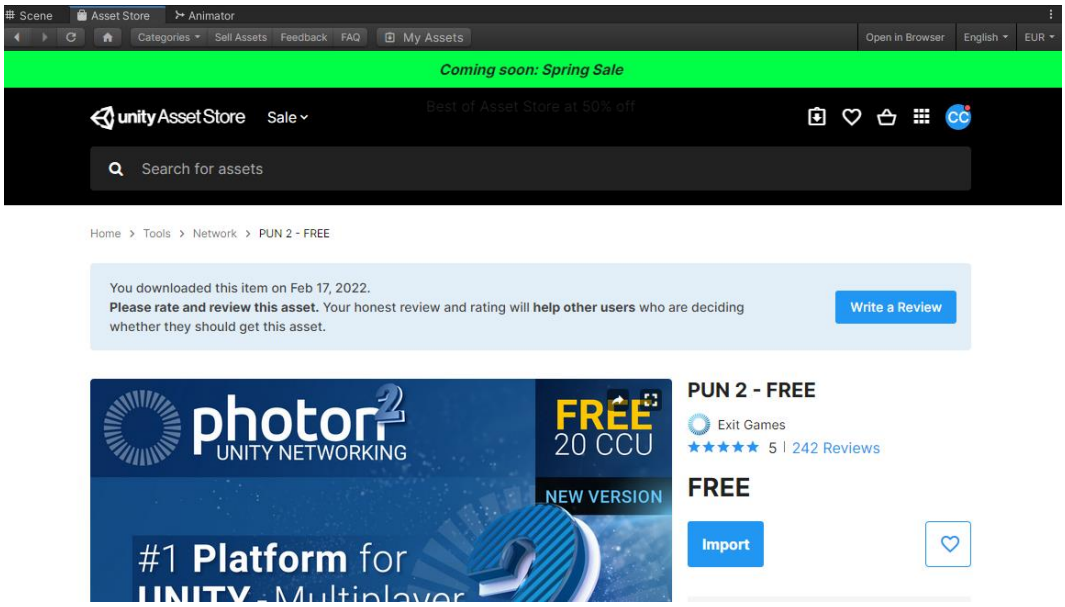
Pro tuto aplikaci postačí jako ukázka vytvořit tři body kolem stolu. V hierarchii vytvoříme prázdný objekt „SpawnPoints“. V tomto objektu vytvoříme další 3 prázdné objekty se jmény „Spawn1“, „Spawn2“ a „Spawn3“. Tyto objekty umístíme na podlahu a natočíme tak, aby když pod ně umístíme naši kameru, směřovala kamera ke stolu. (viz. Obr. 6-15)



Obr. 6-15 Umístění a nastavení Spawn Pointů

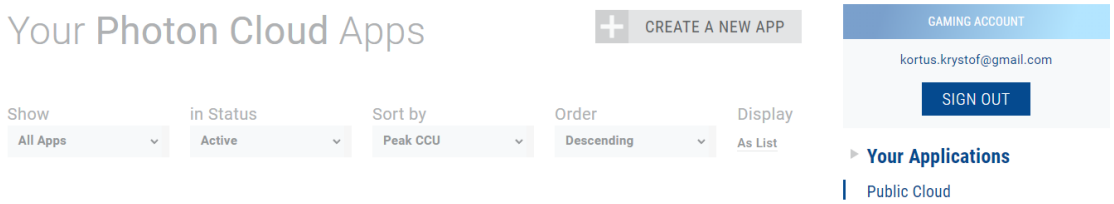
6.3.3. Implementace a nastavení Photon serveru

Jak bylo výše uvedeno, bude prototypová aplikace postavena na síťové architektuře Photon. Pro jeho přidání do Unity 3D projektu přejdeme na okno „Assets“ a zde vyhledáme „Photon“. Nejvhodnější variantou je asset „Photon PUN 2“, který je, jak jsem se již předtím zmínil, zdarma. Klikneme na tlačítko „Import“ a asset se nám přidá do našeho projektu. (viz. Obr. 6-16)



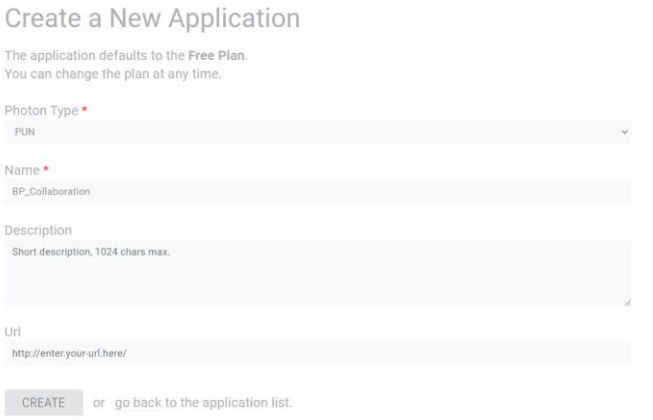
Obr. 6-16 Vkládání Photon serveru přes Asset Store

Po importu serveru nás asset Photon vyzve k zadání Photon ID, které získáme při vytvoření účtu na stránkách Photon, proto přejdeme na stránky https://dashboard.photonengine.com/en-US/, kde si vytvoříme účet. Po tvorbě účtu si na uvedených stránkách vytvoříme vlastní server pro naší aplikaci přes tlačítko „Create a new app“. (viz. Obr. 6-17)



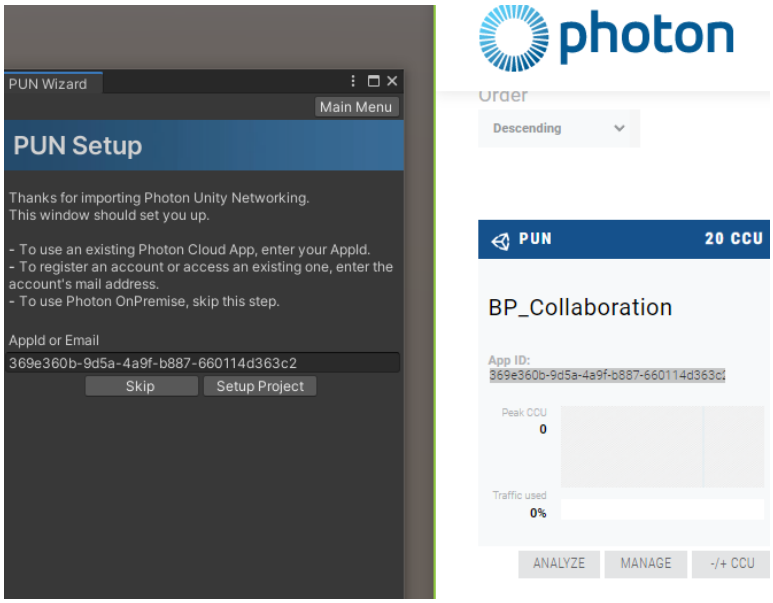
Obr. 6-17 Photon Server tvorba pro aplikaci

Při vytváření našeho serveru nastavíme „Photon Type“ na „PUN“, z důvodu použití v projektu. Následně popíšeme náš server v poli „Name“ a dáme vytvořit. (viz. Obr. 6-18)



Obr. 6-18 Nastavení serveru aplikace

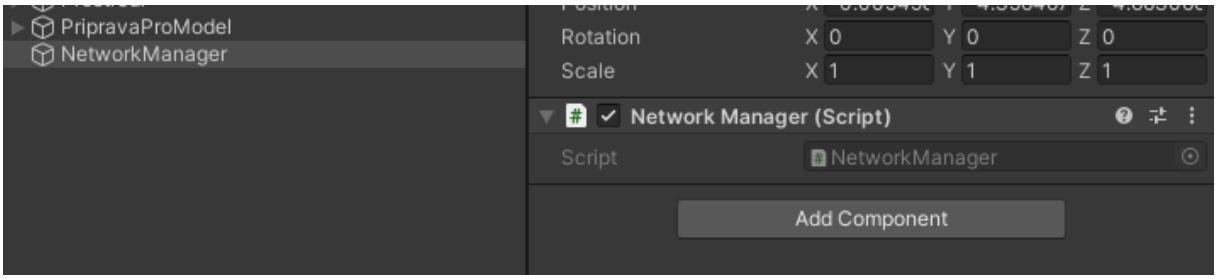
Po vytvoření serveru se nám vygeneruje „App ID“, které právě použijeme v našem projektu pro registraci. Vyplníme tedy naše „AppId“ do pole „AppId or Email“ v okně „PUN Wizzard“. (viz. Obr. 6-19)



Obr. 6-19 AppId pro Photon Server

6.3.4. Propojení objektů a aplikace se serverem

Nyní budeme pokračovat v Unity3D, ve kterém se budeme zabývat připojením uživatele do místnosti. Pro propojení aplikace vytvoříme prázdný objekt s názvem „Network Manager“, na který vytvoříme skript s názvem „NetworkManager“ pomocí „Add Component => New Script“. Tento skript je vytvářen v aplikaci Microsoft Visual Studio, která byla instalována zároveň s verzí Unity 3D a bude propojovat naší aplikaci se serverem. (viz. Obr. 6-20)



