Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Terapeutický systém na báze technológií eXtended (XR) reality

Diplomová práca

2025 **Bc. Adam Mikael**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Terapeutický systém na báze technológií eXtended (XR) reality

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	doc. Ing. Branislav Sobota, PhD.

Košice 2025 **Bc. Adam Mikael**

Abstrakt v SJ

Cieľom diplomovej práce je vylepšenie aplikácie, ktorá poskytuje neurorehabilitáciu vo virtuálnom prostredí. Používateľ sa vo virtuálnom priestore môže pripojiť ako terapeut alebo pacient a následne sa s ďalším pripojeným používateľom na serveri venovať rehabilitácii horných končatín. Vylepšenie spočíva v pridaní ovládania pomocou rúk (hand tracking) a implementácii interakcie pre dané ovládanie, ktoré zabezpečuje kombinácia interaktorov a interakčných prvkov (interactables). Práca obsahuje implementáciu dvoch odlišných XR rigov, ktoré sú kompatibilné s hand trackingom. Následne je implementované riešenie pre správny networking, aby každý používateľ videl správne správanie rúk u každého používateľa na serveri. V projekte je implementované aj vylepšenie scalera, ktoré poskytuje používateľovi možnosť detailného nastavenia veľkosti celého avatara, ako aj iba jeho rúk a dlaní.

Kľúčové slová v SJ

virtuálna realita, objektovo orientované programovanie, neurorehabilitácia, terapia, ovládanie pomocou rúk, synchronizácia, precizácia

Abstrakt v AJ

The aim of the thesis is to enchance an application that provides neurorehabilitation in a virtual environment. User can connect to the virtual space as a therapist or patient and then works with another connected user on the server to rehabilitate the upper limbs. The improvement consists of adding hand control (hand tracking) and implementing interaction for the given control, which is provided by a combination of interactors and interactable objects (interactables). Thesis includes the implementation of two different XR rigs that are compatible with hand tracking. Subsequently, a solution for proper networking is implemented so that each user can see the correct behavior of the hands of each user on the server. The thesis also includes implementation of enchanted scaler, which provides the user with the ability to adjust the size of the entire avatar in detail, as well as only his hands and palms.

Kľúčové slová v AJ

virtual reality, object-oriented programming, neurorehabilitation, therapy, hand tracking control, synchronization, precision

Bibliografická citácia

MIKAEL, Adam. *Terapeutický systém na báze technológií eXtended (XR) reality*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2025. 90s. Vedúci práce: doc. Ing. Branislav Sobota, PhD.

75716

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Terapeutický systém na báze technológií eXtended (XR) reality

Therapeutic System Based on eXtended (XR) Reality Technologies

Študent: **Bc. Adam Mikael**
Školiteľ: **doc. Ing. Branislav Sobota, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:
1. Naštudovať problematiku virtuálnej reality a eXtended reality (XR) platformy a jej technológií a základnú problematiku rehabilitačných procesov.
2. Analyzovať problematiku systémov XR v prostredí rehabilitácií rôzneho typu s dôrazom na horné končatiny s previazaním na možné kognitívne poruchy.
3. Navrhnúť terapeutický systém na báze XR platformy s focusáciou na virtuálnu realitu a jej technológií vrátane časti systému terapeuta a časti systému operátora aj pacienta, u ktorého sa terapia/rehabilitácia vykonáva.
4. Pri návrhu sa sústrediť na základe pokynov vedúceho práce najmä na podporu koordinácie pohybu horných končatín, podporu precizácie, výberu a umiestnenia vrátane scenárov terapeutických pohybov.
5. Na základe bodov 2-4 implementovať programové vybavenie na báze technológií XR platformy.
6. Vykonať experimentálne práce s implementovaným systémom vrátane spracovania alebo dotvorenia potrebných modelov resp. funkcií a zhodnotiť dosiahnuté výsledky.
7. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 17.04.2025
Dátum zadania diplomovej práce: 31.10.2024

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -10-

n.z. P. Vokorokos

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

Košice, 17.4.2025 .....

Vlastnoručný podpis

Poďakovanie

Rád by som poďakoval svojmu vedúcemu práce doc. Ing. Branislavovi Sobotovi, PhD. a odbornému asistentovi a konzultantovi Ing. Štefanovi Korečkovi, PhD. za ich čas a odborné vedenie počas riešenia mojej záverečnej práce.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam zdrojových kódov

Úvod

V dnešnej neustále uponáhľanej dobe sa čoraz viac stretávame s veľkým nárastom psychických porúch alebo fyzických obmedzení u ľudí. Technológie však oproti minulosti pokročili. Z pomalších zariadení s nižším výkonom sa postupom času vyvinuli nové a modernejšie, ktoré zabezpečujú oveľa vyšší výkon a kvalitu výpočtovej techniky. Tak sa otvárajú nové možnosti liečby týchto problémov.

Tieto možnosti zahŕňajú aj využitie virtuálnej a eXtended reality. Virtuálna realita poskytuje virtuálne prostredie, ktoré sa vo veľkej miere dá využiť, či už simuláciou virtuálneho priestoru alebo prvkov, ktoré vieme pri liečbe virtuálne používať bez toho, aby sme ich v realite mali poskytnuté. Rozšírením tejto reality je eXtended realita, ktorá virtuálnu rozširuje o ďalšie typy ako je mixed reality a augmented reality. Táto kombinácia je rovnako účinná na riešenie problémov modernej doby. Pri takejto kombinácii môžu byť pacientovi poskytnuté liečebné pomôcky, ktoré sú zakomponované do reality, ktorá je do VR headestu prepúšťaná pomocou kamier, ktoré sú v modernejších headsetoch aj s farebným obrazom.

Rovnako sa pri modernejších headsetoch čoraz viac zlepšuje a využíva technológia hand tracking. Táto technológia umožňuje namiesto ovládačov headsetu používať ruky. Spracovanie hand trackingu je v súčasnosti celkom precízne, no do budúcna bude táto technológia určite vylepšená ešte precíznejšie a efektívnejšie.

Dnešná doba ponúka množstvo projektov s implementáciou XR. Napríklad existujú projekty, ktoré sa venujú simuláciám. Ďalšie projekty sa venujú tvorbe virtuálnych modelov. Existujú aj projekty, ktoré snímajú pacientov a podľa toho dokážu odhaliť skoré štádium Alzheimerovej choroby. Množstvo projektov používa aj špeciálne vytvorený hardware, ktorý rozširuje softvér o ďalšie interakcie.

Táto diplomová práca sa týka rozšírenia systému pomenovanom "Dôveryhodná interakcia človek–robot a terapeut–pacient vo virtuálnej realite". Systém je vyvíjaný v rámci projektu APVV-21-0105. Prostredie je implementované v Unity a poskytuje vzájomnú interakciu používateľov, ktorá sa vykonáva v reálnom čase.

Motivácia

Vylepšenie terapeutického systému, ktorý poskytuje pacientom podporu koordinácie horných končatín a precizácie. Pridanie hand trackingu sa pacientovi pomôže sústrediť na ruky bez potreby držania ovládačov headsetu. Pacient sa bude cítiť oveľa komfortnejšie a tak bude pre neho liečba efektívnejšia. Zabezpečenie správneho networkingu pre virtuálne ruky zabezpečí všetkým pripojeným používateľom vidieť precízne polohy a rotácie rúk aj prstov navzájom. To sa odzrkadlí na výsledkoch terapie.

Pridanie a úprava už vytvorených objektov tak, aby sa nimi dalo manipulovať aj rukami bez potreby používania ovládačov je pri tomto riešení veľmi vhodná. To zabezpečí úplné využitie funkcionality ovládania pomocou handtrackingu.

Čím viac logických inovácií bude tomuto systému implementovaných, tým lepšie používateľov systém osloví. Tieto inovácie sa týkajú napríklad aj úpravy scalera. Konkrétne, možnosť prispôsobiť veľkosť celého avatara alebo jeho častí podľa požiadaviek používateľa umožní lepšie zosúladenie používateľa a avatara.

Všetky tieto riešenia modernizujú projekt. Takto sa metóda liečby vo virtuálnej realite dostane viac do povedomia a stane sa pre veľa ľudí prístupnejšia, kompatibilnejšia a zaujímavejšia.

Ciele

Hlavná časť pozostáva z návrhu a implementácie hand trackingu. Systém používa ovládanie pomocou ovládačov. Tieto sú nakonfigurované na rôzne úkony používateľa ako je napríklad úchop objektu alebo pohyb po mape. Implementáciou hand trackingu tak interakcia v systéme bude oveľa reálnejšia.

V ďalšej časti je potrebné pridať konkrétne mechanizmy interakcie. Týmito sú rôzne interaktory a interactables, ktoré ponúka mnoho rôznych balíkov. Pri synchronizácii týchto prvkov a prepojení s hand trackingom sa dosiahne plná interakcia pomocou hand trackingu.

Potrebná je aj úprava virtuálneho avatara. Pri prepínaní medzi ovládačmi a rukami je potrebné, aby sa aj telo a ruky avatara pohybovali, správne menili a vždy boli synchronizované s rukami používateľa.

Úprava scalera je taktiež vhodná. Pridanie možností zväčšiť alebo zmenšiť mesh avatara, rúk alebo dlaní je pre používateľa prínosom pre lepšie zosúladenie.

Formulácia úlohy

V hlavnej časti je potrebné navrhnúť a implementovať hand tracking. Novšie verzie headsetov pre virtuálnu realitu, ako je napríklad Meta Quest 2 a 3 poskytujú zabudovaný hand tracking. Táto funkcionalita umožňuje používateľom ovládať headset pomocou rúk a používať gestá namiseto ovládačov. Systém je implementovaný v prostredí Unity a to je kompatibilné s množstvom balíkov zabezpečujúcich fungovanie a konfiguráciu hand trackingu, ktoré je možné vložiť do projektu.

Dvoma z týchto balíkov sú XR Hands a XR Interaction Toolkit, pomocou ktorých je vylepšenie dosiahnuteľné. Interakciu pomocou interaktorov a interactables zabezpečí taktiež balík XR Interaction Toolkit. Pri riešení hand trackingu je pre XR Interaction Toolkit potrebná verzia minimálne 2.3.0. Pre úpravu synchronizácie virtuálneho avatara a používateľa je hlavný balík Final IK. Ten využíva inverznú kinematiku, ktorá je základom pre synchronizáciu končatín. Pri tejto synchronizácii a prepínaní je potrebné sústrediť sa na komponenty Final IK balíka, ako je napríklad VR IK, ktorý je vložený na objekt avatara. S týmto komponentom je potrebné prepojiť logiku prechodu medzi ovládačmi a rukami. Základom je logika samotného prepínania medzi ovládačmi a rukami nezávisle od avatara. K celej funcionalite je potrebné vytvorenie ďalších potrebných skriptov, ktoré zabezpečia plynulé a efektívne používanie.

Pre synchronizáciu a správne zobrazenie na serveri aj pre ostatných používateľov je potrebné využiť balík Mirror, ktorý je v projekte už vložený. Základ je implementovať riešenie tak, aby dopĺňalo a nadväzovalo na vyriešený Mirror networking pre avatara bez hand trackingu. V projekte je použitá verzia 72.0.0.

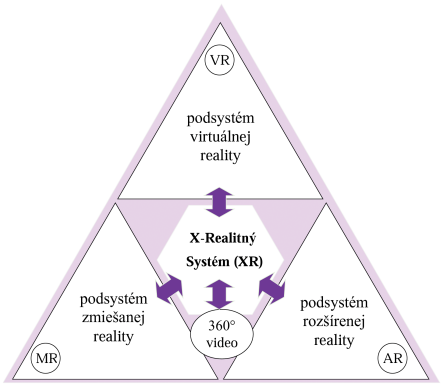
Pri riešení úpravy scalera je potrebné pridať možnosti pre zväčšenie a zmenšenie avatara alebo jeho častí. To je možné zabezpečiť úpravou skriptov, ktoré sú vytvorené pre avatara a riešia túto problematiku. K tejto funcionalite je potrebné pridať aj UI prvky, s ktorými bude mať používateľ možnosť pracovať a nakonfigurovať avatara podľa svojich predstáv.

1 Systémy Virtuálnej Reality

Virtuálno-realitný systém predstavuje interaktívny počítačový systém. Ten vytvára ilúziu priestoru, ktorý nie je reálny ale simulovaný. Toto spojenie nazývame spojenie človek-výpočtový systém. V dnešnej dobe do virtuálnej reality bežný používateľ vstupuje hlavne pomocou VR headsetu. Avšak, to je len jedna z možností. V tejto práci však používame túto možnosť.

1.1 Typy podsystémov

Pri práci vo virtuálnom priestore je potrebné uvedomiť si aký typ reality implementujeme. Základom tejto práce je virtuálna realita. VR je prvkom eXtended reality, ktorá aplikačne zjednocuje 4 podsystémy, ako je vidieť na obrázku 1.1.



Obr. 1.1: X-Realitný Systém (XR)

1.1.1 VR - podsystém virtuálnej reality

Umožňuje používateľovi interagovať s počítačovo vytvoreným prostredím v reálnom čase. Používateľ si nasadí špeciálne HMD (Head Mounted Display) zariadenie. To mu umožňuje vidieť, počuť a interagovať s prostredím virtuálneho sveta a izoluje ho od reálneho sveta.

1.1.2 AR - podsystém rozšírenej reality

Základom je pridanie digitálneho obsahu do reálneho sveta. Používateľ môže vidieť reálny svet a ten je doplnený digitálnymi prvkami, s ktorými môže interagovať. Túto realitu používajú napríklad firmy na prezentáciu svojich produktov.

1.1.3 MR - podsystém zmiešanej reality

Kombinácia virtuálneho a reálneho sveta. Oproti rozšírenej realite sú však prvky zmiešanej reality viac integrované do reálneho prostredia. To vyvoláva pocit, že prvky existujú v reálnom prostredí a komunikujú s ním. Používatelia môžu interagovať s týmito prvkami.

1.2 XR projekty

Existuje množstvo firiem, ktoré pracujú s eXtended realitou. Tieto firmy často vytvárajú aj vlastný hardware, ktorý je s ich XR softvérom najkompatibilnejší. Medzi tieto projekty patria napríklad Pokémon Go, Apple Vision Pro, Google Cardboard. Medzi najznámejšie XR projekty patria:

1.2.1 Meta Quest

Tento projekt je vytvorený firmou Meta Platforms Technologies. V skratke tento headset predstavuje základnú predstavu virtuálnej reality bežného používateľa. Meta Quest headseaty sú udržiavaným projektom a tak stále vznikajú nové verzie tohto headsetu. Momentálne je ako najnovší headset na trhu Meta Quest 3S. S týmito headsetmi používateľ dokáže tvoriť a objavovať rôzne Meta aplikácie. Pri implementácii a testovaní tohto riešenia bol primárne používaný headset Meta Quest 3 (obrázok 1.2).

Čo sa týka technológie passtrhrough, Meta Quest 2 dokáže prepustiť obraz reálneho sveta čiernobielo. Avšak Meta Quest 3 už obraz prepúšťa farebne. Ovládanie pracuje buď pomocou meta ovládačov alebo pomocou technológie hand tracking, ktorá sníma ruky používateľa a premieta ich do headsetu. Tieto a mnoho ďalších informácií sú dohľadateľné na oficiálnej webstránke Meta [1], ktorá ponúka vždy najaktuálnejšie technológie tejto spoločnosti.

Obr. 1.2: Meta Quest 3 headset[2]

1.2.2 Microsoft HoloLens 2

Projekt je vytvorený firmou Microsoft. Hololens 2 okuliare sú vyvíjané na princípe technológie mixed reality(MR). Headset obsahuje displej HoloLens, ktorý dokáže v reálnom svete zobrazovať hologramy a pomocou senzorov, ktoré okuliare obsahujú s nimi používateľ dokáže interagovať. Tento headset je často doplnkom pre firmy, ktoré chcú napredovať v technologickom pokroku. Na oficiálnej webovej stránke Microsoftu [3] je dostupný list konfigurácií headsetu.