Obr. 6-20 Tvorba Network Manageru

Při tvorbě skriptu musíme využít namespace knihovny (pomocí using) serveru Photon pro využití jeho příkazů. Následně si popíšeme veškeré metody, které se nachází v daném skriptu. Úplný skript je k nalezení v 1. příloze.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;

public class NetworkManager : MonoBehaviourPunCallbacks
{
    void Awake()

    void ConnectToServer()

    public override void OnConnectedToMaster()

    public override void OnJoinedRoom()

    public override void OnPlayerEnteredRoom(Player newPlayer)
}
```

- metoda „Awake()“ => Zapříčiní volání kódu hned na začátku puštění aplikace. V této metodě budeme volat jako pod metodou „ConnectToServer“.

- metoda „ConnectToServer()“ => Realizuje připojení k serveru Photon. Následně je v metodě psán příkaz, který nám při puštění v Unity Editor hlásí, že se pokouší o připojení na server.

- metoda „OnConnnectedToMaster()“ => Připravuje připojení scény jako místnost do serveru a nastavuje její parametry (např.: počet hráčů). Zde je také přidán příkaz, který nás informuje o připojení na server.

- metoda „OnJoinedRoom()“ => Připojuje scénu jako místnost. Dále hláška o připojení místnosti na server.

- metoda „OnPlayerEnteredRoom(Player newPlayer)“ => Připojuje hráče do místnosti. Hráč musí být přiřazen pomocí jiného skriptu do lokální proměnné metody „newPlayer“.

Následně si na objekt vytvoříme ještě jeden skript s názvem „NetworkPlayerSpawner“. Do tohoto skriptu také dopíšeme knihovnu „Photon.Pun“. Úplný skript je k nalezení v 2. příloze.

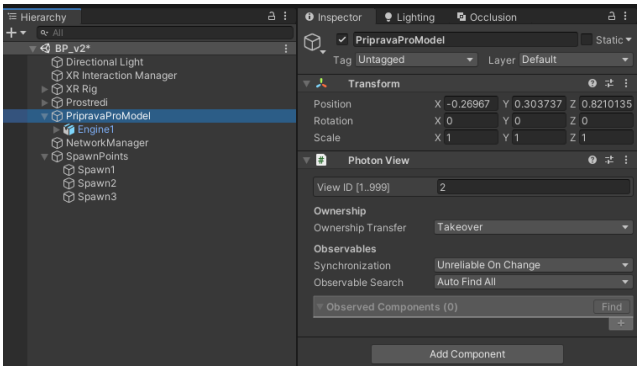
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;

public class NetworkPlayerSpawner : MonoBehaviourPunCallbacks
{
    private GameObject spawnedPlayerPrefab;

    public override void OnJoinedRoom()

    public override void OnLeftRoom()
}
```

- Proměnná „spawnedPlayerPrefab“ => nastavená privátně z důvodu nepotřeby volání v ostatních skriptech.

- metoda „OnJoinedRoom()“ => Připojí serverového hráče do hry jako „Network Player“ pomocí serveru. Přiřadí hráče proměnné „spawnedPlayerPrefab“.

- metoda „OnLeftRoom()“ => Při opuštění hráče z aplikace zničí hráčův prefabrikát.

Motoru, s kterým chceme pracovat v rámci serveru tak, aby všichni zúčastnění mohli vidět jeho pohyby, přiřadíme skript „Photon View“ od společnosti Photon. Protože v budoucnu budeme chtít přidat více modelů nebudeme tento skript přikládat přímo na objekt s motorem, ale na objekt jménem „PripravaProModel“. (viz. Obr. 6-21)

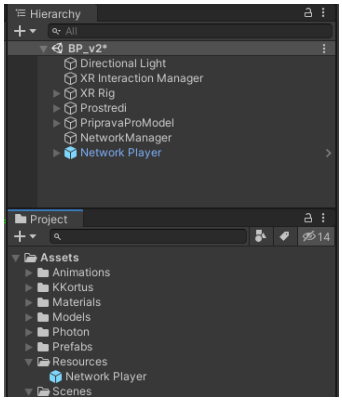


Obr. 6-21 Zviditelnění modelů pro multiplayer

6.3.5. Promítnutí hráče pro multiplayerovou aplikaci

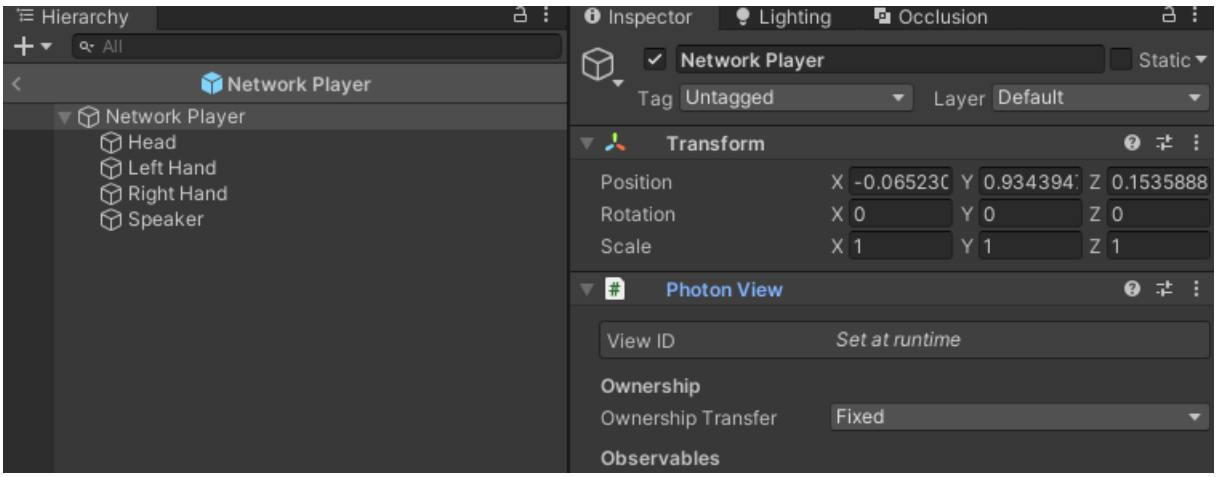
Nyní se budeme zabývat zviditelněním hráče pro server. To zahrnuje nastavení samotných komponent hráče a programování skriptů.

Vytvoříme si prefab (speciální typ komponenty, která umožňuje uložit plně nakonfigurované herní objekty do projektu pro opakované použití) hráče. V okně „Hierarchy“ vytvoříme prázdný objekt s názvem „NetworkPlayer“. Protože ho bude potřebovat vyvolávat pomocí příkazu Instantiate(), musí být vytažen do okna „Project“ do složky „Assets => Resources“. Přetažením objektu “NetworkPlayer“ z okna „Hierarchy“ do okna „Project“ a námi požadované složky z něj vytvoříme výše zmíněný prefabrikát. (viz. Obr. 6-22)



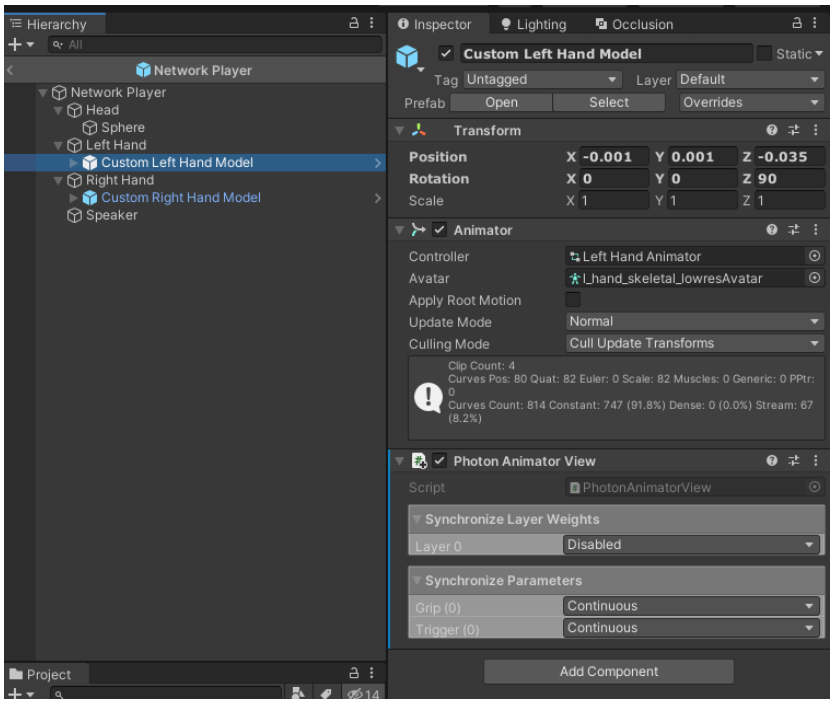
Obr. 6-22 Prefabrikát Network Player

Poklepáme na prefabrikát dvakrát levým tlačítkem myši, čímž je možné prefab editovat. Zde je nutné „zviditelnit“ objekt. Aby náš objekt byl viděn serverem musíme na něj přiřadit skript „Photon View“ od společnosti Photon, který je součástí implementovaného assetu. Zde si předpřipravíme prázdné objekty pro našeho hráče. Prázdné objekty se budou jmenovat „Head“, „Left Hand“, „Right Hand“ a „Speaker“. Aby náš objekt byl viděn serverem musíme na něj přiřadit skript „Photon View“ od společnosti Photon, který je součástí implementovaného assetu. Pro promítnutí pohybu rukou a hlavy pro ostatní serverové hráče naší aplikace přiřadíme skript „Photon Transform View“ a zaškrtneme políčka „Position“ a „Rotation“. (viz. Obr. 6-23)



Obr. 6-23 Přednastavení Network Player

Pro vizualizaci postavy si vytvoříme pod objektem „Head“ pravým tlačítkem 3D objekt typu „Sphere“ a odebereme z něho „SphereCollider“, protože ho nebudeme potřebovat. Pod objekt „Left Hand“ a „Right Hand“ přiřadíme již zmíněné ruce. Abychom viděli pohyb gest na rukách přidáme skript „Photon Animator View“ od Photon. (viz. Obr. 6-24)



Obr. 6-24 Zhmotnění postavy

Nyní vytvoříme skript na prefabrikátu „Network Player“, který se bude jmenovat „NetworkPlayer“. Tento skript bude mít za cíl promítnout našeho hráče kompletně do hry i s jeho interakcemi. Z důvodu rozsáhlosti skriptu budou nejdříve popsány proměnné s knihovnami a následně samotné metody. Úplný skript je k nalezení v 3. příloze.

Jako u předchozích skriptů dopíšeme Photon knihovny, ale také knihovny XR balíčku, protože pracujeme s hráčem, který je s XR kamerou úzce spjat.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR;
using Photon.Pun;
using UnityEngine.XR.Interaction.Toolkit;
using Photon.Realtime;
using System.IO;

public class NetworkPlayer : MonoBehaviour
{
    public Transform head;
    public Transform leftHand;
    public Transform rightHand;

    public Animator leftHandAnimator;
    public Animator rightHandAnimator;

    private PhotonView photonView;

    private Transform headRig;
    private Transform leftHandRig;
    private Transform rightHandRig;

    //Spawn Players
    private GameObject myPlayerAvatar;

    private Player[] allPlayers;
    private int myNumberInRoom;
```

- Proměnné „head“, „leftHand“ a „rightHand“ => Slouží k přiřazení těla avatara. Jsou druhu „Transform“, protože chceme zjišťovat jen jejich pozici.

- Proměnné „leftHandAnimator“ a „rightHandAnimator“ => Přiřadíme do nich animace rukou pro promítnutí do serveru.

- Proměnná „photonView“ => získání „PhotonView“ komponentu na objekt

- Proměnné „headRig“, „leftHandRig“ a „rightHandRig“ => Slouží k načítání objektů XR kamery pro našeho avatara.

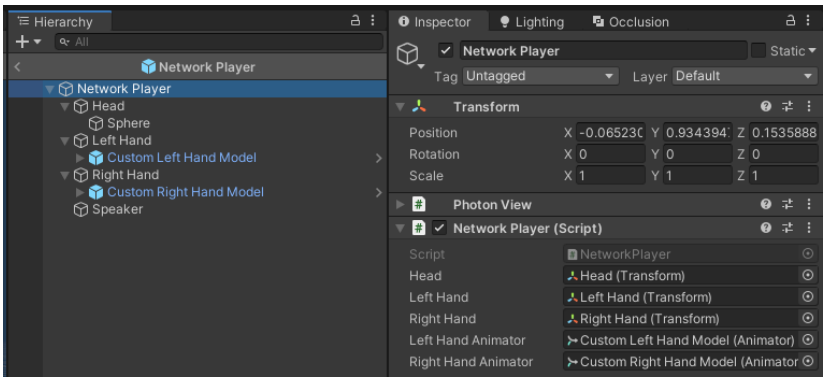
- Proměnná „myPlayerAvatar“ => Přiřadíme do ní XR Rig.

- Pole proměnných „allPlayers“ => Slouží pro získání výčtu hráčů v aplikaci. Je nastavena na bezrozměrné pole, a ne na konkrétní hodnotu pole, protože nevíme, kolik hráčů se bude v aplikaci vyskytovat.

```
    void Start()

    [PunRPC]
    public void SpawnPointCharacter(int point)

    void Update()

    void UpdateHandAnimation(InputDevice targetDevice, Animator handAnimator)