Obr. 1.3: Pracovný meeting s využitím headsetu Microsoft Hololens 2[4]

1.3 XR terapeutické aplikácie

Takýchto aplikácii je na trhu mnoho a každá je niečim výnimočná. Liečba vo virtuálnom prostredí môže pôsobiť ako v reálnom prostredí a to pomáha pacientom vyliečiť sa rýchlejšie. Medzi najrozšírenejšie typy rehabilitačných aplikácií patria kategórie: Rehabilitačné aplikácie, Virtual Reality Exposure Therapy (VRET) a VR Pain Management.

Rehabilitačné aplikácie

Aplikácie navrhnuté na zlepšenie a podporu kognitívnych funkcií ako je pamäť, plánovanie a reakčná rýchlosť. Tieto aplikácie sú prospešné hlavne pre pacientov s poraneniami mozgu, demenciou alebo inými neurokognitívnymi poruchami.

Virtual Reality Exposure Therapy (VRET)

Tento typ terapie simuluje prostredie, ktoré u pacienta vyvoláva úzkosť alebo fóbiu. Pacienti sú vystavený tímto prostrediam. Takýto spôsob pacientom pomáha vyliečiť sa z ich úzkostí a fóbií.

VR Pain Management

Tieto terapie simulujú prostredia, ktoré u pacienta vyvolávajú relax a uvoľnenie. Pri týchto aplikáciach pacient zabúda na bolesť, ktorú cíti. Pacient uvoľní napätie a nastáva efektívnejšia liečba.

1.3.1 Príklady XR terapeutických aplikácií

Aplikácie v tejto sekcii rovnako ako tento projekt pomáhajú ľuďom pomocou liečby vo virtuálnej realite. OxfordVR sa používa hlavne na liečbu psychóz a fóbií a Osso VR je zameraná na zdokonaľovanie chirurgických skúseností.

OxfordVR

Tento projekt je inovatívny startup, ktorý v roku 2016 implementovala Univerzita Oxford. Je vyvinutý na kognitívne terapie vo virtuálnej realite. Obsahuje prostredia, ktoré simulujú psychickú pohodu. Venujú sa hlavne psychózam a akrofóbii (strachu z výšok). Ako uvádza oficiálna webstránka OxfordVR [5], softvér využíva interaktívneho trénera, s ktorým pacienti môžu komunikovať a ten im následne radí. Komunikácia a uvoľnenie sa je v OxfordVR veľmi dôležité. Pomáha pacientom liečiť svoje psychózy, akrofóbie a tým im aj predchádzať. Záber z prostredia tejto aplikácie je zobrazený na obrázku 1.4.

Obr. 1.4: Prostredie OxfordVR

Osso VR

Táto platforma sa zameriava na zdokonaľovanie lekárov v chirurgických skústenostiach. Ich partnerské štúdie dokazujú, že vážne pomáhajú zlepšovať skúsenosti používateľov. Obsahuje simuláciu, v ktorej je vykonávaná operácia. Poskytuje operácie rôzneho typu od najľahších až po najzložitejšie. Doktori sa pripoja do miestnosti, v ktorej majú poskytnuté prostredie, ktoré simuluje operačnú sálu. Následne začína priebeh operácie. Doktori vykonávajú úlohy, ktoré sú navrhnuté na vylepšovanie ich skúseností, ktoré využívajú v realite. Všetky tieto informácie sú poskytnuté na ich webovej stránke [6], na ktorú stále pridávajú najaktuálnejšie informácie. Záber z prostredia tejto aplikácie je zobrazený na obrázku 1.5.

Obr. 1.5: Prostredie OssoVR

1.4 Softvér na tvorbu 3D projektov

Existuje mnoho softvérov, v ktorých je možné vytvárať 3D projekty. Takéto softvéry ponúkajú spoločné základy no každý má svoje výhody a nevýhody. Pri voľbe takéhoto softvéru treba zvážiť parametre, ktoré sú základom pre vývoj. O týchto parametroch poskytuje detailný popis článok s názvom „An Overview for Parameters of Game Engine" [7].

Prvým parametrom je typ projektu. Tu je potrebné zvážiť, čo implementujeme. Napríklad hru, ktorá bude obsahovať mnoho efektov a bude distribuovaná pre hernú komunitu alebo projektom je grafika, ktorá bude vyrenderovaná vo forme sekvencie obrázkov, alebo rovno ako video. Takýchto typov projektov je mnoho a každý 3D softvér je pre nejaký typ výhodnejší a pre iný zas nevýhodnejší.

Ďalším parametrom je cena. Veľa licencií je zadarmo, no pri profesionálnych softvéroch zvyknú byť licencie platené. Pri tomto parametri je dôležité si uvedomiť, čo je od projektu očakávané. Je potrebné zvážiť či bezplatné licencie ponúkajú všetko potrebné alebo, či sú platené licencie vážne potrebné a pri spoplatnených projektoch nevytvoria straty.

Netreba zabúdať ani na to, že vo veľa softvéroch sa pracuje rozlišne. Či už je to ovládanie prostredia alebo programovací jazyk, ktorý je potrebný pre profesionálnu prácu na projektoch.

Plug-iny sú ďalšou dôležitou súčasťou. Či už softvéry poskytujú vlastné plug-iny alebo plug-iny tvorené komunitou, všetky poskytujú rozšírenia v podobe či už vylepšení projektu, alebo zjednodušeniu implementácie. Každý softvér zvykne obsahovať špecifické plug-iny, no nájdu sa aj plug-iny, ktoré sú kompatibilné s viacerými softvérmi.

V tejto práci je použitý herný engine Unity, ktoré sa radí medzi 3 najznámejšie na svete. Pomenovanie herný engine však neznamená, že sú v takomto softvéri tvorené len hry, pretože v týchto enginoch sú často tvorené interaktívne aplikácie, simulácie a systémy, ktoré nie sú priamo zamerané na herný obsah.

ENGINE	LANGUAGES	PLATFORMS	COST	EASE	VISITS
Unity 2005 // 2D + 3D //	C#	mobile, desktop, console, browser, vr, ar	Royalty ($1M+)	2	5.8m
Unreal Engine 1998 // 2D + 3D //	C++, Blueprint	mobile, desktop, console, browser, vr, ar	Royalty ($1M+)	3	3.8m
Godot 2014 // 2D + 3D //	GDScript, C#, C++, VisualScript	mobile, desktop, console, browser	Free	3	1.1m

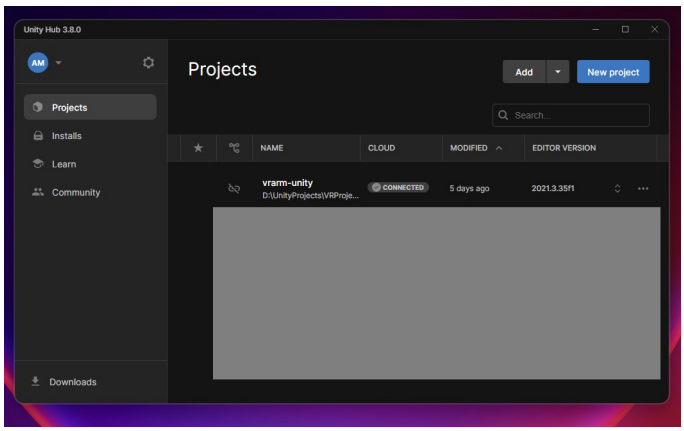
Obr. 1.6: Tri aktuálne najznámejšie herné enginy[8]

1.5 Unity

Známe aj ako Unity3D, patrí Unity v dnešnej dobe k najrozšírenejším vývojovým prostrediam pre tvorbu hier, grafík a všeobecne napríklad 2D alebo 3D projektov. Systém, ktorý je vyvíjaný v tejto práci je implementovaný pomocou Unity Engine. Dokumentáciu Unity a príslušných balíkov je možné nájsť na stránke [9].

1.5.1 Unity Hub

Pomocou tohto nástroja môže vývojár spravovať svoje inštalácie verzií Unity, projekty a licencie. Poskytuje aj možnosť upravovať aké moduly sú obsiahnuté v týchto inštaláciách. Tento nástroj je bezplatne dostupný na webstránke Unity.

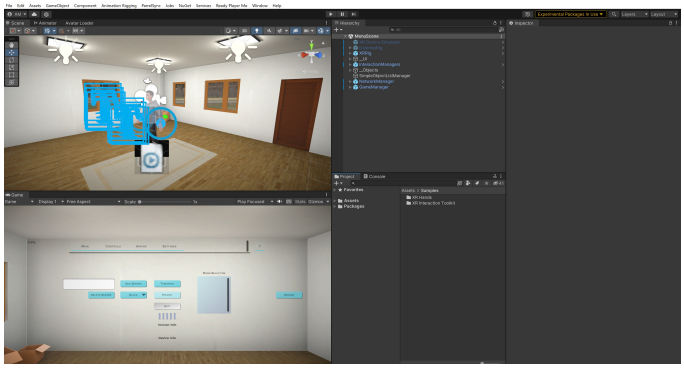


Obr. 1.7: Rozhranie Unity Hub

1.5.2 Unity Editor

Toto hlavné vývojové prostredie obsahuje mnoho okien, ktoré môže vývojar vo svojom projekte používať. Unity Editor poskytuje gridy pre tieto okná, no vývojár si vždy môže rozloženie meniť podľa svojích potrieb. Zoznam základných okien:

- Scene - Náhľad scény s možnosťou úpravy scény.
- Game - Náhľad scény pri spustení.
- Hierarchy - Hierarchia objektov nachádzajúcich sa v scéne.
- Project - Hierarchia a priečinky celého projektu.
- Console - Obsahuje výstupné správy, upozornenia a chybové hlásenia.
- Inspector - Zobrazuje a umožňuje upravovať objekty a ich vlastnosti.



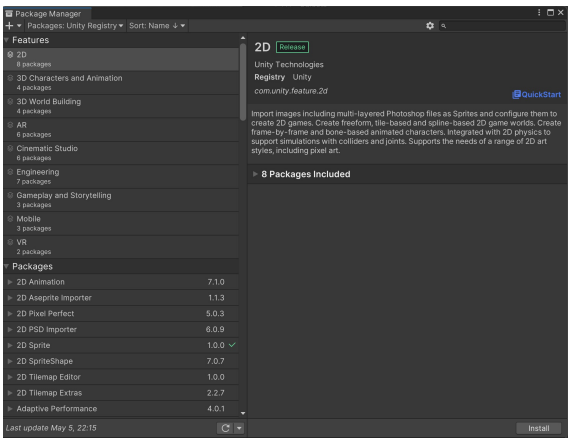
Obr. 1.8: Projekt otvorený v Unity Editore

1.5.3 Asset Store

Ako je uvedené v článku s názvom „A History of the Unity Game Engine" [10], Asset Store je trh tretích strán, vytvorený 10. Novembra 2010. Tento trh je možné nájsť na webe a je aj integrovaný v IDE pod cestou window -> asset store. Poskytuje balíky vytvorené vývojarmi, ktoré môžu byť zakúpené a vložené do projektov. Z predaja si Unity Technologies účtuje 30% a vývojárovi ostáva 70%. Balíky obsahujú napríklad modely, skripty a rôzne komponenty, ktoré pomáhajú pri práci na projekte.

1.5.4 Package Manager

Okno, v ktorom je možné spravovať svoje balíky. Package Manager je dostupný pomocou cesty window -> package manager. Poskytuje inštaláciu balíkov a ich súčastí vložených do projektu. Tieto balíky môžu byť získané z Asset Storu alebo z iných zdrojov.

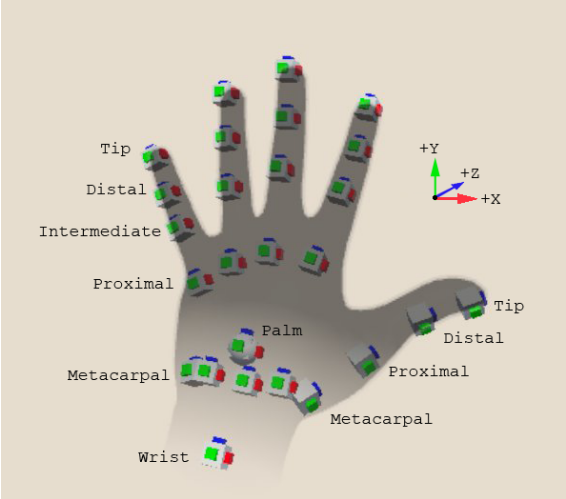


Obr. 1.9: Správca balíkov

1.5.5 XR Hands

Informácie o analýze balíka sa týkajú verzie 1.5.0. Balík je súčasťou rozšírenia XR v Unity, ktoré je navrhnuté na podporu vývoja aplikácii pre virtuálnu a rozšírenú realitu. Poskytuje nástroje a API pre sledovanie a interakciu pomocou rúk. Pomocou týchto nástrojov je umožnený prístup k údajom zo sledovania rúk. Tieto údaje sú poskytnuté zariadeniami podporujúcimi hand tracking. Balík obsahuje aj komponenty pre vizualizáciu virtuálnych rúk.

Sledovanie rúk používateľa v reálnom čase poskytuje informácie o polohe, rotácií, pohyboch a tvare prstov, dlaní a zápästí. Tento balíček je integrovaný s Unity Input System. To umožňuje efektívne mapovanie vstupov hand trackingu a prácu s týmito vstupmi.



Obr. 1.10: Dátový model rúk XR Hands 1.5.0 [11]

Preklad označení z obrázka 1.10:

- Tip - Koniec prsta.
- Distal - Distálna časť prsta.
- Intermediate - Stredná časť prsta.
- Proximal - Proximálna časť prsta.
- Metacarpal - Časť ruky medzi zápästím a prstami.
- Palm - Dlaň.
- Wrist - Zápästie.

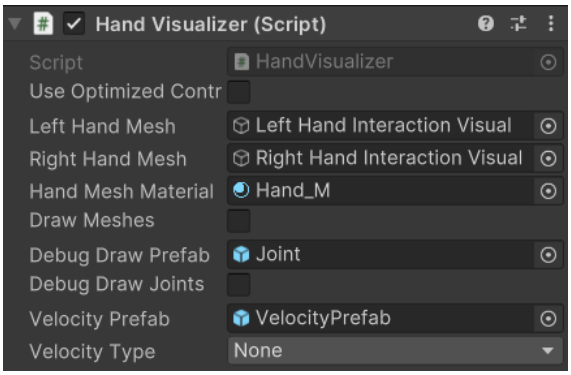
Hand Data Model

Hand tracking pomocou balíka XR hands poskytuje model (obrázok 1.10), ktorý poskytne údaje o rukách, ktoré sú sledované pomocou kamier headsetu. Tento model poskytuje možnosť získavať údaje o konkrétnych kostiach ruky. V tomto balíku sa ako kosti berú všetky body, ktoré dokážeme pri hand trackingu získať. Tieto kosti dokážeme zároveň vykresliť. Údaje obsahujú:

- Informácie o polohe rúk/kostí - Každá kosť ma svoju polohu v scéne a tá je poskytovaná vždy v reálnom čase.
- Informácie o orientácii rúk/kostí - Rovnako ako poloha je v reálnom čase poskytovaná aj orientácia kostí.
- Informácie o zmenách polohových vektorov bodov ruky - Znázorňujú rýchlosť a smer kostí v metroch za sekundu a rýchlosť otáčania kostí ako uhlovú rýchlosť v metroch za minútu.

Hand Visualizer

Hand tracking model poskytuje aj vizualizáciu meshu rúk. Vizualizácia je zabezpečená pomocou Hand Visualizera. Je to objekt v scéne, ktorý obsahuje komponent Hand Visualizer, ktorý zabezpečuje konfiguráciu a vykresľovanie meshu rúk a kostí.



Obr. 1.11: Komponent vizualizátora rúk

- Script - Vykonáva funkcionalitu komponentu HandVisualizer.
- Use Optimized Controls - Nastavenie, či bude zapnutý System internal feature flag pre optimalizovanie vstupov.
- Left Hand Mesh - Mesh, ktorú vykresli Hand Visualizer ako ľavú ruku.