    void MapPosition(Transform target, Transform rigTransform)
}
```

- metoda „Start()“ => V této metodě přiřadíme komponentu „PhotonView“, najdeme kameru pro „mPlayerAvatar“, přiřadíme XR komponenty proměnným těla. Dále zjistíme počet hráčů ve hře a proměnné „myNumberInRoom“ přiřadíme toto číslo. V další části voláme přes metodu „photonView.RPC()“ metodu „SpawnPointCharacter(int point)“ pro umístění hráče na předdefinovanou pozici.
- metoda „SpawnPointCharacter(int point)“ => Je zařazena do funkce RPC, což znamená „Vzdálené volání procedur“. Metoda umístí hráče na místo podle jeho čísla pomocí skriptu „AU_GameController“, který si zmíníme později. Místa, kde budou hráči umístěni jsou ve scéně pod objektem „SpawnPoints“.
- metoda „Update()“ => Pracuje s tělem pomocí pod metody „MapPosition“ a animacemi rukou našeho avatara pomocí pod metody „UpdateHandAnimation“. Tato část promítá naše pohyby pro ostatní hráče v místnosti.
- metoda „UpdateHandAnimation(InputDevice targetDevice, Animator handAnimator) => Získává hodnoty animací rukou pro promítnutí do scény.
- metoda „MapPosition(Transform target, Transform rigTransform)“ => mapuje pozici XR Rig kamery.

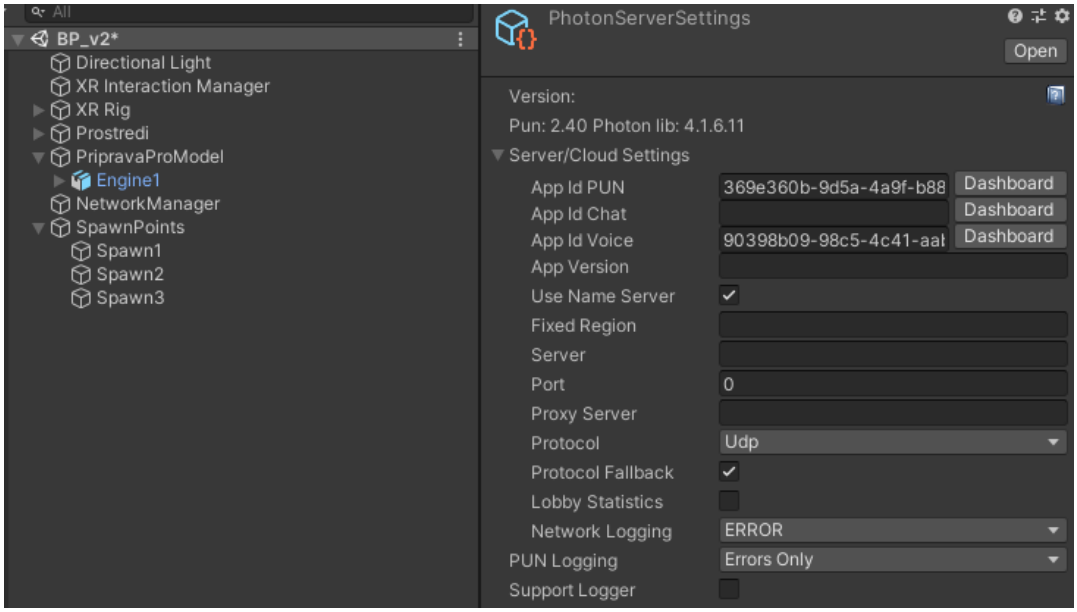
Následně skriptu přiřadíme do proměnných požadované objekty. (viz. Obr. 6-25)



Obr. 6-25 Vkládání objektů do proměnných skriptu NetworkPlayer

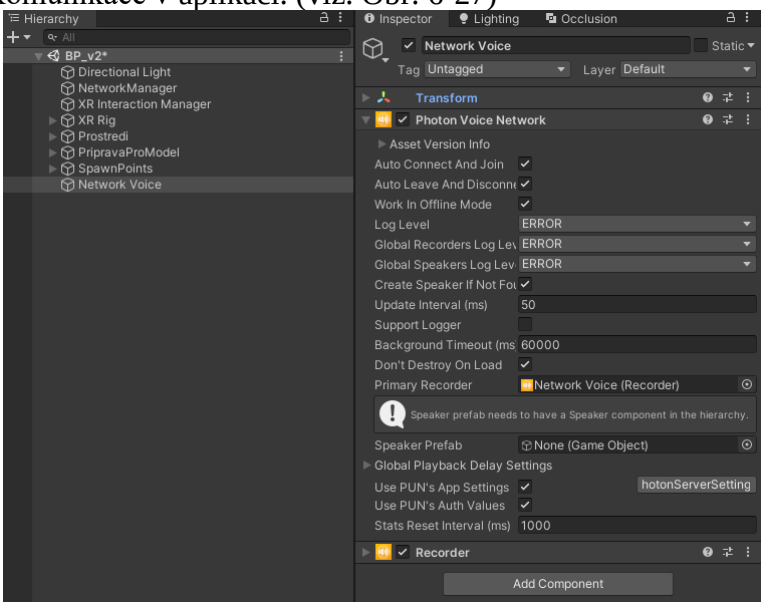
6.3.6. Audio vizualizace avatara

Pro komunikaci hráče s prostředím je nutno stáhnout další asset od firmy Photon s názvem „Photon Voice 2“ a vložit ho do projektu stejným způsobem jako tomu bylo u assetu „Photon PUN 2“. Na webových stránkách si přes náš účet vytvoříme nový server. Pro tentokrát v kolonce „Photon Type“ vybereme možnost „Photon Voice“ a pojmenujeme v kolonce „Name“ jako „BP_Voice“. Nyní se nám vytvoří nový zvukový server s vlastním „App ID“. Tento „App ID“ přiřadíme do nastavení Photonu v našem projektu do kolonky „App Id Voice“. (viz. Obr. 6-26)



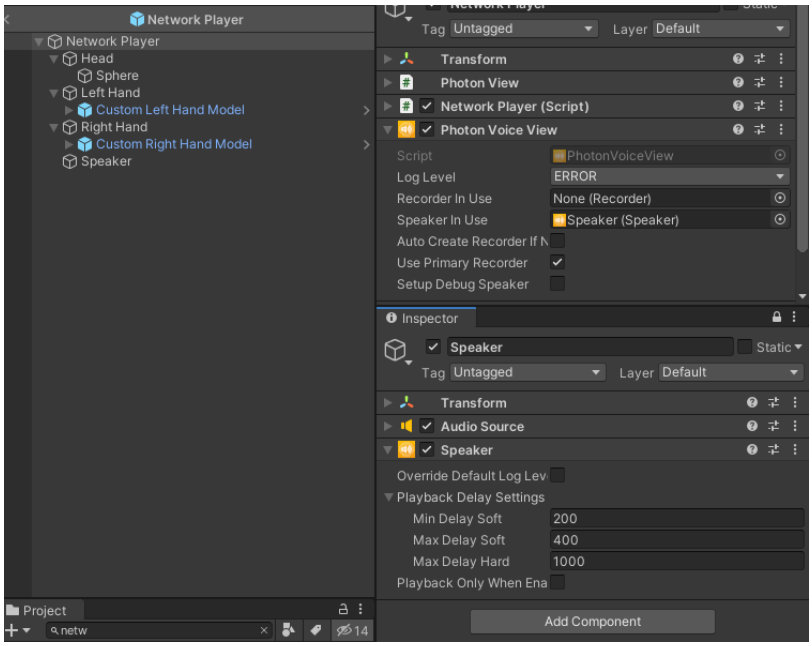
Obr. 6-26 Photon Voice App Id Voice

Dále ve scéně vytvoříme prázdný objekt s názvem „Network Voice“, na který připojíme skript „Photon Voice Network“ a „Recorder“ od společnosti Photon. Skript „Recorder“ přiřadíme do proměnné „Primary Recorder“ ve skriptu „Photon Voice Network“. Skripty nám zajistí nastavení komunikace v aplikaci. (viz. Obr. 6-27)



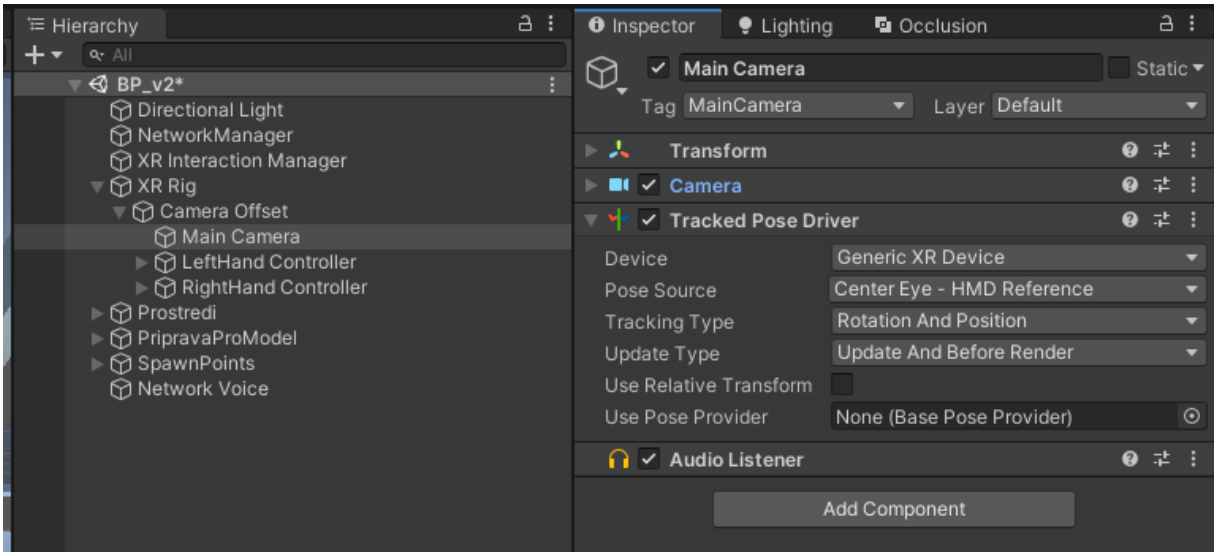
Obr. 6-27 Network Voice

Aby hráče bylo slyšet musíme nastavit i samotného avatara. Proto přejdeme na jeho prefabrikát a na objekt „Speaker“ přidáme komponentu „Audio Source“, na kterém změníme hodnotu „Spatial Blend“ z 0 na 1, „Min Distance“ na 10 a „Max Distance“ na 30. Tím docílíme prostorového zvuku. V prostoru, který jsme si definovali, uslyšíme okolní hráče. Následně přidáme skript „Speaker“ od společnosti Photon. Na objekt „Network Player“ přidáme skript „Photon Voice View“ a do proměnné „Speaker In Use“ přiřadíme náš „Speaker“. Hodnotu „Use Primary Recorder“ nastavíme na „True“. (viz. Obr. 6-28)



Obr. 6-28 Nastavení audia u avatara

Protože se náš avatar váže na XR kameru, tak pro poslech okolí musíme přiřadit na „Main Camera“ komponentu „Audio Listener“. (viz. Obr. 6-29)



Obr. 6-29 XR kamera – nastavení Audio Listener

Těmito kroky jsme zařídili komunikaci mezi hráči.

6.3.7. Místa pro umístění hráčů

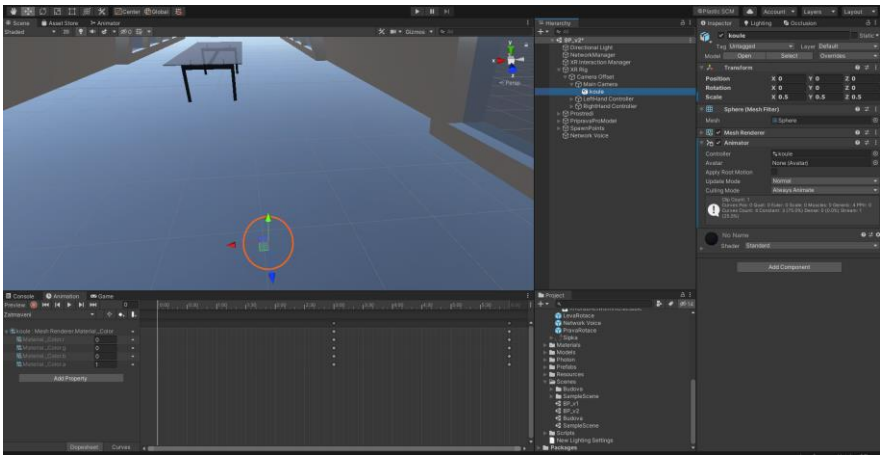
Jak jsme již zmínili v kapitole „7.3.5. Promítnutí hráče pro multiplayerovou aplikaci“, zde budeme pracovat se skriptem pro načtení všech našich Spawn pointů.

Vytvoříme si na objektu „SpawnPoints“ skript s názvem „AU_GameController“. Tento skript je volán ve skriptu „NetworkPlayer“, pro přiřazení konkrétního spawn pointu, podle hodnoty hráče na vstupu do místnosti.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AU_GameController : MonoBehaviour
{
    public static AU_GameController instance;

    public Transform[] spawnPoints;
    void Start()
    {
        spawnPoints = new Transform[this.transform.childCount];

        for(int i = 0; i < this.transform.childCount;i++)
        {
            spawnPoints[i] = this.transform.GetChild(i);
        }

        instance = this;
    }
}
```

Skript se zabývá načítáním objektů pod hlavním objektem, na kterém je skript nasazen. Takzvaně zjišťuje počet dětí rodičovského objektu, nastavuje podle počtu dětí velikost pole a následně přiřadí všechny jeho děti do pole „spawnPoints“.

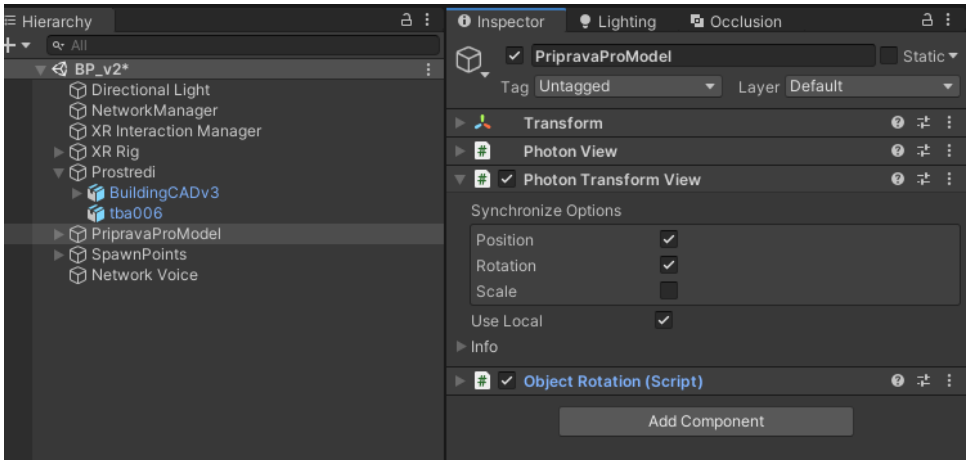
Protože by pro hráče mohlo být nepříjemné být teleportován na jiné místo, přiřadíme pod kameru kouli s otočenými normálami, kde pomocí animace docílíme postupného zprůhlednění změnou hodnoty „alfa“ v komponentě materiálu. Uživateli tak zprostředkujeme vjem tzv. Fade-In efektu. (viz. Obr. 6-30)