- Right Hand Mesh - Mesh, ktorú vykresli Hand Visualizer ako pravú ruku.
- Hand Mesh Material - Materiál, ktorý je priradený pre mesh rúk.
- Draw Meshes - Nastavenie, či sa majú vykresľovať meshe rúk.
- Debug Draw Prefab - Prefab, ktorý sa používa na vizualizáciu debugovania jednotlivých kostí ruky.
- Debug Draw Joints - Nastavenie, či sa majú zobrazovať debug body.
- Velocity Prefab - Prefab, ktorý sa používa na vizualizáciu smeru, rýchlosti a otáčania bodov.
- Velocity Type - Typ vizualizácie (Linear - priama rýchlosť a smer, Angular - uhlová rýchlosť).

1.5.6 XR Interaction Toolkit

Hlavnou úlohou tohto balíka je interakcia s virtuálnymi objektmi. Tieto interakcie sú napríklad uchopenie, otáčanie, posun a uvoľnenie objektov. XR Interaction Toolkit je vysokoúrovňový interakčný systém založený na komponentoch, pomocou ktorých je implementovaný virtuálny zážitok. Rámec, ktorý tento balík poskytuje, sprístupňuje interakcie 3D a UI rozhrania zo vstupov, ktoré sú poskytnuté pre Unity. To umožňuje rýchlu a efektívnu implementáciu interakčných mechanizmov bez potreby vytvárania základných systémov od začiatku.

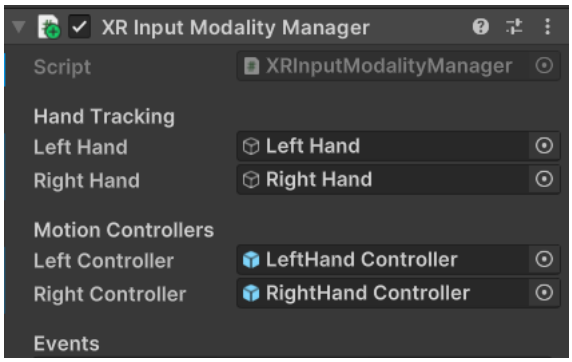
Základné komponenty poskytnuté týmto balíkom sú:

- XR Interaction Manager - Riadi všetky interakcie v scéne. Rieši aj detekciu a spracovanie vstupov zo zariadení. Interaction Manager vystupuje ako sprostredkovateľ medzi interaktormi a interactables. V projekte je možné mať viac interaktorov a každý z nich má svoju vlastnú platnú sadu interakcií.
- NearFar Interactor - Poskytuje interakciu kombináciou interakcie na diaľku a na blízko. Ak je objekt vzdialený raycast (lúč) je viditeľný a interaguje s objektom. Ak sa objekt priblíži k ruke, raycast zmizne a interakcia je prenesená z raycastu na prsty.
- Gaze Interactor - Poskytuje interakciu s prvkami prostredníctvom pohľadu. Riešenie obsahuje raycasty, ktoré aktualizujú aktuálnu množinu platných cieľov interaktora.

- XR Ray Interactor - Umožňuje interakciu pomocou raycastov, ktoré vedú z ovládačov alebo ruky. Tento typ ovládania je prispôsobený na diaľku.
- XR Direct Interactor - Umožňuje interakciu pomocou uchopenia a manipulácie s objektmi. Tento typ ovládania je prispôsobený na blízko.
- XR Grab Interactable - Umožňuje objektom v scéne reagovať na uchopenie a manipuláciu. Je kľúčová pre vytvorenie realistickej interakcie.
- XR Simple Interactable - Umožňuje základné jednoduchšie interakcie. Vhodná pre objekty, ktoré nevyžadujú komplexnú manipuláciu.

XR Input Modality Manager

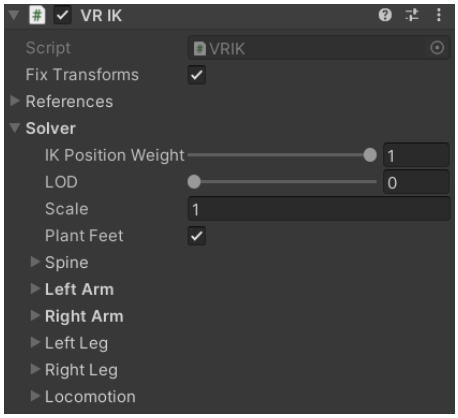
Tento komponent balíka XRI poskytuje výmenu medzi rukami a ovládačmi v scéne. Konrétnejšie je tento komponent možné využiť na výmeny ovládania medzi hand trackingom a ovládaním pomocou ovládačov v scéne. Ako vstupy tento komponent berie objekty hand trackingu (ľavú a pravú ruku) a objekty ovládačov v scéne. Komponent obsahuje aj konfiguráciu eventov, ktoré sa vykonajú pri aktivovaní a deaktivovaní hand trackingu a ovládania pomocou ovládačov.



Obr. 1.12: Komponent správcu vstupných XR modalít

1.5.7 Final IK

Final IK je balík vytvorený na animovanie avatarov s využitím inverznej kinematiky. Tento balík je už v projekte použitý. Avšak je potrebné zabezpečiť kompatibilitu funkcionality s hand trackingom. Dôležité je prepojenie a synchronizácia virtuálnych rúk a prstov s rukami a prstami avatara v reálnom čase. Hlavnými komponentmi balíka sú VR IK a Finger Rig.



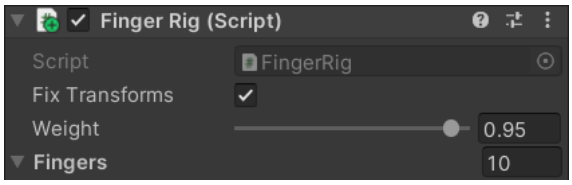
Obr. 1.13: Komponent VR IK

VR IK

Ako základ pre animáciu avatara slúži komponent VR IK. Tento komponent poskytuje konfiguráciu sledovania požadovaných bodov bodmi rúk avatara. Ak sú dostupné údaje pre Spine, Left Arm a Right Arm komponent pomocou týchto údajov dokaže simulovať pohyby celého avatara.

Finger Rig

Tento komponent poskytuje konfiguráciu a realizáciu pohybu prstov avatara. Obsahuje pole `Fingers`, v ktorom sú vytvorené premenné pre každý prst avatara, ktorým je možné pohybovať. Pre každý prst je potrebné konfigurovať jeho váhu, váhu otáčania, DOF rotácie, nastavenia rotácie, kosti rúk avatara, ktoré chceme v prste otáčať a kosť detekovanú pomocou hand trackingu, podľa ktorej sa nastaví koncový bod prsta avatara.



Obr. 1.14: Komponent rigovania prstov

1.5.8 Mirror

Ako uvádza dokumentácia [12], Mirror Networking poskytuje multiplayer funkcionalitu pre Unity projekty. Je postavený na nižšej úrovni sieťovej komunikácie a zabezpečuje mnoho bežných funkcií pre prenos dát medzi používateľmi. Funguje na princípe server autority. Umožňuje aj režim, pri ktorom je jeden z pou-

žívateľov klient a zároveň aj server. Tým odpadá potreba tvorby dedikovaného servera. V projekte sú použité obe spôsoby.

SyncVar

Atribút tried, ktoré dedia z NetworkBehaviour. Sú synchronizované zo servera na klientov. Pri vytvorení nového game objektu alebo pripojení používateľa je zaslaný najnovší stav všetkých SyncVars, ktoré sú pre nich viditeľné. Atribút hook je možné použiť na určenie funkcie, ktorá je zavolaná keď SyncVar zmení hodnotu.

NetworkTransform

Tento komponent synchronizuje polohu, rotáciu a mierku objektov na serveri. Každý objekt, ktorý je potrebné synchronizovať na serveri musí mať pridaný svoj vlastný Network Transform komponent. Tomuto komponentu je možné nakonfigurovať viacero parametrov. Medzi hlavné z nich patria:

- Target - Transform komponent pre synchronizáciu.
- Sensitivity - Senzitivita zmien potrebných pre synchronizáciu polohy, rotácie a mierky.
- Selective Sync a Interpolation - Voľba parametrov, ktoré sa majú synchronizovať (pozícia, rotácia, mierka).
- Sync Direction - Smer syncrhonizácie. Server to Client je predvolená hodnota, ktorá zabezpečuje serverovú autoritu. Client to Server umožňuje odosielať dáta na server. Ten dáta spracuje a odošle ostatným klientom.
- Sync Interval - Čas v sekundách, po ktorom sú synchronizované a odosielané zmeny.

Command

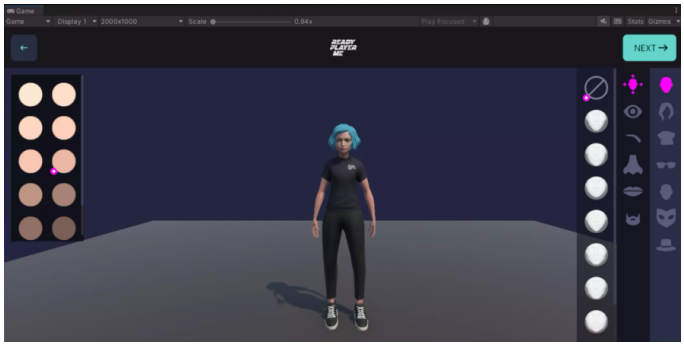
Odosiela sa z objektov klienta na objekty servera. Pre bezpečnosť môže byť príkaz odosielaný iba lokálnym používateľom, takže nie je možné ovládať objekty iných používateľov. Je možné obísť kontrolu oprávnenia pomocou príkazu (requiresAuthority = false). Z funkcie sa vytvorí príkaz tak, že sa jej pridá atribút [Command]. Takáto funkcia je spustená na serveri, kde je volaná klientom. Všetky parametre, ktoré sú povoleného dátového typu sú automaticky odovzdané serveru. Tieto funckie majú predponu Cmd a nemôžu byť statické.

ClientRPC

Odosiela sa z objektov servera na objekty klienta. Môže byť odoslané objektmi, ktoré majú priradený komponent NetworkTransform. Keďže autoritu má server, nevznikajú žiadne bezpečnostné problémy. Z funkcie sa vytvorí ClientRPC tak, že sa jej pridá atribút [ClientRPC]. Takáto funkcia sa vykoná na klientoch, keď je zavolaná na serveri. Všetky parametre, ktoré sú povoleného dátového typu sú automaticky odovzdané klientom. Tieto funckie majú predponu Rpc a nemôžu byť statické.

1.5.9 Ready Player Me

Balík umožňuje implementáciu 3D prispôsobiteľných avatarov do aplikácií a hier. Ready Player Me poskytuje platformu na vytváranie týchto 3D avatarov. Avatary môžu byť prispôsobené podľa vlastných preferencií, vrátane fyzických vlastností, oblečenia a doplnkov. Balík posyktuje jednoduchú integráciu s Unity. To umožňuje rýchlu implementáciu avatarov. Poskytuje API na načítanie, prispôsobenie a riadenie Avatarov v Unity.



Obr. 1.15: Prispôsobenie avatara ReadyPlayerMe

1.5.10 Meta XR All-in-One SDK

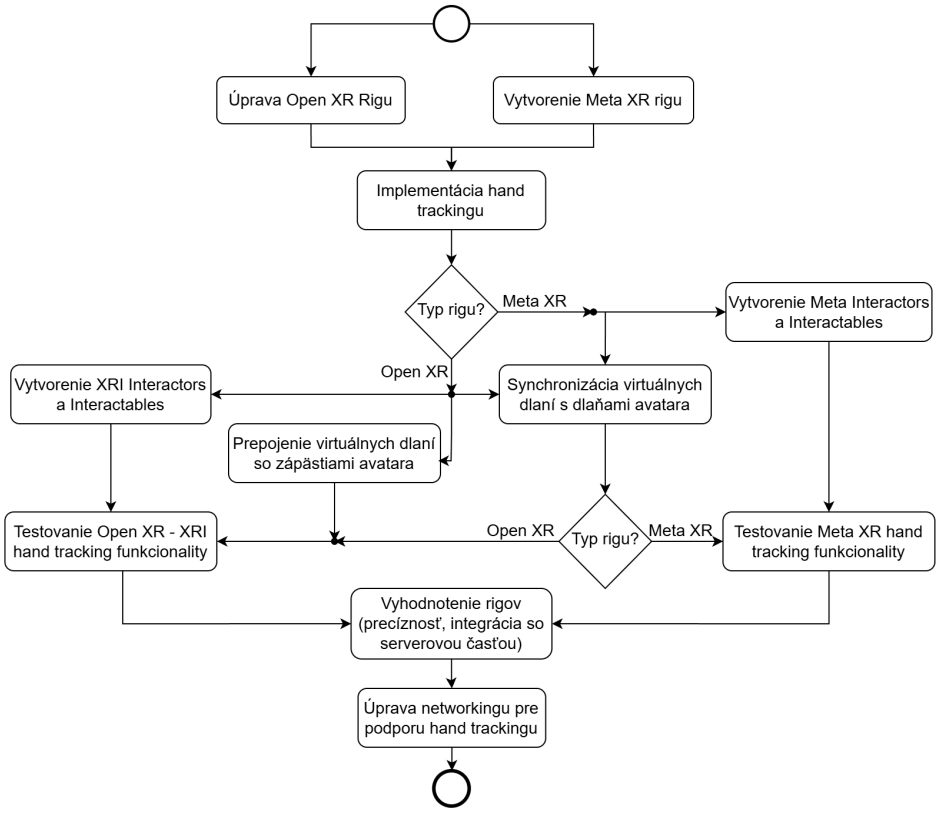
Balík je určený na vývoj aplikácií pre Oculus (Meta) zariadenia, ako sú napríklad Meta Quest 2 alebo Meta Quest 3. SDK poskytuje kompletnú podporu pre vývoj virtuálnej a zmiešanej reality, vrátane hand trackingu a interakcie s objektmi. Dokumentáciu SDK je možné nájsť na stránke [13].

2 Návrh a implementácia vylepšení systému

Základom pre implementáciu vylepšení je návrh. Nasledujúce diagramy prezentujú postupy, ktoré sú použité v tejto práci. Postupy obsahujú tvorbu a testovanie hand tracking rigov, interaktorov aj interactables a vylepšenie scalera.

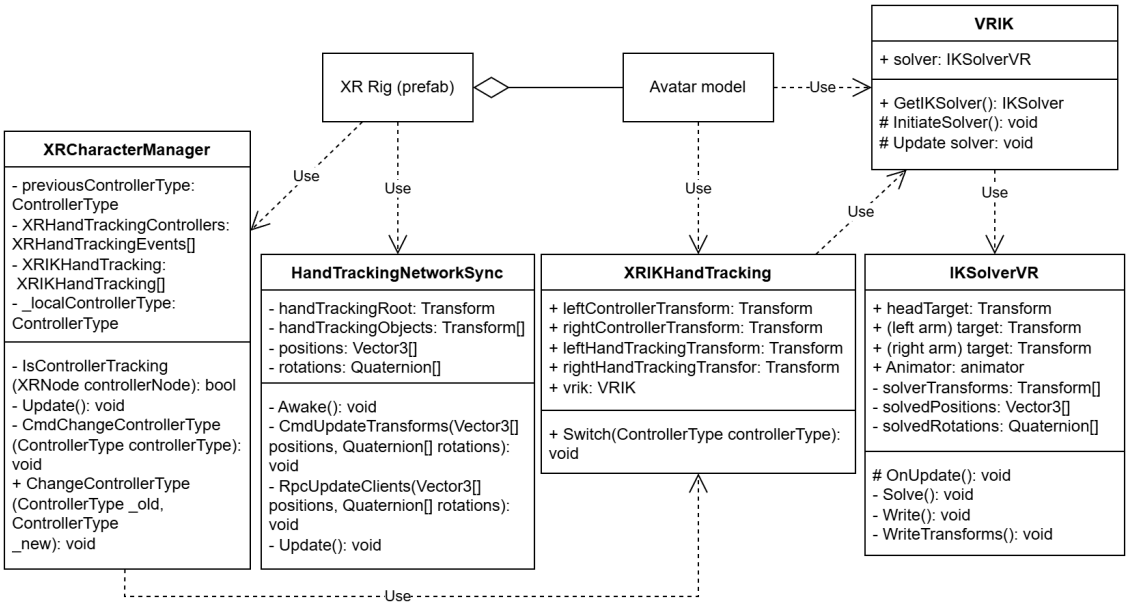
2.1 Návrh hand tracking interakcie

Návrh implementácie XR Rigov a ich interakcií je opísaný v diagrame 2.1. Tieto procesy opisujú postupy od súčasného stavu systému po systém s funkčnými singleplayer a multiplayer rigmi podporujúcimi hand tracking.



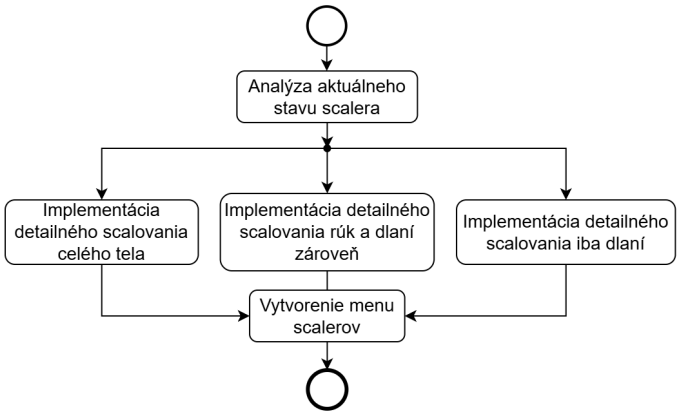
Obr. 2.1: Procesy implementácie XR Rigov a interakcií

V diagrame 2.1 je opísaná úprava vytvoreného Open XR Rigu pre podporu hand trackingu. Návrh obsahuje prepojenie virtuálnych rúk so zápästiami avatara aj synchronizáciu dlaní avatara so skrytím virtuálnych rúk. Pri Meta XR je v návrhu popísané vytvorenie nového rigu a synchronizácia dlaní avatara. Ku každému typu je potrebné vytvorenie príslušných interaktorov a interactables pre zabezpečenie správnej interakcie. Následne je potrebné porovnanie a vyhodnotenie precíznosti rigov a ich kompatibility so serverovým riešením aplikácie. Po vyhodnotení je potrebná úprava vybraného rigu pomocou knižnice mirror tak, aby rig podporoval networking.



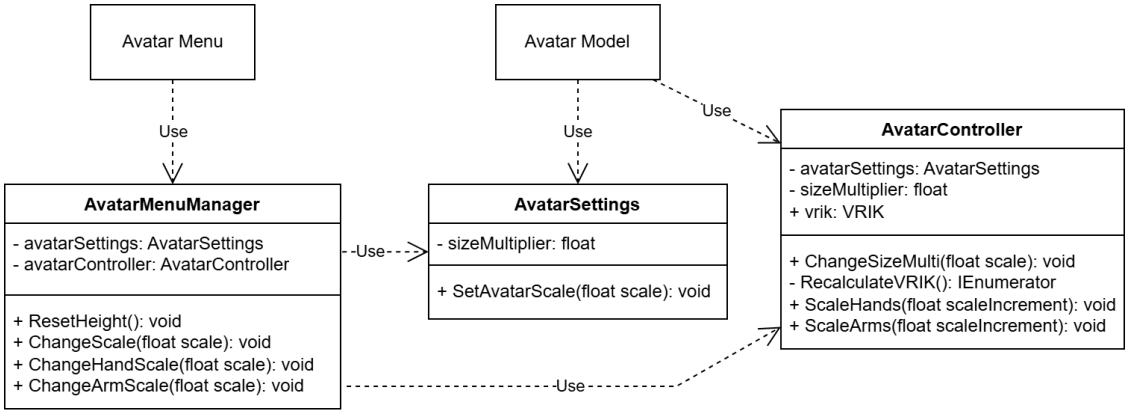
Obr. 2.2: UML diagram návrhu hand trackingu

V UML diagrame 2.2 sú opisané triedy, ktoré je potrebné upraviť/vytvoriť pre dosiahnutie funkcionality návrhu hand trackingu spolu so serverovým riešením. Pri úprave XRCharacterManager je potrebná implementácia detekcie a reakcie na vstup. Tým je zabezpečené, že systém správne detekuje, či sú aktívnym vstupom ruky alebo ovládače a podľa toho správne reaguje. Je vhodné zachovávanie hodnoty predošlého ovládania. Tým je pri zmene ovládania dosiahnuté správne prepnutie systému späť na posledné ovládanie pred hand trackingom. Novo vytvorená trieda XRIKHandTracking je používaná triedou XRCharacterManager. Pri prepnutí ovládania zabezpečuje zmenu cieľových bodov vo vytvorenom VR IK komponente modelu avatara. Vďaka tomu dokážu ruky avatara sledovať aktívne vstupy a prispôsobovať sa im. Na správny výpočet a vykreslenie avatara je triedou VR IK používaná trieda IKSolverVR. Novo vytvorená trieda HandTrackingNetworkSync zabezpečuje synchronizáciu prstov na dlaniach avatarov.



Obr. 2.3: Procesy implementácie vylepšenia scalera

Úprava vytvoreného scalera je opísaná v diagrame 2.3. Ako prvé je vhodné zistenie presnej funkcionality scalera. V ďalších krokoch je potrebné rozdelenie scalera na tri časti. Tým je zabezpečené detailné scalovanie avatara používateľom. Zároveň je potrebná implementácia logiky pridania a ubratia scalu, pretože momentálne je možné scalovanie avatara len pomocou automatického tlačidla Reset Scale. Následne je vhodné vytvorenie menu, ktoré obsahuje vytvorené scalery.



Obr. 2.4: UML diagram vylepšenia scalera

Triedy, ktoré je potrebné upraviť pre vylepšenie scalerov sú opísané v UML diagrame 2.4. AvatarMenuManager obsahuje funkcie, ktoré sú zavolané pomocou tlačidiel v menu. Následne sú podľa voľby scalera zavolané funkcie z tried AvatarSettings a AvatarController, ktoré zabezpečujú konkrétne scalovanie.

2.2 Postup implementácie vylepšení

V tejto a nasledujúcich sekciách kapitoly sú podrobne popísané navrhnuté a implementované postupy vylepšení systému pre neurorehabilitačný VR systém. Pr-

vým výstupom týchto postupov je vylepšený používaný XR Rig, pre ktorý je zabezpečené ovládanie a interakcia pomocou rúk. Tento rig pracuje v Menu scene podľa singleplayer konfigurácie a v Main scene podľa multiplayer konfigurácie, takže sú na serveri všetky pohyby rúk viditeľné pre každého používateľa.

Druhým výstupom je scéna, ktorá obsahuje jednoduché prostredie pre Meta XR All-in-One SDK funkcionalitu. V tejto scéne je implementovaný aj nový typ rigu, ktorý je plne kompatibilný s týmto balíkom. Scéna zabezpečuje prezentáciu a testovanie implementovaných funkcionalít pomocou tohto balíka.

Ďalším výstupom je úprava scaleru, ktorý bol nastavený iba na auto-calculate size. Použité postupy pre tieto výstupy obsahujú implementácie:

- Úprava používaného XR rigu pre podporu ovládania rukami.
- Implementácia ovládania rukami pre XR Rig.
- Vytvorenie a konfigurácia interaktorov a intractables balíka XR Interaction Toolkit.
- Vytvorenie a konfigurácia nového rigu Meta XR OVRCameraRigInteraction s podporou hybridného ovládania (jedna ruka a jeden ovládač).
- Vytvorenie a konfigurácia interaktorov a interactables balíka Meta XR All-in-One SDK.
- Vylepšenie scalera používaného v projekte a úprava hlavného menu pre Meta XR OVRCameraRigInteraction Rig.
- Úprava vylepšeného XR rigu pre zabezpečenie správneho Mirror networkingu.
- Zabezpečenie Mirror networkingu pre hand tracking na serverovej časti (XR Rig).

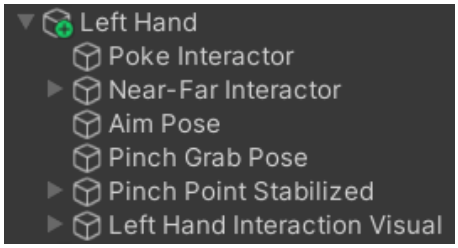
2.3 Open XR hand tracking

Základ hand trackingu v systéme je implementovaný pomocou balíkov XR Hands (verzia 1.5.0) a XR Interaction Toolkit (verzia 3.0.6). V prvom kroku je potrebné povolenie podpory Meta Hand Tracking Aim a Hand Tracking Subsystem v nastaveniach OpenXR pre Android a Windows.

Do scény sú pridané objekty virtuálnych rúk. Ich názov je Left Hand a Right Hand. Ruky sú implementované z balíka XR Interaction Toolkit z prefabu XR Origin Hands (XR Rig). Prefab je unpacknutý a z neho sú vytiahnuté objekty Right

Hand, Left Hand a Hand Visualizer. V ďalšom kroku je pre rig nakonfigurovaný komponent Hand Visualizer. Objekty sú vytvorené ako child objekty pre objekt Camera Offset. Pre Hand Visualizer komponent sú nastavené hodnoty:

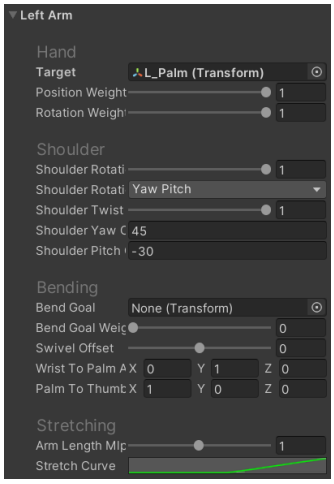
- Use Optimized Controls - Ruky avatara pracujú v poriadku s oboma nastaveniami tohto parametra.
- Left Hand Mesh - Left Hand Interaction Visual - Tento objekt je child objekt ruky, ktorá je v projekte importovaná ako objekt Left Hand.
- Right Hand Mesh - Left Hand Interaction Visual - Tento objekt je child objekt ruky, ktorá je v projekte importovaná ako objekt Right Hand.
- Hand Mesh Material - Nie je použitý žiaden objekt (tým je dosiahnutá vizualizácia priehľadného objektu ruky). V projekte je pre potrebu vytvorený Hand_M, ktorý slúži ako jednoduché zafarbenie rúk. Tento materiál je nakonfigurovaný ako bežná farba.
- Draw Meshes - Momentálne je v projekte vypnuté, kedže je dosiahnuté prepojenie medzi avatarom, ktorého ruky sa hýbu ako hand tracking ruky.
- Debug Draw Prefab - Prefab Joint z prefabov balíka XR Hands.
- Debug Draw Joints - Momentálne vypnuté, pri zapnutí je možné efektívne testovanie polohy a rotácie kostí.
- Velocity Prefab - Prefab Velocity Prefab z prefabov balíka XR Hands.
- Velocity Type - Hodnota None, pretože nie je v projekte momentálne vizualizuovaná.



Obr. 2.5: Hierarchia ľavej ruky

Každá ruka obsahuje child objekty:

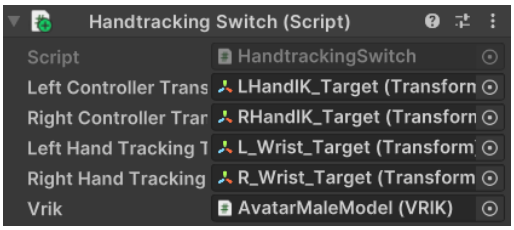
- Poke Interactor - Umožňuje ruke ”pichnúť” do objektov, čím sa aktivuje interakcia. Je využitý na dotykové interakcie s virtuálnymi tlačidlami alebo povrchmi.
- NearFar Interactor - Interakcia s objektmi je umožnená v blízkom aj vzdialenom dosahu. Je použitý na dosahovanie a manipuláciu s objektmi, ktoré sú fyzicky mimo dosahu ruky.
- Aim Pose - Póza ruky pri mierení. Je použitý najmä na interakciu s objektmi na diaľku, kde je potrebné ukazovanie alebo cielenie.
- Pinch Grab Pose - Póza ruky pri chytaní objektov pomocou pinch gesta (úchop objektov medzi palcom a ukazovákom).
- Pinch Point Stabilized - Stabilizovaný bod pre úchop. Zlepšuje presnosť a stabilitu počas interakcie s objektmi pomocou "pinch"gesta.
- (Left/Right) Hand Interaction Visual - Obsahuje child objekty vizualizujúce ruku a jej časti.



Obr. 2.6: VR IK nastavenie pre ľavú ruku

V projekte je implementovaná základná funkcionalita tohto komponentu. Avšak pri implementácii hand trackingu sú v tomto komponente vykonané zmeny. Tieto zmeny sa sústredia na cieľové body pre ľavú a pravú ruku. Sú to body, ktoré ruka sleduje a podľa nich sa hýbe. Funkcionalitu týchto cieľových bodov je potrebné zmeniť tak, aby sa správali podľa nastavenia ovládania (Hand Tracking/Ovládače). Pri hand trackingu je potrebné získanie týchto cieľov z virtuálnych

rúk. Tieto body sú body zápästia ľavej a pravej ruky. Pri ovládaní pomocou ovládačov tieto ciele musia byť body získané z polohy ovládačov. Túto funkcionalitu zabezpečuje nový skript Handtracking Switch.



Obr. 2.7: Komponent prepínača cieľových bodov pre OpenXR Rig

Skript obsahuje dve funkcie, a to sú hlavná funkcia `Switch()` a pomocná funkcia `IsControllerActive()`. Funkcia `Switch()` sleduje či sú v scéne aktívne virtuálne ruky alebo ovládače. Ak je aktívne ovládanie pomocou ovládačov, funkcia priradí pre hodnoty ľavej a pravej ruky avatara vo VR IK komponente body ovládačov, ktoré majú byť sledované a pomocou nich simulované pohyby. Ak tieto ovládače nie sú aktívne a je aktívne ovládanie pomocou virtuálnych rúk, funkcia priradí ako body, ktoré sú sledované body zápästí virtuálnych rúk. V takomto prípade ruky avatara sledujú zápästia virtuálnych rúk.

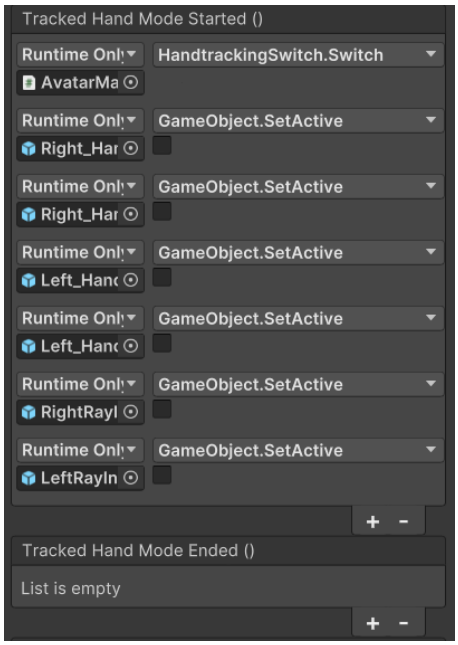
Funkcia `IsControllerActive()` zabezpečuje informáciu o tom, či je ovládač aktívny. Funkcii je poskytnutý parameter daného ovládača pre poskytnutie informácie, ktorý ovládač je potrebné sledovať a výstupom je informácia o tom, či je ovládač aktívny.

```
public Transform leftControllerTransform, rightControllerTransform;
public Transform leftHandTrackTransform, rightHandTrackTransform;
public VRIK vrik;
public void Switch(){
  if (IsControllerActive(XRNode.LeftHand)) {
    vrik.solver.leftArm.target = leftControllerTransform;
  }
  else {
    vrik.solver.leftArm.target = leftHandTrackTransform;
  }
  if (IsControllerActive(XRNode.RightHand)) {
    vrik.solver.rightArm.target = rightControllerTransform;
  }
  else {
    vrik.solver.rightArm.target = rightHandTrackTransform;
  }
}
```

Zdrojový kód 2.1: Funkcia `Switch()` triedy HandtrackingSwitch

Tento komponent je priradený pre objekt modelu avatara v scéne. Vstupmi pre tento komponent sú:

- Left Controller Transform - Cieľový objekt pre sledovanie ľavého ovládača.
- Right Controller Transform - Cieľový objekt pre sledovanie pravého ovládača.
- Left Hand Tracking Transform - Cieľový objekt pre sledovanie zápästia ľavej ruky.
- Right Hand Tracking Transform - Cieľový objekt pre sledovanie zápästia pravej ruky.
- VR IK - VR IK Komponent, v ktorom je vykonaná zmena ovládania (HandTracking/Ovládače).