Obr. 6-30 Zprůhlednění ochrany přemístění kamery

6.3.8. Interakce s objekty v rodičovském objektu „PripravaProModel“

Pro interakci s modelem v aplikaci nastavíme ruční rotaci modelu, v našem případě modelu motoru. Pro promítnutí rotace, popřípadě pozice, pro všechny hráče je nutné objekt „PripravaProModel“ zahrnout do serveru. Proto využijeme skriptu „Photon Transform View“ od společnosti Photon a zajistíme, aby řádky „Position“ a „Rotation“ byly nastaveny na „True“. Aby s objektem mohlo být rotováno, vytvoříme si na „PripravaProModel“ skript s názvem „ObjectRotation“. (viz. Obr. 6-31)

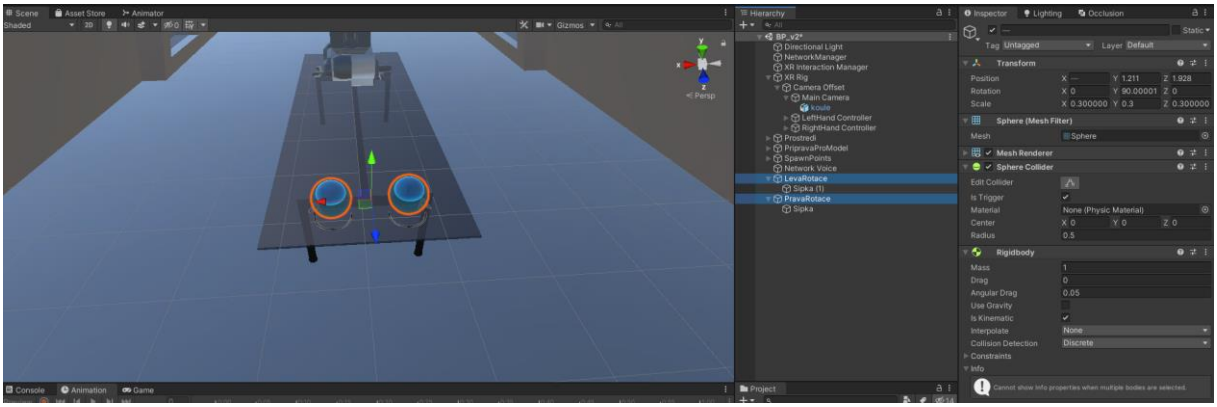


Obr. 6-31 Rotace modelu – hlavní objekt

Skript se zabývá, pomocí metody „LateUpdate()“, transformací tělesa v y-ové souřadnici o rychlosti proměnné „speedRotation“, která je ze základu nastavená na hodnotu 0.

```
using UnityEngine;
using System.Collections;

public class ObjectRotation : MonoBehaviour {

	public float speedRotation = 0f;

	void LateUpdate () {

		transform.Rotate(-Vector3.up * Time.deltaTime * speedRotation);
	}
}
```

Abychom s objektem mohli pohybovat dle vlastního uvážení, vytvoříme v čele stolu pro pořadatele schůzky ovládací prvky – zde dva objekty kulového tvaru. Na tyto objekty si vytvoříme materiál v okně „Project“. Pod koule umístíme obrázky pro orientaci točení. Z důvodu interakce s těmito objekty na ně přiřadíme komponenty „Sphere Collider“, kde nastavíme „Is Trigger“ na „True“ a „Rigidbody“, kde nastavíme „Is Kinematic“ na „True“. (viz. Obr. 6-32)

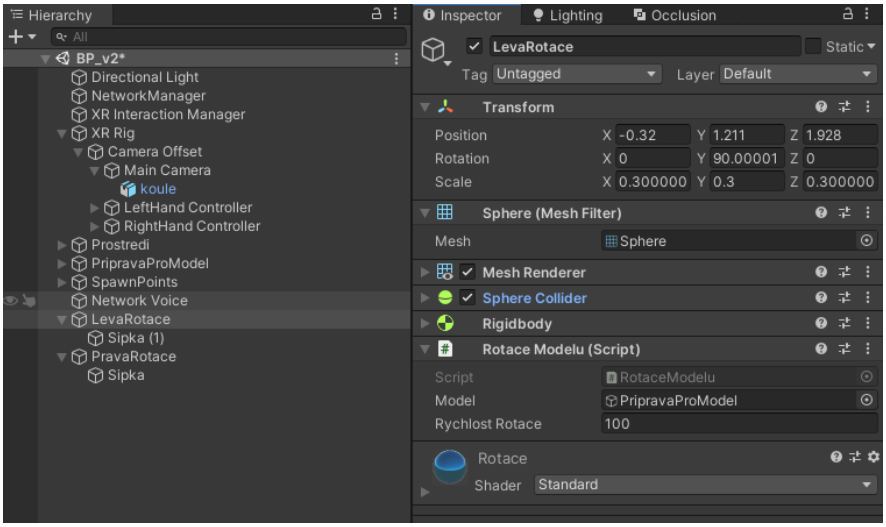


Obr. 6-32 Objekty pro pohyb modelu

Nyní si vytvoříme na těchto objektech skript pro pohyb modelu. Skript vytvoříme s názvem „RotaceModelu“. Tento skript bude vkládat předdefinovanou hodnotu zrychlení modelu do skriptu, který je na objektu „PripravaProModel“. Dít se tak bude v případě, že do koule vložíme ruku. Pokud z koule ruku vytáhneme model se přestane točit.

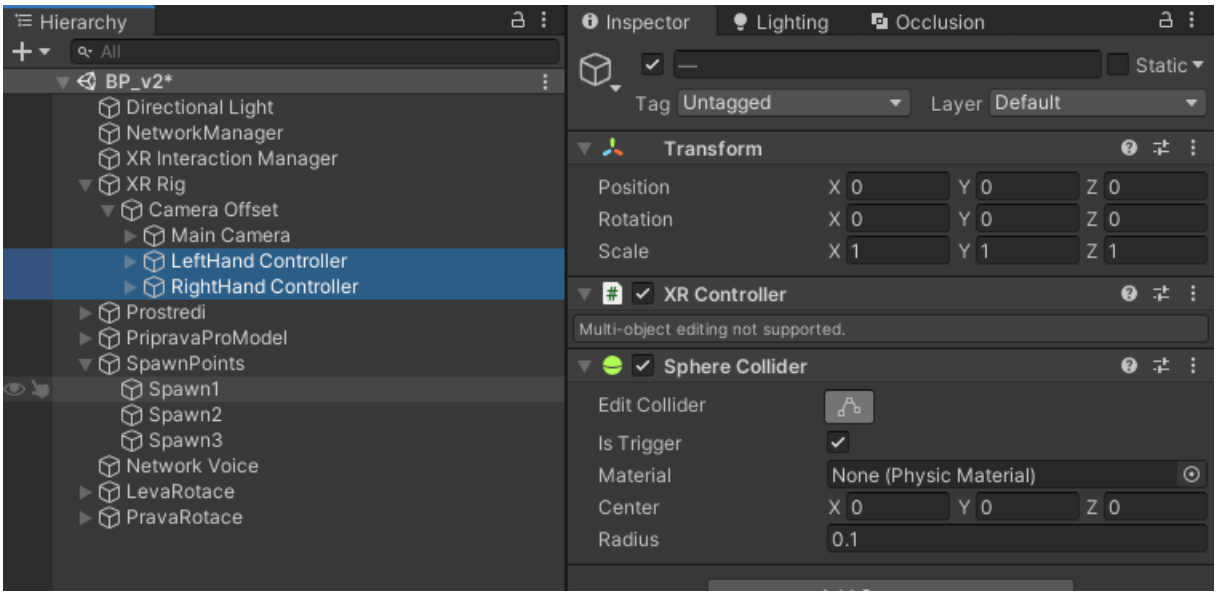
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RotaceModelu : MonoBehaviour
{
    public GameObject model;

    public float rychlostRotace;

    private void OnTriggerEnter(Collider other)
    {
        model.GetComponent<ObjectRotation>().speedRotation = rychlostRotace;