Obr. 2.8: Unity udalosti pri zapínaní rúk

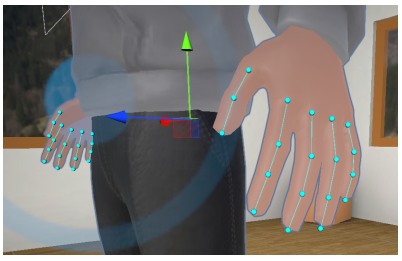
Funkcionalita tohto komponentu je použitá v XR Interaction Modality Managerovi pomocou eventov. Ak sa v XR Interaction Modality Managerovi aktivuje hand tracking, je zavolaná funkcia `Switch()` z komponentu Handtracking Switch. Tým je zabezpečené prepnutie sledovania na ruky. Ak sa v XR Interaction Modality Managerovi aktivuje ovládanie pomocou ovládačov, tak je rovnako zavolaná funkcia `Switch()`. Tá zabezpečí prepnutie sledovania na ovládače.

V týchto eventoch je zároveň zabezpečené vypínanie (obrázok 2.8) a zapínanie meshu rúk. Táto funkcionalita zabezpečuje, že ak je aktívny hand tracking,

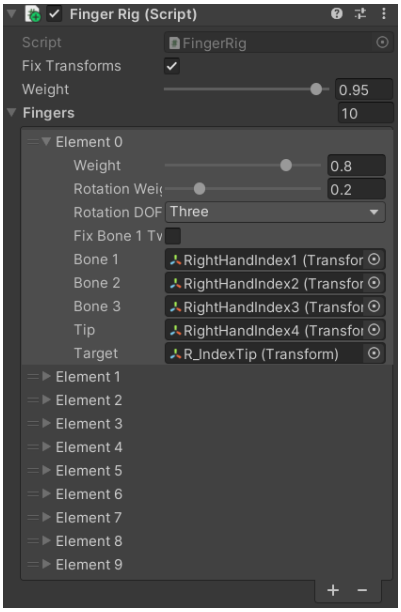
mesh ruky avatara je nastavený v sćene ako neaktívny a zmizne zo scény. Pri tejto funkcionalite je potrebné zapnutie parametra Draw Meshes na true v nastaveniach Hand Visualizera. Tento postup zabezpečuje zobrazenie rúk hand trackingu namiesto rúk avatara. Pri aktivovaní ovládania pomocou ovládačov sú tieto meshes nastavené ako aktívne. To zabezpečí, že ruky avatara sú znova zobrazené v scéne. Virtuálne ruky automaticky zmiznú, keďže nie je aktívne ovládanie pomocou hand trackingu.

Obr. 2.9: Virtuálne ruky prepojené s avatarom

Pre zabezpečenie pohybu prstov na dlani avatara je implementovaný komponent Finger Rig. Tento komponent obsahuje pole, do ktorého sú pridané prsty, ktoré sa majú pohybovať. Pomocou parametrov bone1, bone2 a voliteľného bone3 je nastavené, aké kosti prstu avatara sa majú pohybovať. Ako parameter Tip je koniec prsta na avatarovej ruke. Nastavenie týchto parametrov v editore zobrazí prepojenú čiaru medzi danými kosťami. Ako Target je koniec prsta na virtuálnych rukách. Komponent podľa tohto cieľa virtuálneho prsta vypočíta a nastaví pozíciu a rotáciu pre všetky kosti prstu ruky avatara v reálnom čase. Tým je zabezpečený pohyb prstov na dlani avatara súmerne s pohybom virtuálnych rúk používateľa v reálnom čase.



Obr. 2.10: Vizualizácia rigovaných prstov v editore



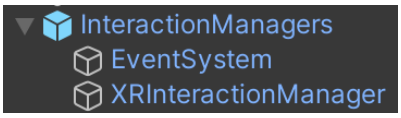
Obr. 2.11: Konfigurácia rigovania prstov

Týmito nastaveniami je zabezpečná vizualizácia a zobrazovanie rúk avatara v reálnom čase. Ruky sú namapované na virtuálne ruky z hand trackingu a pohybujú sa v scéne v reálnom čase podľa pohybu rúk používateľa. Ak sú ovládače aktivované, ruky opäť sledujú ovládače a virtuálne ruky nie sú aktívne.

Obr. 2.12: Ruky avatara synchronizované s virtuálnymi rukami

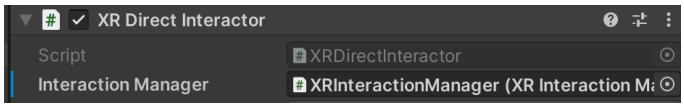
2.4 XRI Hand Interactors

Interakcia s objektmi je zabezpečená XR Interaction Managerom. Pre správnu konfiguráciu pre ovládače aj pre ruky je potrebné vykonať pár zmien. Na obrázku 2.13 je zobrazená hierarchia Interaction manažérov v scéne.



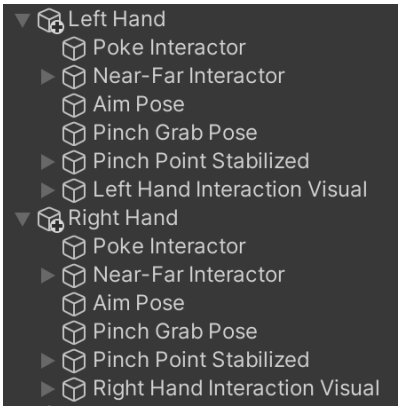
Obr. 2.13: Hierarchia interakčných manažérov

Každý ovládač potrebuje mať priradený komponent XR Direct Interactor. To už v projekte zabezpečené je no je potrebná jedna zmena. V komponente pre každý ovládač je potrebné priradiť ako parameter Interaction Manager konkrétny Interaction manažér v scéne:



Obr. 2.14: Komponent priameho XR interaktora

Pre interakciu pomocou rúk sú v predošlých krokoch vytvorené virtuálne ruky. Hierarchia ich interaktorov je zobrazená na obrázku 2.15.



Obr. 2.15: Hierarchia XRI interaktorov rúk

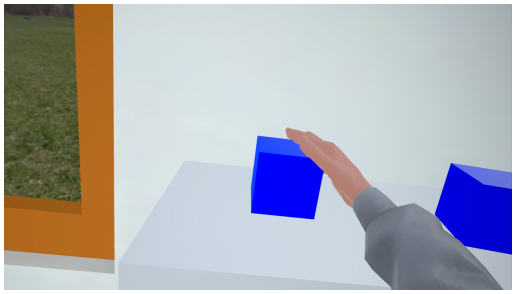
2.4.1 Poke Interactor

Tento interaktor je použitý na interakciu s tlačidlami alebo dotykovými povrchmi pomocou prsta. Funguje tak, že detekuje kolízie s objektmi, ktoré podporujú XR Poke Interaction. Tieto objekty sú vytvorené aj vysvetlené v ďalšej časti práce. Hlavné parametre XR Poke Interactora v projekte sú nakonfigurované nasledovne:

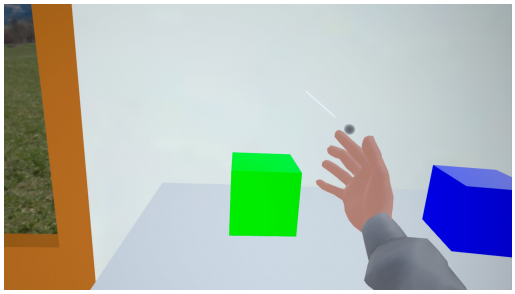
- Handedness - Left pre komponent ľavej ruky, Right pre komponent pravej ruky.
- Poke Depth - 0.1 (hĺbka, do ktorej musí prst preniknúť do objektu, aby sa aktivovala interakcia).
- Poke Width - 0.0075 (šírka detekčnej oblasti pre dotyk).
- Poke Select Width - 0.015 (šírka detekčnej oblasti pre aktiváciu výberu).
- Enable UI Interaction - Povolené (umožnenie interakcie s UI prvkami ako virtuálne menu).
- Physics Trigger Interaction - Ignore (eliminácia nežiadúcich fyzikálnych interakcií medzi prstom a objektmi, ktoré nepodporujú poke interakciu).

Poke Interactor obsahuje aj Tracked Pose Driver. Komponent slúži na správne priraďovanie polohy a rotácie rúk v 3D priestore na základe vstupných dát z handtrackingu. Hlavná konfigurácia v projekte:

- Tracking Type - Rotation and Position (sledovanie pozície a rotácie ruky).
- Update Type - Update And Before Render (aktualizácia prebieha v update cykloch a pred renderovaním).
- Position Input - XRI (Right/Left)/Poke Position (vstupné dáta pre pozíciu ruky).
- Rotation Input - XRI (Right/Left)/Poke Rotation (vstupné dáta pre rotáciu ruky).
- Tracking State Input - XRI (Right/Left)/Tracking State (sledovanie, či je handtracking aktívny).



Obr. 2.16: XRI interakcia pri dotyku objektu



Obr. 2.17: XRI interakcia pri vzdialení od objektu

2.4.2 NearFar Interactor

NearFar Interactor poskytuje interakciu zblízka aj na diaľku. Zblízka je uchopenie predmetov možné pomocou pinch gesta a na diaľku pomocou namierenia raycastov a následného pinch gesta.

Konfigurácia hlavnych komponentov NearFar Interactora v projekte:

NearFar Interactor

- Handedness - Left pre komponent ľavej ruky, Right pre komponent pravej ruky.
- Enable Near Casting - Povolené (interakcia s objektmi na blízko).
- Enable Far Casting - Povolené (interakcia s objektmi na diaľku).
- Select Input - Child objekt Select Input (spracovávanie vstupov na výber).

Komponent Interaction Attach Controller

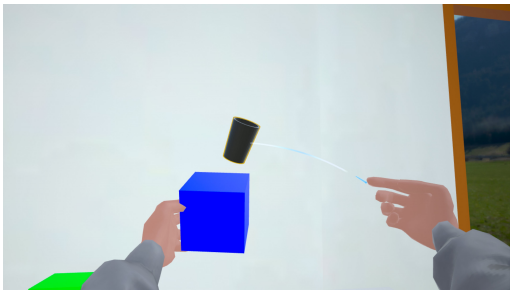
- Transform To Follow - Child objekt ruky Pinch Grab Pose (sledovanie a kopírovanie pohybu).
- Motion Stabilization Mode - With Position Offset (stabilizácia s posunom pozície).
- Position Stabilization - 0.25 (stabilizácia pozície).
- Angle Stabilization - 20 (stabilizácia rotácie).
- Use Distance Based Velocity - Povolené (rýchlosť pohybu objektu závisí od vzdialenosti od používateľa).

Sphere Interaction Caster

- Cast Origin - Child objekt ruky Pinch Grab Pose (správne nasmerovanie).
- Cast Radius - 0.1 (oblasť detekcie interakcie).

Curve Interaction Caster

- Cast Origin - Child objekt ruky Pinch Grab Pose (správne nasmerovanie).
- Cast Distance - 10 (maximálna vzdialenosť interakcie).



Obr. 2.18: XRI interakcia interaktora pre blízke a vzdialené ovládanie

2.5 XRI Virtual Interactables

Pre plné využitie handtrackingu v projekte je potrebné pridanie a konfigurácia interactables. Tieto spolupracujú s interaktormi, a tak tvoria interakcie. V projekte sú implementované dva typy interactables:

- Grab Interactable - Umožňuje používateľovi uchopiť objekt pomocou pinch gesta na ruke.
- Poke Interactable - Umožňuje používateľovi interakciu pomocou dotyku objektu.

Tieto interactables interagujú s objektmi:

2.5.1 Grab Interactable

Pre vytvorenie Grab Interactable objektu je potrebné pridať na objekt komponent XR Grab Interactable. Ak objekt neobsahuje komponent rigidbody, tento komponent je automaticky pridaný pri pridaní komponentu XR Grab Interactable.

Pre parameter Collision Detection v XR Grab Interactable je nastavená hodnota Continuous Dynamic. To znamená, že detekcia kolízií je neustále aktívna a dynamicky sa aktualizuje. Takto objekt plynulo reaguje na pohyb ruky/ovládača počas uchopenia a presúvania. Je potrebné priradiť Interaction Managera.

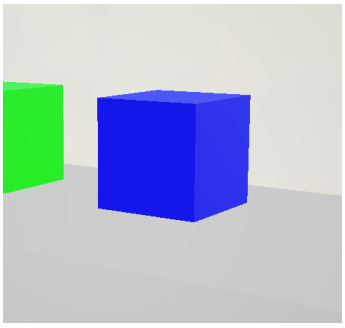
Rovnako ako pri ostatných objektoch je priradený XR Interaction Manager zo scény. Je možnosť aj nastavenia layer masky. V tomto prípade by sa interaktor a interactable nastavili na rovnakú vrstvu. V tomto projekte sú interakcie nakonfigurované na default layer.

Ako Select Mode je zvolená hodnota Single. To zabezpečuje interakciu pomocou jednej ruky. Ďalšia dôležitá vec je Movement Type. Táto možnosť ponúka výber z troch odlišných typov:

- Velocity Tracking - Objekt sa pohybuje podľa rýchlosti a pohybu ovládača.
- Kinematic - Objekt sa pohybuje hladko na základe kinematiky.
- Instantaneous - Objekt je udržiavaný v pohybe na základe fyzických síl pôsobiacich na objekt.

Ďalšími parametrami sú Track Position a Track Rotation. Pri každom komponente môžu byť nastavenia rôzne, no pri základnom interactable objekte sú obe možnosti povolené. To zabezpečí, že sa sleduje a mení pozícia a rotácia objektu podľa rúk alebo ovládačov. Nasleduje parameter Attach Transform. Tento parameter určuje pozíciu a rotáciu, ktorá bude centrom úchopu.

S takto nakonfigurovaným objektom je v projekte možné plne interagovať pomocou rúk alebo ovládačov. Pri úchope ovládačom je potrebné stlačenie grip tlačidla na boku Meta Quest ovládača. Pri handtrackingu je potrebné uchopiť objekt pomocou pinch gesta (spojenie ukazováka a palca).



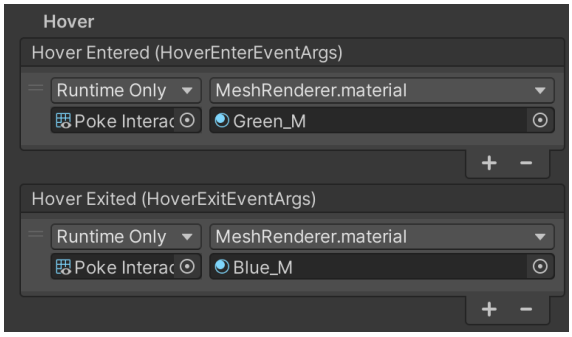
Obr. 2.19: XRI uchopiteľný objekt

2.5.2 Poke Interactable

Tento typ objektu je v projekte nakonfigurovaný tak, že je na objekt pridaný komponent XR Simple Interactable. Následne je objektu pridaný komponent XR Poke Filter. Pri tomto komponente je možnosť, že sa na objekt XR Poke Filter nepridá a na to je potrebné nastaviť parameter Poke Interactora - Require Poke Filter na nepovolený.

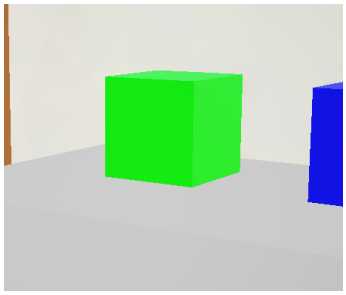
Ďalším parametrom pre Poke Interactable je Poke Direction v časti Poke Configuration. Tento parameter určuje smer, v ktorom sa má interakcia vykonať. Je možnosť výberu osí x, y, z a negatívnych hodnôt týchto osí.

Následne je v komponente XR Simple Interactable zabezpečená základná konfigurácia na zmenu farby objektu pri dotyku:



Obr. 2.20: XRI dotykové udalosti

Pri tejto konfigurácii je v projekte zabezpečená poke interakcia. Objekt zmení farbu pri dotyku a po ukončení dotyku je farba objektu ako pred dotykom.



Obr. 2.21: XRI dotykový objekt

2.5.3 Úprava existujúcich interakčných objektov

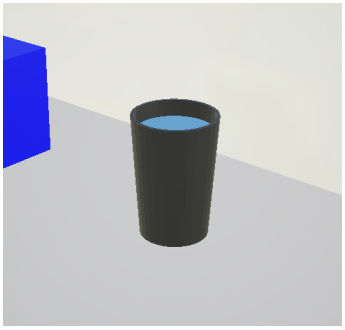
V projekte už existujú objekty, s ktorými sa vykonáva interakcia a tie pre správne fungovanie s handtrackingom a vytvorenými interaktormi musia byť upravené. Ako príklad je pohár. Je potrebné povolenie track rotation. Následne k povoleniu

rotácie je potrebné v komponente Rigidbody v časti constraints nepovoliť freeze rotation ani na jednej osi. To odblokuje rotácie objektu, avšak táto konfigurácia závisí od potrieb používateľa.

Avšak hlavná vec, ktorú je potrebné doplniť je Attach Transform. Pri tomto parametri je potrebné zabezpečiť pre každý objekt dva attach transform objekty, pretože ovládače aj ruky uchytia objekt rôznou rotáciou a tak je objekt pri týchto úchopoch odlišný. Pre pohár sú vytvorené child transform objekty:

- AttachPoint_Controllers - Position [0; 0; 0], rotation[90; 0; 0].
- AttachPoint_Handtracking - Position [0; 0; 0], rotation[90; 0; -90].

Prepínanie medzi týmito attach transforms je zabezpečené v XR Input Modality Manager komponente objektu XRRig. Pri evente Tracked Hand Mode Started je zabezpečené prepnutie attach transformu pre objekt pohára na AttachPoint_Handtracking a pri evente Motion Contoller Mode Started je to AttachPoint_Controllers.



Obr. 2.22: XRI objekt pohára

2.6 Hybridný vstupný systém

Hybridné ovládanie v tomto projekte znamená možnosť ovládania systému rôznymi vstupmi súčasne. Ovládanie poskytuje možnosť kombinácie ovládania ovládačom a zároveň rukou pomocou hand trackingu. Napríklad, pravá ruka je ovládaná ovládačom a ľavá pomocou ruky snímanej hand trackingom. Rovnako je to možné aj opačne a to kombináciou pravej ruky a ľavého ovládača. Toto ovládanie poskytuje používateľom možnosť flexibility a prispôsobenia ovládania pri rehabilitácii. Pacient si tak môže vybrať, ktoré ovládanie na konkrétnej ruke mu vyhovuje viac a viac vyhovuje jeho aktuálnym schopnostiam ovládania systému.

Pri tomto spôsobe ovládania pacienti môžu začínať s plným ovládaním ovládačmi a postupne prechádzať na ovládanie rukami, čo podporuje zlepšovanie

motorických schopností krok za krokom. Pri dlhodobom používaní a rehabilitácií môže používanie hybridného ovládania znížiť zaťaženie svalových skupín.

Hybridné ovládanie podporuje aj rozvoj jemnej motoriky a koordinácie medzi rukami a prstami. To je jednou z kľúčových vecí pri rehabilitácii. Sledovanie oboch možností umožňuje terapeutovi podrobnejšie sledovanie rôznych pohybov, vďaka čomu je poskytnutý širší pohľad na zručnosti pacienta a jeho pokrok.

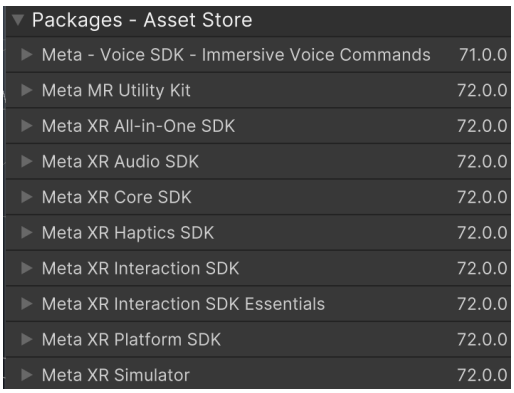
Možnosť kombinovať ovládače a ruky môže byť pre pacientov viac zábavná, čo zvyšuje ich motiváciu a ochotu pokračovať ďalej v rehabilitácii. Pre takéto ovládanie je samozrejme aj širšia škála mechanizmov, ktoré sú takýmto ovládaním pre pacienta zábavnejšie. Napríklad cvičenie, pri ktorom by pacient ovládačom ľavej ruky držal napríklad prak a pomocou pravej ruky by vzal loptičku a umiestnil ju na prak, natiahol prak a trafil nejaký bod.

2.6.1 Príprava balíka Meta XR All-in-one SDK

Pre implementáciu je vytvorená nová scéna HTtestScene, v ktorej sú implementované a testované všetky funkcionality a mechanizmy vytvorené cez Meta SDK.

Scéna obsahuje prvky potrebné pre zabezpečenie testovania, ako sú jednoduchá plocha na testovanie teleportu a ovládania, jednoduchá stena a jednoduchý stôl, na ktorom sú položené interactables. Nad stolom je umiestnené virtuálne menu určené na testovanie nových prvkov, ktoré je tiež možné ovládať pomocou hand trackingu. Scéna obsahuje aj zrkadlo z MenuScene, v ktorom je vidno avatar. Tým je zabezpečené lepšie testovanie hand trackingu a synchronizácie.

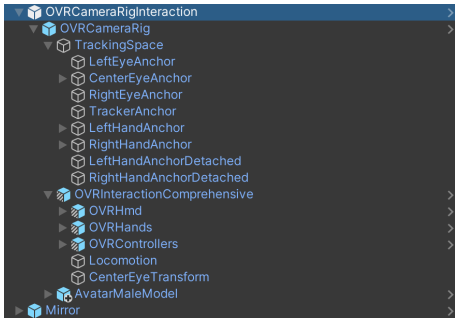
Na inštaláciu Meta XR All-in-One SDK je možné použiť asset store, kde je poskytnutý tento balíček k stiahnutiu zdarma. Po získaní tohto balíka je dostupný v projekte v Package Managerovi a obsahuje všetky tieto SDK (obrázok 2.23):



Obr. 2.23: Zoznam Meta All-in-One súborov nástrojov pre vývojárov

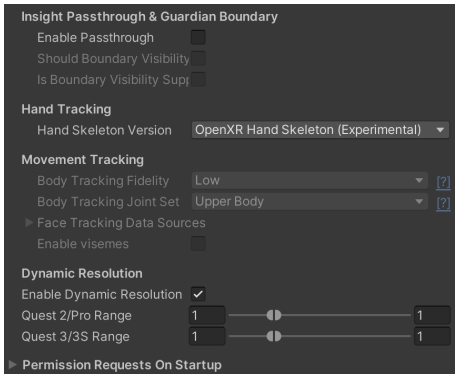
2.6.2 Zmena XRRig na OVRCameraRigInteraction

Ako prvá hlavná zmena, ktorá je aplikovaná je zmena rigu, ktorý je základom pre používateľa. Tento rig zabezpečuje dosiahnutie pokročilejšej interakcie s rukami a ovládačmi. Obsahuje množstvo nových prvkov, ktoré vylepšujú pôsobenie používateľa vo virtuálnom priestore. Všetky potrebné prvky sú vysvetlené tak, aby boli jednoducho pochopené.



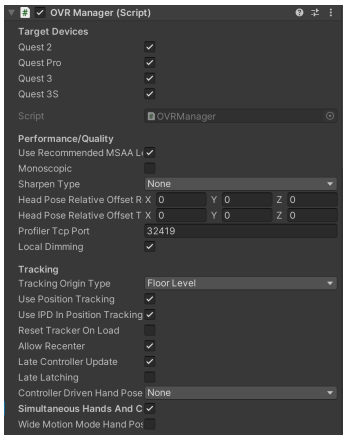
Obr. 2.24: Hierarchia prefabu OVRCameraRigInteraction

Hlavným komponentom tohto rigu je OVR manažér. Tento komponent slúži na správu XR funkcionalít v systéme. Nástroj je kľúčový pre optimalizáciu a rozšírenie možností interakcie v systéme. Hlavné funkcie OVR manažéra:

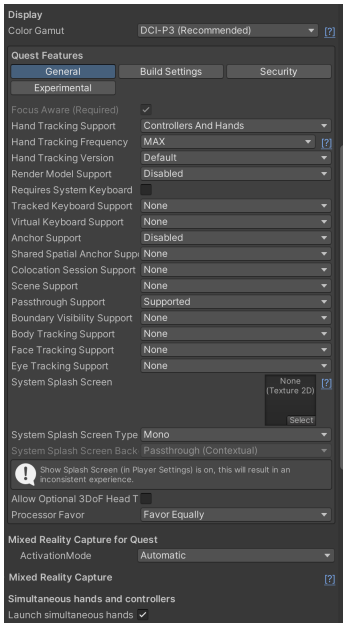


Obr. 2.25: OVR manažér a

- Podpora pre Meta zariadenia - Výber cieľových zariadení.
- Nastavenia sledovania (Tracking) - Konfigurácia hand trackingu, ovládačov, pôvodu sledovania a podobne.
- Nastavenia hand trackingu a hybridného ovládania – Umožňuje používanie rúk a ovládačov súčasne.
- Passthrough interakcie – Podpora zmiešanej reality vrátane interakcií a vizuálneho prechodu medzi virtuálnym a reálnym svetom.



Obr. 2.26: OVR manažér b



Obr. 2.27: OVR manažér c

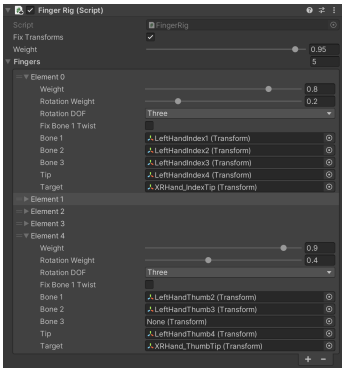
Táto funcionalita je vyvíjaná pre zariadenie Meta Quest 2 a 3 a tak je hlavná aktivácia týchto zariadení v OVR manažérovi ako na obrázku 2.26.

Taktiež je dôležité v tomto komponente nastavenie parametra Hand Tracking Support na Controllers and Hands pre umožnenie prepínania medzi ovládačmi a rukami podľa potreby. Pre Hand Tracking Frequency je v systéme zvolená hodnota MAX. Táto hodnota zabezpečuje najvyššiu obnovovaciu frekvenciu sledovania rúk. Senzory snímajú pohyb rúk čo najčastejšie. To zabezpečuje plynulú a presnú detekciu pohybov. Pre rehabilitačný systém je tento parameter kľúčový a maximálna frekvencia je najlepšou voľbou.

Dôležité pre komponent je aj nastavenie parametra Simulataneous Hands and Controllers na enabled čo zabezpečuje možnosť využívania kombinovaného ovládania pomocou ruky a ovládača zároveň. Bežne Meta Quest povoľuje buď ovládanie oboma ovládačmi alebo rukami. Aktivácia tohto nastavenia umožnuje používanie napríklad jedného ovládača v ruke a druhej ruky ako hand tracked vstupu. Avšak pre správne fungovanie systému je potrebné zabezpečenie ešte ďalších častí. Do OVRCameraRigInteraction rigu je vložený model avatara. V tomto prípade AvatarMaleModel. Dôležitým prvkom pre model je armature, ktorá je veľmi dôležitá pre synchronizáciu a pohyb. Armature musí obsahovať kosti prstov, ak je potrebné zabezpečiť ich pohyb.

Na objekt avatara je pridaný komponent Finger Rig. Nastavenie je opísané v kapitole Open XR hand tracking 2.3. Avšak v tomto kroku je použitie tohto kom-

ponentu vylepšené. Na rozdiel od použitia jedného komponentu, ktorý pracuje pre všetkých 10 prstov oboch rúk je komponent vložený dvakrát. Každý komponent pracuje iba s prstami jednej ruky. To zabezpečuje možnosť zapínať a vypínať Finger rig na každej ruke zvlášť. Zmena XRRigu si zároveň vyžaduje zmenu určitých parametrov v komponente Finger Rig.



Obr. 2.28: Konfigurácia rigovania ľavej ruky

Pre každý Finger Rig sú ako prvé zadané kosti prstov z Armature avatara. S týmito kosťami avatara Finger Rig pracuje a mení ich polohu a rotáciu. Ako Target je nastavený XRHand_IndexTip. Tento objekt je objektom rúk nového rigu. Kosti prstov avatara sú konfigurované na základe tohto objektu.

Pri aktivácii ovládania rukou sa zároveň zobrazí aj konkrétna virtuálna ruka. Pri tomto rigu však nie je použitý rovnaký objekt Hand Visualizer s komponentom pre správu vizualizácie rúk ako pri prvom. Nový rig obsahuje pre každú ruku osobitný objekt, ktorý vizualizuje danú ruku.

Obr. 2.29: Hierarchia objektu vizualizácie ľavej ruky

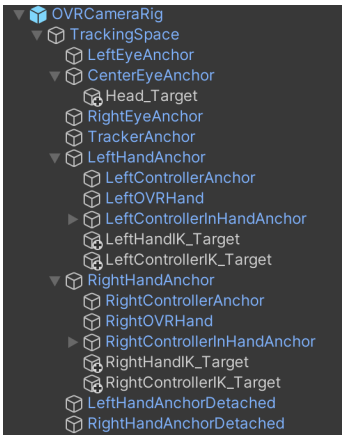
Obe tieto objekty obsahujú viacero vizualizácií rúk podľa konfigurácie rigu. V tomto prípade používame OpenXRLeftHand a OpenXRRightHand. Preto sú OculusHand_L a OculusHand_R vypnuté.

Aby funkcionalita virtuálnych rúk pri behu aplikácie bola zachovaná a mesh nebola viditeľná je potrebné deaktivovanie prvkov LeftHand a RightHand, ktoré sú child objekty OpenXR rúk. Týmto je zabezpečené skrytie virtuálnych rúk a zachovanie ich funkcionality, takže je zachované aj trackovanie zápästiami avatara.

Pre aktuálny rig je potrebné vytvorenie nových bodov sledovania zápästí. Rig obsahuje objekt Tracking Space, ktorý obsahuje objekty kotiev slúžiace ako referenčné body pre sledovanie polohy a orientácie rôznych častí tela vo VR priestore.

Hlavné poskytnuté bodové kotvy sú:

- Left Eye Anchor - Pozícia a orientácia ľavého oka používateľa.
- Right Eye Anchor - Pozícia a orientácia pravého oka používateľa.
- Center Eye Anchor - Stredová pozícia medzi ľavým a pravým okom používateľa.
- Left Hand Anchor - Pozícia a orientácia ľavej ruky používateľa.
- Right Hand Anchor - Pozícia a orientácia pravej ruky používateľa.