        model.GetComponent<ObjectRotation>().enabled = true;
    }

    private void OnTriggerExit(Collider other)
    {
        model.GetComponent<ObjectRotation>().enabled = false;
    }
}
```

Pro rotaci modelu přiřadíme do proměnně „model“ na obě koule objekt „PripravaProModel“. Abychom ovlivnili směr rotace, tak na objektu „LevaRotace“ nastavíme hodnotu „rychlostRotace“ na 100 a na objektu „PravaRotace“ na -100. (viz. Obr. 6-33)



Obr. 6-33 Nastavení skriptu RotaceModelu

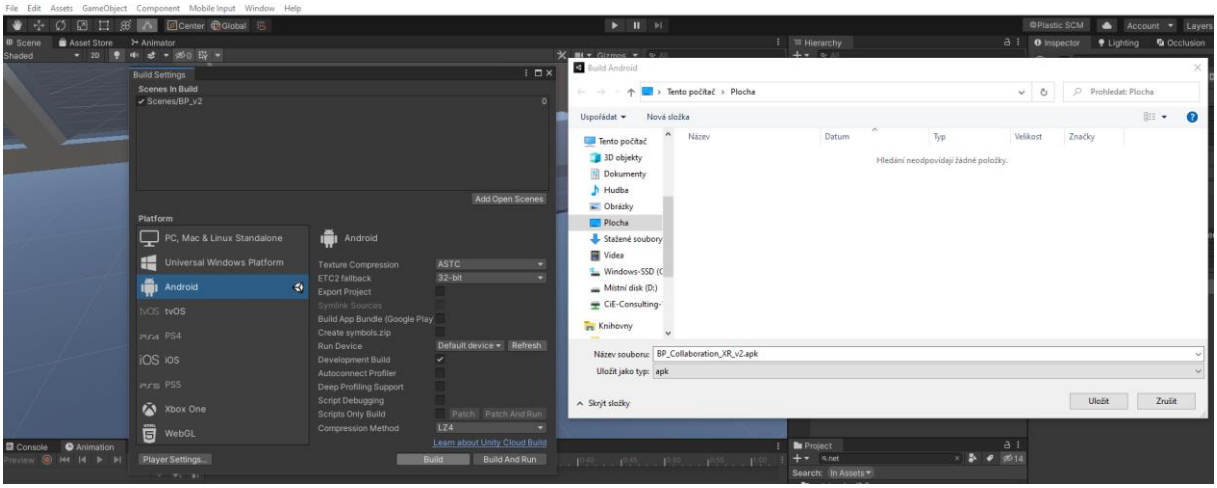
Protože ve skriptu používáme metodu „OnTrigger“, která funguje na základě vstupu jednoho „Collideru“ do druhého, tak přiřadíme oběma rukám komponentu „Sphere Collider“ s „Is Trigger“ nastavenou na „True“. Tím docílíme, po vstupu ruky do koule, k provedení akce. (viz. Obr. 6-34)



Obr. 6-34 Nastavení „Sphere Collider“ na ruce

6.4. Build aplikace Virtuální kolaborace

Tímto krokem je naše aplikace kompletní a můžeme přejít k samotnému vytvoření. Přejdeme do „File => Build Settings“. Zde přidáme pomocí tlačítka „Add Open Scenes“ naši scénu s aplikací a klikneme na tlačítko „Build“. Aplikaci uložíme do složky, o které budeme vědět, že ji jednoduše najdeme. (viz. Obr. 6-35)

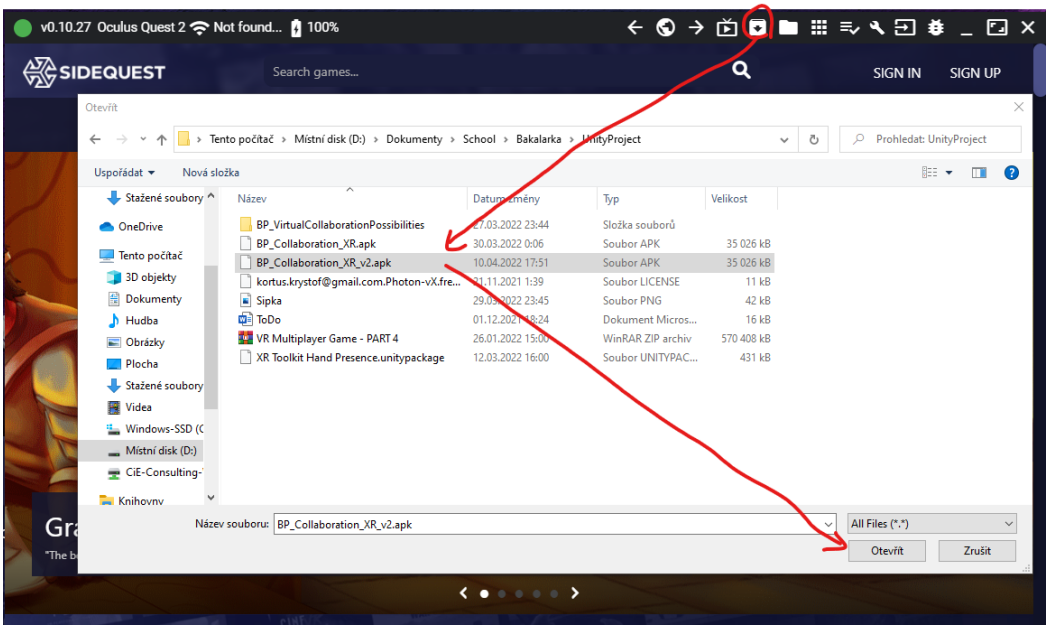


Obr. 6-35 Build aplikace

7. Instalace aplikace a její testování

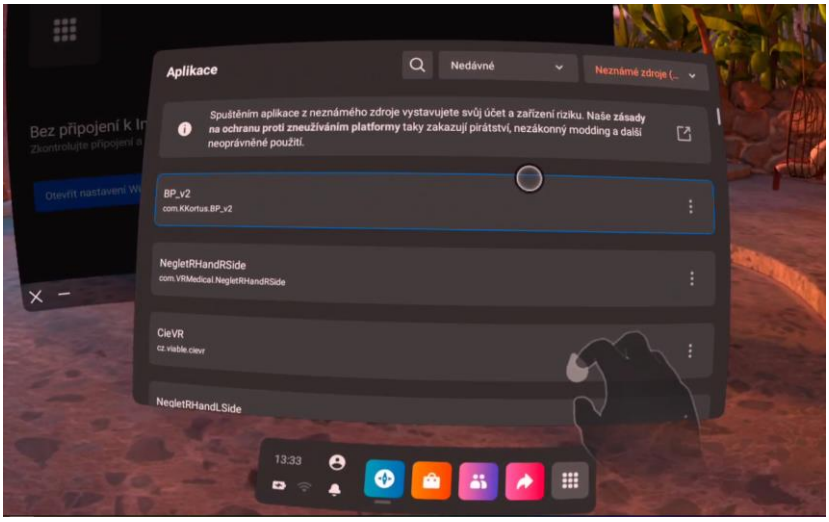
Pro instalaci aplikace do Oculus Quest 2, je zapotřebí využít aplikaci SideQuest. Tato aplikace lze zdarma stáhnout na oficiálních stránkách https://sidequestvr.com/.

Instalaci aplikace provedeme připojením brýlí k počítači a následným spuštěním aplikace SideQuest. Dále pokračujeme na tlačítko „Install APK file from folder on computer“, poté vybereme soubor .apk, který jsme vytvořili v aplikaci Unity 3D. (viz. Obr. 7-1)



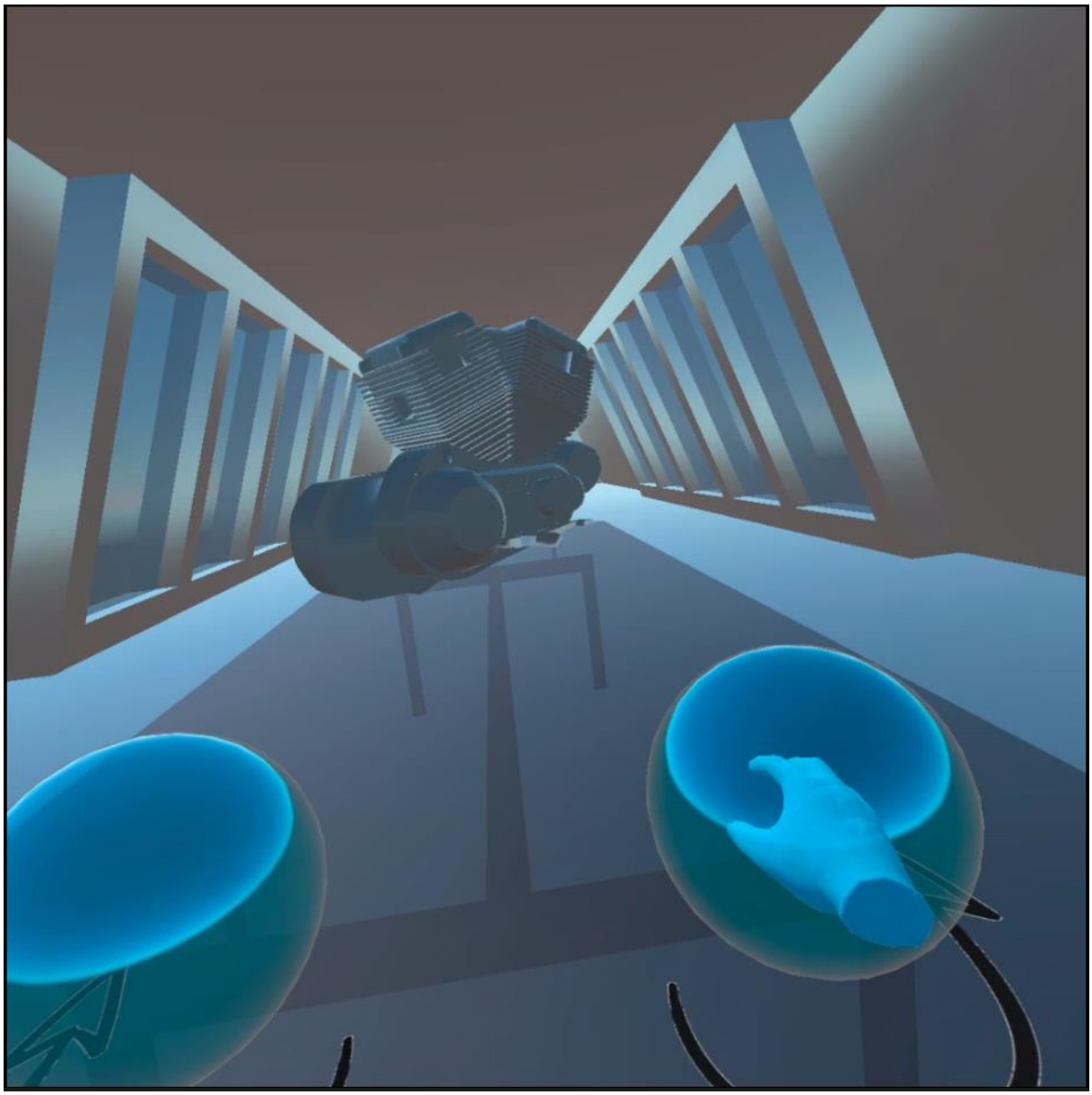
Obr. 7-1 Instalace .apk do brýlí

Aplikace je poté k nalezení v Oculus Quest 2. Pokud přejdeme do záložky s názvem „Aplikace“, následně rozbalíme horní lištu a vybereme „Neznámé zdroje“. Poklepáním na vyvinutou aplikaci bude tato spuštěna. (viz. Obr. 7-2)



Obr. 7-2 Nalezení a spuštění aplikace

Nyní se již nacházíme v naší aplikaci. Kdo aplikaci v rámci sítě spustí první, stává se tzv. vedoucím schůze. Nacházíme se v čele stolu, kde je umožněna manipulace s modelem, který se nachází uprostřed stolu. Naše pozice nám také umožnuje mít dobrý přehled o lidech, kteří se připojí do naší schůzkové místnosti. Následným připojení dalšího uživatele uvidíme ve formě nového avatara umístěného kolem stolu. Po jeho načtení je možné s ním komunikovat jak v rámci mluvení, tak v rámci gest. Pokud uživatel ukončí aplikaci, schůzková místnost zůstane zachována, ale avatar daného hráče se odstraní. (viz. Obr. 7-3)



Obr. 7-3 Ukázka aplikace

8. Zhodnocení nákladů a přínosů aplikace

Cílem práce bylo vytvoření prototypu aplikace, která dokáže zprostředkovat komunikaci napříč světem, s využitím základního modelu pro konzultaci, zejména pak v rámci průmyslového inženýrství. Nedílnou součástí práce je analýza softwarových technologií, které umožňují síťovou komunikaci. Sice již podobné aplikace na světě existují, bylo však důležité vytvořit vlastní verzi, kterou bude možné dále vyvíjet a upravovat. Aplikace je zároveň přizpůsobena pro fungování v brýlích pro virtuální realitu bez potřeby dalšího hardwaru. Dále také umožnila prozkoumat možnosti virtuální kolaborace pro budoucí vývoj a její využití pro praxi.

Aplikaci, která byla vytvořena pro virtuální kolaboraci je jednoduchá na ovládání, rychlá a funkční, je tak přínosná především pro studijní účely a pro firmy, neboť je možné importovat CAD modely do dalších verzí. Aplikace je uživatelsky nenáročná, a proto je velice efektivní pro rychlou diskuzi nad modelem.

Vytvoření aplikace zabralo kolem 120 člověkohodin. Za tuto dobu bylo vyvinuto plnohodnotné prostředí pro komunikaci ve VR. Pokud budeme brát v úvahu, že den programátora juniora v Plzni stojí 2000 a zároveň budeme počítat s náklady na marketing, a další věci s prodejem spojený, v hodnotě 20 000,- Kč, pak základní verze aplikace by tedy mohla stát kolem 20 000,- Kč + cena za virtuální brýle (pokud je firma již nevlastní). Poté by bylo zapotřebí prodat minimálně 3 licence, aby se aplikace stala zisková. Z důvodu tvorby aplikace dle potřeb zákazníka, by další moduly byly zpoplatněny na základě náročnosti práce.

Na rozdíl od aplikací, které jsou pro VR brýle zadarmo (např. VRChat), je tato aplikace zaměřená přímo na průmyslové inženýrství a je snadno přizpůsobitelná budoucím požadavkům na vývoj. Je zde jednoduchá práce s modelem a umožňuje vizuální i audiovizuální soukromou kolaboraci.

Pokud budeme brát v potaz aplikace placené nebo firmy, které jsou schopné vytvořit aplikaci na míru pro zákazníka, má tato aplikace výhodu ve své nízké finanční náročnosti. Konkrétní cenové porovnání nebylo možné z důvodu doptání firem na konkrétní zakázku.

Závěr

Bakalářská práce se zabývala možnostmi virtuální kolaborace. Za aktuální a zajímavější možnost pro trh vyšla kolaborace ve virtuální realitě. Dále byla řešena hardwarová zařízení, na kterých lze zprovoznit virtuální kolaboraci ve VR. Z hlavních výrobců virtuálních brýlí byly vybrány brýle Oculus a Pico. Následně byly zmíněny další možnosti virtuální kolaborace.

Za účelem vytvoření síťové aplikace se teoretická část zabývá technickými možnostmi síťové komunikace ve vývojovém prostředí Unity 3D, které bylo zvoleno na základě poznatků jako nejvhodnější engine pro vývoj aplikací pro VR brýle. Následně byl engine Unity 3D stručně popsán. Unity 3D má pro serverové řešení systém Unity 3D Multiplayer and Networking, který svojí masivností a tvárností dává vývojářům volnou ruku. Poté byly popsány dva typy serverů Non-Authoritative, kde server nekontroluje výsledek každého uživatelského vstupu a Authoritative Server, kde naopak server vyžaduje, aby server prováděl veškerou simulaci světa, aplikoval herní pravidla a zpracovával vstupy od hráčských klientů. Dále byly zmíněny dva typy komunikace. Remote Procedure Calls, který zajišťuje stejné zobrazení událostí z jednoho počítače na ostatní počítače v síti. Například pokud uživatel napíše do skupinové zprávy, RPC „zařídí“, aby se zobrazila zpráva všem účastníkům tohoto chatu. Druhý typ je synchronizace stavu, která se používá ke sdílení dat, která se neustále mění. Jak již bylo zmíněno, nejlepším příkladem je pozice hráče v akční hře.

Následně byl vytvořen souhrn poznatků a analýza zjištěných informací, což autorovi pomohlo pro následné vybrání nejvhodnější varianty serveru. Pro vybrání konkrétní architektury byly stanoveny body, které vycházely z předešlé teorie, preferencí zadavatele a autora a možnosti bezplatného použití. Z důvodu velké škály možností architektur, bylo vybráno na základě komunitních fór několik vybraných kandidátů, jenž byli jednotlivě a stručně popsány. Nejvhodnějším kandidátem se stal Photon Server. Vybraný server byl následně více popsán.

Praktickým výstupem bakalářské práce je aplikace, která využívá prvky virtuální reality a je kompatibilní s platformou Android pro Oculus Quest 2. Protože aplikace byla nastavena tak, aby fungovala i na nižších verzí androidu, bude fungovat i když použijeme Oculus Quest 1.

Kolaborace ve virtuální realitě slouží ke zjednodušení komunikace na dálku a odstraňuje bariéry přímého kontaktu. Aplikace by měla být vhodnou pomůckou pro veškeré firmy, které konají schůzky. Především pro firmy, které komunikují se zahraničními firmami a dávají přednost přímé komunikaci. V aplikaci je možno vidět svého kolegu v rámci avatara a diskutovat s ním nad daným modelem. S modelem je možno rotovat. Skripty aplikace byli vytvořeny tak, aby přidání jiného modelu bylo jednoduché.

Při testování aplikace bylo zjištěné, že pro virtuální brýle musí být příhodné světelné podmínky. Proto se doporučuje s brýlemi manipulovat v místnosti s přiměřeným světlem. Komunikace v aplikaci a interakce s objektem nevykázala žádné chyby. Připojení osob do místnosti pro schůzku je prakticky okamžitý po spuštění aplikace.

Aplikace bude testována více lidmi, u kterých budou sbírána data typu: připomínky na zlepšení a obecná zpětná vazba. Na základě těchto poznatků bude moct být aplikace vylepšena.

Z osobního pohledu pro budoucí zlepšení aplikace by bylo přínosné zahrnout možnost přepínání předem vybraných modelů v rámci scény. Dále přidat popisovač, díky kterému by vedoucí schůze mohl popisovat určité části zobrazeného modelu v reálném čase před zúčastněnými. Z hlediska lepšího rozlišení účastníků v aplikaci by bylo zapotřebí vytvořit více různě zbarvených avatarů. K možnosti lepšího zdokumentování výroků vedoucího schůze by mola být přidána tabule na psaní. Tím by se stala aplikace více variabilní. Zlepšením aplikace by se autor rád zabýval v rámci své budoucí diplomové práce.

Bibliografie

[1] S. M. LaValle, Virtual Reality, Cambridge University: http://lavalle.pl/vr/, 2020.

[2] M. B. L. A. &. K. K. Stephan Lukosch, „Collaboration in Augmented Reality,“ 2015. [Online]. Available: https://link.springer.com/article/10.1007/s10606-015-9239-0?_tmc=HBGJwssw7YD3Qk-juGH-mMNHfziIjRtoVBDQMh4qIuM&error=cookies_not_supported&error=cookies_not_supported&code=87fffd72-02a8-4ff5-b8f1-a37b984dff3a&code=59eef39c-b09e-4662-b922-ca8739a247b2. [Přístup získán 17 11 2021].

[3] S. Productions, „RPC (Remote Procedure Call) Definition. The Tech Terms Computer Dictionary,“ 2021. [Online]. Available: Dostupné z: https://techterms.com/definition/rpc. [Přístup získán 1 12 2021].

[4] L. A. r. reserved, „TCP vs. UDP: What’s the Difference? - Lifesize,“ 2021. [Online]. Available: https://www.lifesize.com/en/blog/tcp-vs-udp/. [Přístup získán 1 12 2021].

[5] „100× RYCHLEJŠÍ, 100× LEVNĚJŠÍ. DÍKY VIRTUÁLNÍ REALITĚ - ŠKODA Kariéra,“ Copyright © ŠKODA AUTO a.s., 2021. [Online]. Available: https://www.skoda-kariera.cz/blog/2019-04-03-100x-rychlejsi-100x-levnejsi-diky-virtualni-realite. [Přístup získán 10 11 2021].

[6] COGconnected, „How Oculus Quest 2 Changes the Game for VR... Literally.,“ Copyright © 2021 COG Connected, [Online]. Available: https://cogconnected.com/feature/how-oculus-quest-2-changes-the-game-for-vr-literally/. [Přístup získán 10 11 2021].

[7] U. V. |. T. V. R. Expert, „Oculus Quest 2 vs. Pico Neo 2: Which VR Headset is Better? - Unbound VR,“ Unbound VR, 2017. [Online]. Available: https://unboundvr.eu/oculus-quest-2-vs-pico_neo_2. [Přístup získán 10 11 2021].

[8] P. Hořejší, Habitační práce: VYUŽITÍ VIRTUÁLNÍ A ROZŠÍŘENÉ REALITY V PRŮMYSLOVÝCH PODNICÍCH, Plzeň: ZČU, 2019.

[9] GeekWire, „From AR to VR: After games, what’s the next big market for ‘extended reality?’ - GeekWire,“ Breaking News in Technology & Business, 2011. [Online]. Available: https://www.geekwire.com/2018/ar-vr-games-whats-next-big-market-extended-reality/. [Přístup získán 17 11 2021].

[10] P. C. W. P. B. J. C. A. B. H. a. O. K. GALAMBOS, *VirCA NET: A case study for collaboration in shared virtual space,* 2012.

[11] R. |. F. a. s. research, „Objects in the VirCA scene | Download Scientific Diagram,“ 2008. [Online]. Available: https://www.researchgate.net/figure/Objects-in-the-VirCA-scene_fig5_224259080. [Přístup získán 17 11 2021].

[12] 3. M. Permanently, „Teamcenter | Siemens Software,“ Siemens, 2021. [Online]. Available: https://www.plm.automation.siemens.com/global/en/products/teamcenter/. [Přístup získán 17 11 2021].

[13] G. a. R. J. Kipper, *Augmented Reality - An Emerging Technologies Guide to AR,* 2013.

[14] S. -. V. S. T. B. U. Together, „Spatial — About,“ 2021. [Online]. Available: https://spatial.io/about. [Přístup získán 17 11 2021].

[15] S. C. A. P. T. A. R. I. A. 3. W. -. VRScout, „VRScout - Virtual Reality News and VR Videos,“ 2020. [Online]. Available: https://vrscout.com/news/spatial-collaborative-ar-platform/#. [Přístup získán 17 11 2021].

[16] A. Okita, Learning C# Programming with Unity 3D, second edition, Routledge, 2019.

[17] Jead, „What platforms are supported by Unity? - Unity,“ 2021. [Online]. Available:

https://support.unity.com/hc/en-us/articles/206336795-What-platforms-are-supported-by-Unity-. [Přístup získán 20 11 2021].

[18] U. Technologies, „Unity - Manual: Multiplayer and Networking," 2021. [Online]. Available: https://docs.unity3d.com/Manual/UNet.html. [Přístup získán 20 11 2021].

[19] Collaboradev, „Visualizing 3D Network Topologies Using Unity | Collaboradev," 2014. [Online]. Available: https://collaboradev.com/2014/03/12/visualizing-3d-network-topologies-using-unity/. [Přístup získán 20 11 2021].

[20] U. Technologies, „Unity - Manual: High Level Networking Concepts (Legacy)," 2016. [Online]. Available: https://docs.unity3d.com/540/Documentation/Manual/net-HighLevelOverview.html. [Přístup získán 20 11 2021].

[21] gotoAndPlay, „SmartFoxServer: massive multiplayer game server for Flash, Unity, HTML5, iOS and Android games, MMO, virtual worlds and communities," 2004 - 2021. [Online]. Available: https://www.smartfoxserver.com/products/sfs2x#p=intro. [Přístup získán 21 11 2021].

[22] Photon, „On-Premises Cross Platform Multiplayer Game Backend | Photon Engine," 2021. [Online]. Available: https://www.photonengine.com/en/server. [Přístup získán 21 11 2021].

[23] I. GitHub, „GitHub - lidgren/lidgren-network-gen3: Lidgren Network Library," 2021. [Online]. Available: https://github.com/lidgren/lidgren-network-gen3. [Přístup získán 21 11 2021].

[24] U. Technologies, „[Released] Sockets Under Control - Unity Forum," 2021. [Online]. Available: https://forum.unity.com/threads/released-sockets-under-control.981168/. [Přístup získán 21 11 2021].

[25] gotoAndPlay, „SmartFoxServer: massive multiplayer game server for Flash, Unity, HTML5, iOS and Android games, MMO, virtual worlds and communities," 2004 - 2021. [Online]. Available: https://www.smartfoxserver.com/. [Přístup získán 1 12 2021].

[26] U. Technologies, „Powerful 2D, 3D, VR, & AR software for cross-platform development of games and mobile apps.," Unity Technologies, 2022. [Online]. Available: https://store.unity.com/#plans-individual. [Přístup získán 9 4 2022].

Přílohy

Tato kapitola označuje seznam příloh daného projektu.

Příloha 1.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;

public class NetworkManager : MonoBehaviourPunCallbacks
{
    // Start is called before the first frame update
    void Awake()
    {
        ConnectToServer();
    }

    void ConnectToServer()
    {
        PhotonNetwork.ConnectUsingSettings();
        Debug.Log("Try Connect To Server...");
    }

    public override void OnConnectedToMaster()
    {
        Debug.Log("Connected To Server.");
        base.OnConnectedToMaster();
        RoomOptions roomOptions = new RoomOptions();
        roomOptions.MaxPlayers = 10;
        roomOptions.IsVisible = true;
        roomOptions.IsOpen = true;

        PhotonNetwork.JoinOrCreateRoom("Room 1", roomOptions, TypedLobby.Default);
    }

    public override void OnJoinedRoom()
    {
        Debug.Log("Joined a Room");
        base.OnJoinedRoom();
    }

    public override void OnPlayerEnteredRoom(Player newPlayer)
    {
        Debug.Log("A new player joined the room");
        base.OnPlayerEnteredRoom(newPlayer);
    }

}
```