Obr. 2.30: Hierarchia bodových kotiev sledovania priestoru

Ako je vidno na obrázku 2.30 pre synchronizáciu polohy a rotácie rúk avatara ovládačmi sú vytvorené objekty LeftControllerIK_Target a RightControllerIK_Target. Pre synchronizáciu pri ovládaní rukami pomocou hand trackingu sú vytvorené body RightHandIK_Target a LeftHandIK_Target. Pre synchronizáciu polohy hlavy (hlavná kamera, pomocou ktorej použíateľ vidí virtuálny svet z prvej osoby) je vytvorený bod Head_Target. Tieto cieľové objekty je potrebné presne umiestniť a zrotovať tak, aby sa dosiahlo správne zobrazenie avatara.

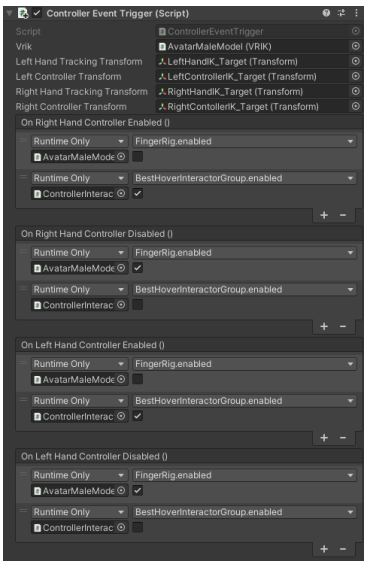
Pri uchopení ovládačov je potrebné objekty nakonfigurovať tak, aby bola ruka nasmerovaná na začiatok ovládača a zároveň aby sedela rotácia úchopu. Pri hand trackingu netreba konfigurovať pozíciu ale iba rotáciu, aby sa predišlo nesprávnemu zobrazeniu meshu rúk avatara. Pri hlave je potrebné mierne vysunúť pohľad aby nebola mesh avatara viditeľná zvnútra.

Target	Poloha [x,y,z]	Rotácia [x,y,z]
Head	[0; -0.1; -0.1]	[0; 0; 0]
LeftHandIK	[0; 0; 0]	[90; 0; 0]
LeftControllerIK	[-0.02; -0.05; -0.12]	[0; 90; 90]
RightHandIK	[0; 0; 0]	[90; 0; 0]
RightControllerIK	[0.03; -0.05; -0.11]	[0; -90; -90]

Tabuľka 2.1: Pozície a rotácie cieľových objektov

2.6.3 Komponent Controller Event Trigger

Vytvorením tohto komponentu je zabezpečené vylepšenie komponentu Handtracking Switch z prvej časti tak, že tento komponent pracuje nezávisle na každej ruke zvlášť a zároveň je kompatibilný s novým rigom.



Obr. 2.31: Komponent prepínača cieľových bodov pre MetaXR Rig

V skripte je implementovaná následná logika. Ako prvý parameter, ktorý je potrebne v skripte nastaviť je VR IK komponent, ktorý je nastavený na modeli avatara. V tomto komponente sa nachádzajú premenné, ktorým sú priradené body pozície dlaní avatara. Pomocou týchto bodov a bodu pozície hlavy tento komponent nakonfiguruje mesh avatara a aktualizuje ho.

S týmto sú prepojené ďalšie dve parametre komponentu. Skript sleduje, či je na ľavej a pravej ruke osobitne aktívne ovládanie ovládačom alebo hand trackingom. Podľa toho priradí komponentu VR IK pre danú ruku buď `Left/Right Hand Tracking Transform` alebo `Left/Right Controller Transform` (zdrojový kód 2.2). Pomocou týchto premenných sa hýbu a synchronizujú zá-

pästia avatara. Následne sú v skripte vytvorené UnityEventy, ktoré sa vykonajú so zmenou bodov kotiev pre ruky:

- On Right Tracking Hand Controller Enabled - Pri ovládaní ovládačom sa vypne Finger Rig na pravej ruke, aby sa ruka dostala do polohy držania ovládača. Následne je povolený Ray Interactor ovládača pre interakciu s prostredím.
- On Right Tracking Hand Controller Disabled - Finger Rig na pravej ruke sa zapne aby bol aktivovaný synchronizovaný pohyb prstov pravej ruky. Následne je vypnutý Ray Interactor ovládača.
- On Left Tracking Hand Controller Enabled - Rovnaká funcionalita ako pri On Right Tracking Hand Controller Enabled ale pre ľavú ruku.
- On Left Tracking Hand Controller Disabled - Rovnaká funcionalita ako pri On Right Tracking Hand Controller Disabled ale pre ľavú ruku.

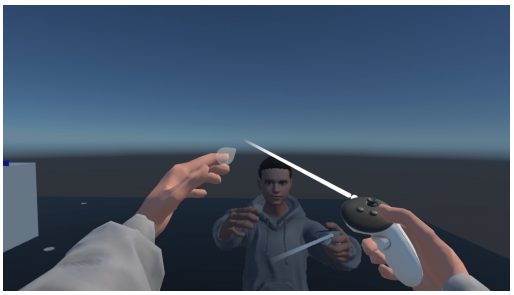
```
private void Update(){
  CheckLeftControllerStatus();
  CheckRightControllerStatus();
}

private void ChechRightControllerStatus(){
  InputDevice device = InputDevices.GetDeviceAtXRNode(XRNode.RightHand);
  if (device.isValid){
    bool ContrEnb;
    if (device.TryGetFeatureValue(CommonUsages.isTracked, out ContrEnb) && ContrEnb){
      if (!isRightHandControllerActive){
        isRightHandControllerActive = true;
        OnRightHandControllerEnabled?.Invoke();
        vrik.solver.rightArm.target = rightControllerTransform;
      }
    }
    else {
      if (isRightHandControllerActive){
        isRightHandControllerActive = false;
        OnRightHandControllerDisabled?.Invoke();
        vrik.solver.rightArm.target = rightHandTrackingTransform;
      }
    }
  }
}
```

Zdrojový kód 2.2: Funkcie `Update()` a `CheckRightContollerStatus()` triedy ControllerEventTrigger

Týmito krokmi je dosiahnuté funkčné hybridné ovládanie. Pri spustení aplikácie systém hneď sleduje vstupy používateľa a priraďuje im konkrétnu funkcionalitu. V reálnom čase je zabezpečené prepínanie medzi ovládačmi a rukami.

Ak používateľ položí ovládač na stôl, model aj raycast ovládača zmizne a zobrazí sa synchronizovaná ruka. Ako pomocné body pre lokalizáciu ovládača slúžia malé kruhové body pred ovládačmi, ktoré aj po zmiznutí ovládačov ostávajú viditeľné.

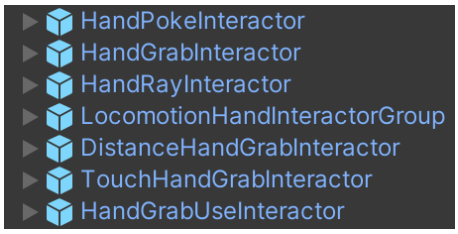


Obr. 2.32: Hybridné ovládanie avatara

2.7 Meta XR Hand Interactors

Meta XR ponúka množstvo interaktorov, ktoré je možné používať. Tieto objekty sú umiestnené ako child objekty rúk OVRCameraRigInteraction rigu. Každý interaktor ponúka jedinečnú funkcionalitu. Zoznam interaktorov dostupných v rámci OVRCameraRigInteraction tak pokrýva rôzne potreby a umožňuje prispôsobiť ovládanie rúk podľa požiadaviek projektu. Dostupné interaktory a všetky informácie ohľadom ich implementácie do projektu je možné nájsť v oficiálnej dokumentácii Meta pre Unity [14].

Zoznam interaktorov použitých v projekte na každej ruke (obrázok 2.33):

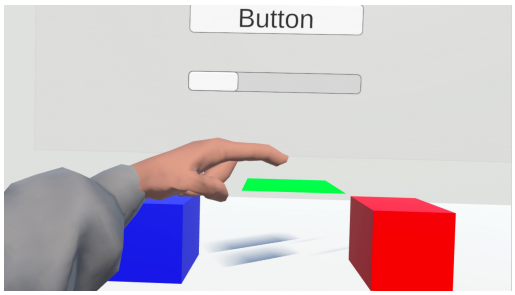


Obr. 2.33: Zoznam Meta XR interaktorov virtuálnych rúk

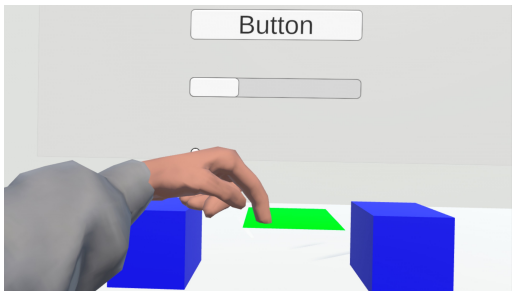
2.7.1 HandPokeInteractor

Tento systém umožňuje interakciu dotykom. Funguje tak, že sleduje bod v priestore a zisťuje, či sa dotýka nejakého interaktívneho povrchu (Poke Interactable). Ak sa bod priblíži k povrchu, systém rozpozná, že je blízko. Ak sa následne prst

pohne tak, že prenikne cez povrch v smere jeho normály (kolmo na povrch), systém to vyhodnotí ako stlačenie. Takto je umožnené napríklad stláčanie tlačidiel dotykom prsta. Pre implementáciu HandPokeInteractoru je potrebné na objekt každej ruky a controllera pridať komponent Poke Interactor.



Obr. 2.34: Meta dotykový interaktor (nestlačené tlačidlo)



Obr. 2.35: Meta dotykový interaktor (stlačené tlačidlo)

Konfigurácia Hand Poke Interactoru na objektoch v projekte je:

Komponent Hand Ref

- Hand - Left/Right (referencia na konkrétnu ruku interaktora).

Tento komponent je pridaný aj pre všetky ostatné interaktory v scéne a rovnako pridaný danej ruke.

Komponent Hand Poke Interactor

- Max Iterations Per Frame - 3.
- Point Transform - HandIndexFingertip pre ruky a PokeLocation pre contollery.
- Radius - 0.005.
- Touch Relese Threshold - 0.002.
- Equal Distance Threshold - 0.001.

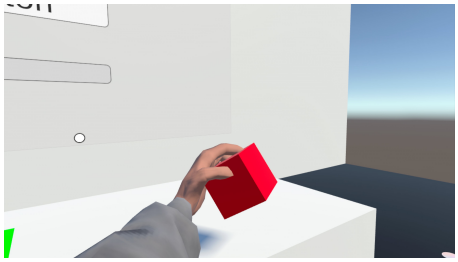
Komponent Active State Tracker

- Active State - HandPokeInteractor (Hand Ref).
- Include Children as Dependents - Povolené.

Tento komponent je pridaný aj pre všetky ostatné interaktory v scéne a rovnako nakonfigurovaný pre príslušný objekt interaktora.

2.7.2 HandGrabInteractor

Grab Interactor umožňuje uchopovať objekty pomocou virtuálnych rúk. Komponent je pripojený k ruke používateľa cez HandGrabInteractor prefab, ktorý je pripojený k ruke v rámci OVRCameraRigInteraction rigu.



Obr. 2.36: Meta interaktor priameho úchopu

Konfigurácia Hand Grab Interactoru na objektoch v projekte je:

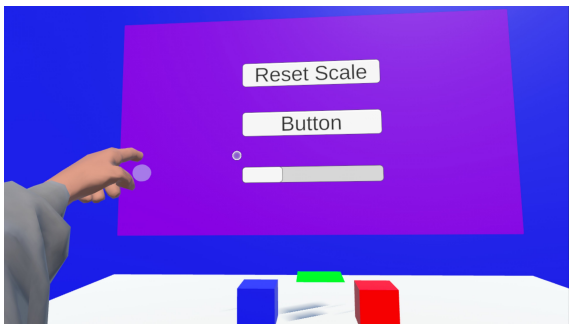
Komponent Hand Grab Interactor

- Max Iterations Per Frame - 3.
- Hand - HandGrabInteractor (Hand Ref).
- Rigidbody - Rigidbody (Rigidbody) = Child objekt objektu Hand Grab Interactor, ktorého komponent Rigidbody je nastavený na Use Gravity: Zakázané a is Kinematic: Povolené. Tento Rigidbody ma child objekty GripPoint, PinchPoint a PinchPointEase, ktoré obsahujú child objekty collidre a následné komponenty: GripPoint - Hand Root Offset, PinchPoint - Hand Pitch Offset, PinchPointEase - Filtered Transform.
- Hand Grab Api - HandGrabAPI (Hand Grab API) = Child objekt objektu Hand Grab Interactor, ktorý obsahuje komponent Hand Grab API, ktorého parameter Hand je HandGrabInteractor (Hand Ref).
- Supported Grab Types - Everything.
- Hover On Zero Strength - Povolené.
- Grab Origin - HandWristPoint = Child objekt objektu HandGrabInteractor, ktorý obsahuje komponent Hand Root Offset.
- (optional)Grip Point - GripPoint (Transform).
- (optional)Grip Collider - Collider (Sphere Collider) = Child objekt objektu GripPoint.

- (optional)Pinch Point - PinchPointEase (Transform).
- (optional)Pinch Collider - Collider (Sphere Collider) = Child objekt objektu PinchPointEase.
- (optional)Active State - HandGrabIntercator (Hand Ref).

2.7.3 HandRayInteractor

Tento interaktor definuje pôvod a smer raycastov pre interakciu pomocou lúča. Je možné nastaviť maximálnu vzdialenosť pre túto interakciu. Avšak tento interaktor sám o sebe nepodporuje mechanizmus výberu a tak je k nemu pripojený aj selector. V tomto projekte je zvolený pinch selector, čo znamená, že select je vykonaný spojením palca a ukazováka.



Obr. 2.37: Meta lúčový interaktor

Konfigurácia Hand Ray Interactoru na objektoch v projekte je:

Komponent Ray Interactor

- Max Iterations Per Frame - 1.
- Selector - Selector (Index Pinch Selector) = Child objekt objektu HandRayInteractor, ktorý obsahuje komponent Index Pinch Selector s parametrom Hand - HandRayInteractor (Hand Ref).
- Ray Origin - PointerPose (Transform) = Child objekt objektu Model (Child objekt objektu HandRayInteractor), ktorý obsahuje komponent Hand Pointer Pose s parametrom Hand - HandRayInteractor (Hand Ref).
- Max Ray Length - 5.
- Equal Distance Threshold - 0.001.
- (optional)Active State - HandRayInteractor (Active State Group).

Komponent Active State Group

- Active States - Element 0 = PointerPose (Hand Pointer Pose), Element 1 = HandRayInteractor(Interactor Active State).
- Logic Operator - OR.

Komponent Interactor Active State

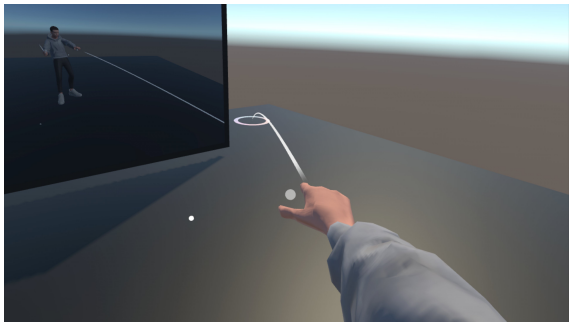
- Interactor - HandRayInteractor (Ray Interactor).
- Property - Has Selected Interactable.

2.7.4 LocomotionHandInteractorGroup

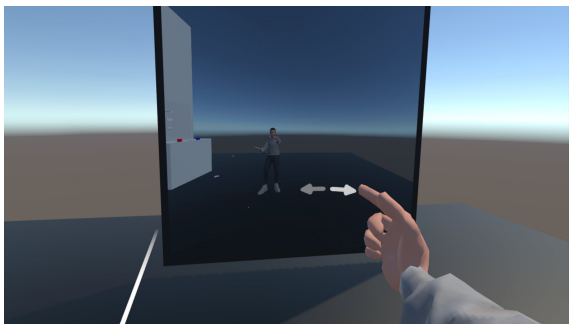
Hlavné interaktory tejto skupiny sú TeleportHandInteractor a TurnerInteractor. TeleportHandInteractor vytvára oblúk na miesto, kde sa chce používateľ teleportovať. Následne vytvorí na mieste kde je možný teleport modrý kruh. Ak teleport na miesto nie je možný, kruh buď vôbec nie je vytvorený alebo je červenej farby. Po potvrdení, ak je možný teleport na dané miesto, je používateľ prenesený na zvolenú pozíciu.