Příloha 2.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;

public class NetworkPlayerSpawner : MonoBehaviourPunCallbacks
{
    private GameObject spawnedPlayerPrefab;

    public override void OnJoinedRoom()
    {
        base.OnJoinedRoom();
        spawnedPlayerPrefab = PhotonNetwork.Instantiate("Network Player",
transform.position, transform.rotation);

    }

    public override void OnLeftRoom()
    {
        base.OnLeftRoom();
        PhotonNetwork.Destroy(spawnedPlayerPrefab);
    }

}
```

Příloha 3.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR;
using Photon.Pun;
using UnityEngine.XR.Interaction.Toolkit;
using Photon.Realtime;
using System.IO;

public class NetworkPlayer : MonoBehaviour
{
    public Transform head;
    public Transform leftHand;
    public Transform rightHand;
    public Animator leftHandAnimator;
    public Animator rightHandAnimator;
    private PhotonView photonView;
    private Transform headRig;
    private Transform leftHandRig;
    private Transform rightHandRig;

    //Spawn Players
    private GameObject myPlayerAvatar;

    private Player[] allPlayers;
    private int myNumberInRoom;

    void Start()
    {

        photonView = GetComponent<PhotonView>();
        myPlayerAvatar = GameObject.Find("XR Rig");

        XRRig rig = FindObjectOfType<XRRig>();
        headRig = rig.transform.Find("Camera Offset/Main Camera");
        leftHandRig = rig.transform.Find("Camera Offset/LeftHand Controller");
        rightHandRig = rig.transform.Find("Camera Offset/RightHand Controller");

        //Spawn Players
        allPlayers = PhotonNetwork.PlayerList;
        foreach (Player p in allPlayers)
        {
            if (p != PhotonNetwork.LocalPlayer)
            {
                myNumberInRoom++;
            }
        }

        if(photonView.IsMine)
        {
            foreach (var item in GetComponentsInChildren<Renderer>())
            {
                item.enabled = false;
            }

            photonView.RPC("SpawnPointCharacter", RpcTarget.AllBuffered,
myNumberInRoom);


        }
```

```
    }

    [PunRPC]
    public void SpawnPointCharacter(int point)
    {
        this.transform.parent = AU_GameController.instance.spawnPoints[point];
        this.transform.localPosition = new Vector3(0, 0, 0);
        this.transform.localEulerAngles = new Vector3(0, 0, 0);
        myPlayerAvatar.transform.parent =
AU_GameController.instance.spawnPoints[point];
        myPlayerAvatar.transform.localPosition = new Vector3(0, 0, 0);
        myPlayerAvatar.transform.localEulerAngles = new Vector3(0, 0, 0);
    }

    void Update()
    {
        if(photonView.IsMine)
        {
            MapPosition(head, headRig);
            MapPosition(leftHand, leftHandRig);
            MapPosition(rightHand, rightHandRig);

            UpdateHandAnimation(InputDevices.GetDeviceAtXRNode(XRNode.LeftHand),
leftHandAnimator);
            UpdateHandAnimation(InputDevices.GetDeviceAtXRNode(XRNode.RightHand),
rightHandAnimator);
        }

    }

    void UpdateHandAnimation(InputDevice targetDevice, Animator handAnimator)
    {
        if (!handAnimator)
            return;

        if (targetDevice.TryGetFeatureValue(CommonUsages.trigger, out float
triggerValue))
        {
            handAnimator.SetFloat("Trigger", triggerValue);
        }
        else
        {
            handAnimator.SetFloat("Trigger", 0);
        }

        if (targetDevice.TryGetFeatureValue(CommonUsages.grip, out float gripValue))
        {
            handAnimator.SetFloat("Grip", gripValue);
        }
        else
        {
            handAnimator.SetFloat("Grip", 0);
        }
    }

    void MapPosition(Transform target, Transform rigTransform)
    {
        target.position = rigTransform.position;
        target.rotation = rigTransform.rotation;
    }
}
```