TurnerInteractor funguje ako 1D os, ktorá obsahuje šípky smerov otočenia. Používateľ si môže vybrať smer doprava alebo doľava a po pohnutí gesta týmto smerom je šípka zvýraznená. Ak používateľ potvrdí smer, je otočený.

Pre aktiváciu týchto interaktorov je v systéme potrebné ľubovoľnou rukou vytvoriť gesto. Gesto pozostáva z týchto pravidiel - ukazovák a palec musia byť vystreté smerom od dlane, prostredník, prsteník a malíček musia byť ohnuté smerom k dlani. Ak je ruka vo zvislej pozícii zobrazí sa TurnerInteractor. Ak je ruka vo vodorovnej pozícii zobrazí sa TeleportHandInteractor.



Obr. 2.38: Meta interaktor teleportu



Obr. 2.39: Meta interaktor otáčania

Konfigurácia objektu LocomotionHandInteractorGroup v projekte je:

Komponent Hmd Ref

- Hmd - OVRHmd (Hmd) = Child objekt objektu OVRInteractionComperhensive.

Komponent Best Select Interactor Group

- Interactors - Element 0 = TeleportHandInteractor (Teleport Interactor), Element 1 = TurnerInteractor (Locomotion Turner Interactor).
- Elements per frame - 3.

Komponent Locomotion Events Connection

- Broadcasters - Element 0 = TeleportHandInteractor (Teleport Interactor), Element 1 = TurnerInteractor (Turner Event Broadcaster).
- Handler - Locomotion (Player Locomotor).

Konfigurácia objektu TeleportHandInteractor (Child objekt objektu LocomotionHandInteractorGroup) v projekte je:

Komponent Hmd Ref (rovnaký aj pre TurnerInteractor)

- Hmd - LocomotionHandInteractorGroup (Hmd Ref).

Komponent Teleport Interactor

- Max Iteraction Per Frame - 3.
- Selector - Selector (Index Pinch Safe Release Selector) = Child objekt objektu TeleportHandInteractor.

- Teleport Arc - TeleportHandInteractor (Teleport Arc Gravity).
- Equal Distance Threshold - 0.1.
- Hmd - TeleportHandInteractor (Hmd Ref).
- Active State - TeleportActiveState (Virtual Active State) = Child objekt objektu LocomotionGate (Child objekt objektu LocomotionHandInteractorGroup).

Komponent Teleport Arc Gravity

- Origin - ArcOrigin (Transform) = Child objekt objektu TeleportHandInteractor, komponent Hand Root Offset (Parametre Hand - TeleportHandInteractor (Hand ref), Pos Offset - x = 0, y = -0.05, z = 0.12, Rot Offset - x = 0, y = 3.415095e-06, Z = 0, Mirror Offsets For Left Hand - Povolené).
- Stabilization Point - Shoulder (Transform) = Child objekt objektu TeleportHandInteractor, komponent Shoulder Estimate Position (Parametre Hmd - TeleportHandInteractor (Hmd Ref), Hand - TeleportHandInteractor(Hand Ref)).
- Gravity Modifier - 2.3.
- Arc Points Count - 30.

Komponent Interactor Unity Event Wrapper

- Interactor View - TeleportHandInteractor (Teleport Interactor).
- When Preprocessed() - Runtime Only -> TeleportHandInteractor (TeleportArcGravity) -> TeleportArcGravity.UpdateArcParameters.

Konfigurácia objektu TurnerInteractor (Child objekt objektu LocomotionHandInteractorGroup) v projekte je:

Komponent Locomotion Turner Interactor

- Max Iteraction Per Frame - 3.
- Origin - Origin (Transform) = Child objekt objektu TurnerInteractor.
- Selector - Selector (Index Pinch Safe Release Selector) = Child objekt objektu TurnerInteractor.

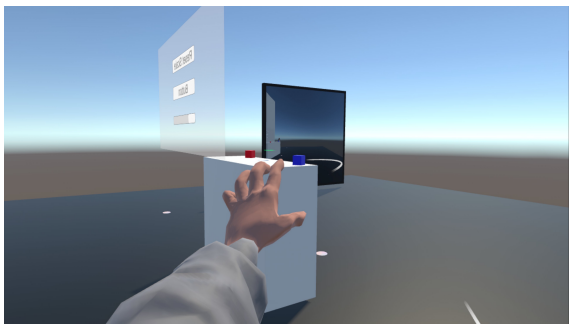
- Stabilization Point - Shoulder (Transform) = Child objekt objektu TurnerInteractor.
- Transformer - OVRInteractionComperhensive (Tracking To World Transformer OVR) = Child objekt objektu OVRInteractionComperhensive.
- Drag Threshold - 0.04.
- (optional)Active State - TurningActiveState (Virtual Active State) = Child objekt objektu LocomotionGate (Child objekt objektu LocomotionHandInteractorGroup).

Komponent Turner Event Broadcaster

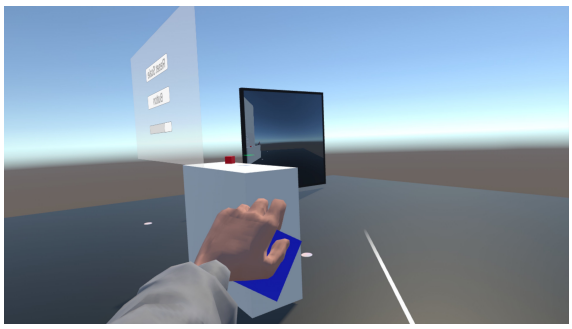
- Interactor - TurnerInteractor (Locomotion Turner Interactor).
- Axis - TurnerInteractor (Locomotion Turner Interactor).
- Turn Method - Smooth (Pravá ruka), Snap (Ľavá ruka).
- Snap Turn Degrees - 45 (Pri Turn Method snap).
- Fire Snap On Unselect - Povolené.

2.7.5 DistanceHandGrabInteractor

Tento interaktor umožňuje používateľovi uchopiť objekty v prostredí na diaľku bez toho, aby sa ich musel priamo dotýkať alebo byť veľmi blízko týchto objektov. V systéme to vyzerá tak, že používateľ namieri na objekt a ak mieri správne, tak je k objektu predĺžený raycast. Následne môže používateľ ukázať gesto úchopu a objekt mu je pritiahnutý priamo do dlane.



Obr. 2.40: Meta interaktor vzdialeného úchopu (mierenie na objekt)



Obr. 2.41: Meta interaktor vzdialeného úchopu (pritiahnutie objektu)

Konfigurácia objektu DistanceHandGrabInteractor v projekte je:

Komponent Hmd Ref

- Hmd - OVRHmd (Hmd) = Child objekt objektu OVRInteractionComperhensive.

Komponent Distance Hand Grab Interactor

- Max Iterations Per Frame - 3.
- Hand - DistanceHandGrabInteractor (Hand Ref).
- Hand Grab Api - HandGrabAPI (Hand Grab API).
- Supported Grab Types - Everything.
- Grab Origin - FilteredWristPoint (Transform) = Child objekt objektu DistanceHandGrabInteractor, komponent Hand Root Offset (Parametre Hand - DistanceHandGrabInteractor (Hand Ref), Rot Offset - x = 0, y = 90, z = 0).
- Selection Frustrum - HandFrustrumNarrow (Conical Frustrum) = Child objekt objektu FilteredWristOffset (Child objekt objektu DistanceHandGrabInteractor).
- Deselection Frustrum - HandFrustrumWide (Conical Frustrum) = Child objekt objektu FilteredWristOffset (Child objekt objektu DistanceHandGrabInteractor).
- Aid Frustrum - HeadFrustrum (Conical Frustrum) = Child objekt objektu DistanceHandGrabInteractor.
- Aid Blending - 0.

- Detection Delay - 0.1.
- (optional)Grip Point - GripPoint (Transform) = Child objekt objektu DistanceHandGrabInteractor.
- (optional)Pinch Point - PinchPoint (Transform) = Child objekt objektu DistanceHandGrabInteractor.
- (optional)Active state - DistanceHandGrabInteractor (Hand Ref).

2.7.6 TouchHandGrabInteractor

Na rozdiel od HandGrabInteractora, ktorý uchopí objekt so zachovanou ľubovoľnou polohou dlane, tento interaktor poskytuje detailné uchopenie objektu. Napríklad úchop pohára. Pri GrabInteractore je pohár uchopený a dlaň drží tvar päste. Pri použití TouchHandGrabInteractora sa prsty na dlani detailne prispôsobia povrchu meshu pohára.

Obr. 2.42: Meta úchopový detailný interaktor

Obr. 2.43: Meta interaktor na úchop a použitie

Konfigurácia objektu TouchHandGrabInteractor v projekte je:

Komponent Touch Hand Grab Interactor

- Max Interations Per Frame - 3.
- Hand - TouchHandGrabInteractor (Hand Ref).
- Open Hand - HandSphereMap (Hand) = Child objekt objektu TouchHandGrabInteractor.
- Hand Sphere Map - HandSphereMap (Hand Sphere Map) = Child objekt objektu TouchHandGrabInteractor.
- Hover Location - HoverLocation (Transform) = Child objekt objektu TouchHandGrabInteractor.

- Grab Location - Wrist (Transform) = Child objekt objektu TouchHandGrabInteractor.
- Min Hover Distance - 0.05.
- Curl Delta Threshold - 3.
- Curl Time Threshold - 0.05.
- Iterations - 10.

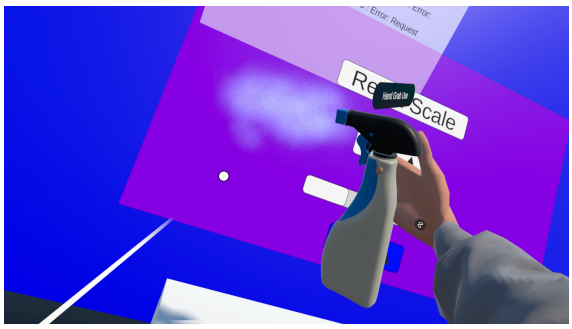
Komponent Touch Hand Grab Interactor Visual

- Interactor - TouchHandGrabInteractor (Touch Hand Grab Interactor).
- Synthetic Hand - (Right/Left)HandTouchGrabSynthetic (Synthetic Hand) = Child objekt objektu OVRHands.

2.7.7 HandGrabUseInteractor

Tento interaktor je špecializovaný interaktor, ktorý umožňuje používateľovi interagovať s objektmi pomocou ich "Use" interakcií. To znamená, že interaktor je určený pre prípady, keď používateľ nie len uchopí objekt, ale aj s daným objektom vykoná akciu. Ako príklad stlačenie spúšte alebo rôznych tlačidiel objektu, ktoré je možné stlačiť až pri uchopení. To umožňuje realistické zobrazovanie a monitorovanie sily, ktorá je použitá pre interakciu s konkrétnym objektom.

Obr. 2.44: Meta interaktor na úchop a použitie (úchop objektu)



Obr. 2.45: Meta interaktor na úchop a použitie (použitie objektu)

Konfigurácia objektu HandGrabUseInteractor v projekte je:

Komponent Hand Grab Use Interactor

- Max Interations Per Frame - 3.
- Use API - HandGrabUseInteractor (Use Finger Curl API).
- (optional)Hand - HandGrabUseInteractor (Hand Ref).
- (optional)Active State - HandGrabUseInteractor (Hand Ref).

Komponent Use Finger Curl API

- Hand - HandGrabUseInteractor (Hand Ref).

Komponent Secondary Interactor Connection

- Primary Interactor - HandGrabInteractor (Hand Grab Interactor).
- Secondary Interactor - HandGrabUseInteractor (Hand Grab Use Interactor).

2.8 Meta XR Virtual Interactables

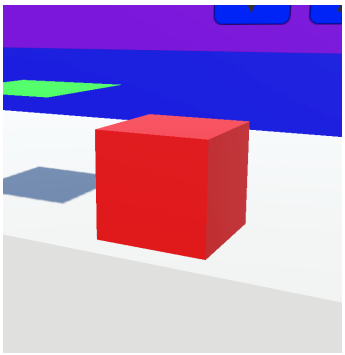
Meta XR all-in-one SDK ponúka mnoho možností interakcií. Objekty pre každú interactable môžu byť ľubovoľné. Kocka je použitá ako základ implementácie funkcionality. Medzi interactables Mety, ktoré sú implementované v projekte patria:

2.8.1 Grabbable

Ako prvé je vložený do scény 3D objekt kocky, ktorý obsahuje Box Collider a Mesh Renderer. Scale tohto objektu je zmenený na x = 0.08, y = 0.08, z = 0.08.

Na tento objekt je pripojený komponent Grabbable. Následne je pripojený na objekt komponent Grab Interractable. Ako parameter Pointable Element v komponente Grab Interactable je použitý komponent Grabbable. Ako ďalší je pridaný komponent Rigidbody. Tento komponent je vložený do parametra Rigidbody pre komponent Grab Interactable. Rigidbody komponent v časti Options obsahuje parameter Physics Grabbable. Pre tento paramter je pridaný na objekt komponent Physics Grabbable. Physics Grabbable je následne vložený do tohto Rigidbody Physics Grabbable parametra.

Takto je zabezpečená interakcia pomocou controllerov. Avšak nie je možné interagovať s objektom pomocou rúk. Na zabezpečenie interackie aj pomocou rúk je pre objekt pridaný komponent Hand Grab Interactable. Takto je zabezpečená kompletná interakcia s objektom aj pomocou rúk aj pomocou controllerov. Názov objektu v HTtestScene je Cube Grabbable.

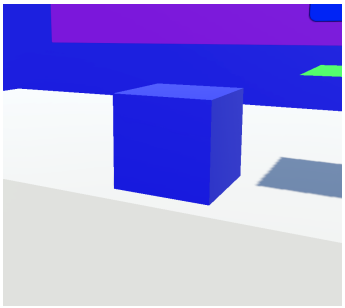


Obr. 2.46: Meta priamo uchopiteľný objekt (HTtestScene)

2.8.2 Distance Grabbable

Tento typ interactable je vytvorený nasledovne. Základnú funkcionalitu má rovnakú ako Grabbable, takže vytvorený Grabbable objekt je skopírovaný a na tomto objekte sú vykonané nasledovné úpravy. Pre objekt je vložený ďalší komponent Distance Grab Interactable a do parametra Point Element je vložený už vytvorený komponent Grabbable. To zabezpečí interakciu s controllermi.

Ako ďalší je objektu pridaný komponent Distance Hand Grab Interactable. To zabezpečí interakciu pomocou rúk. Takýmto spôsobom je vytvorený objekt, ktorý podporuje interakciu pomocou ovládačov aj rúk na blízko aj na diaľku. Názov objektu v HTtestScene je Cube Distance Grabbable.



Obr. 2.47: Meta vzdialene uchopiteľný objekt (HTtestScene)

2.8.3 Poke Interactable

Tento objekt je vytvorený tak, že do scény je pridaný empty objekt. Tento objekt je premenovaný na Poke Interactable. Pre Poke Interactable je pridaný child objekt Quad. Scale Quad objektu je x = 0.1, y = 0.1, z = 0.1. Rotácia objektu Poke Interactable je x = 90, y = 0, z = 0.

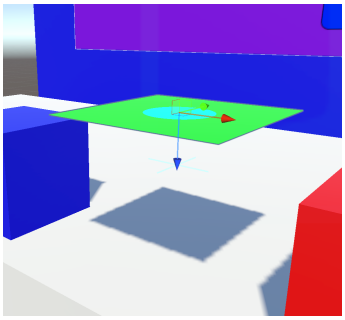
Pre Poke Interactable je následne pridaný komponent Poke Interactable. Tento komponent potrebuje aj parameter Surface Patch. Preto je v ďalšom kroku vytvorený empty child objekt pre objekt Poke Interactable a je pomenovaný ako Surface. Surface objektu je pridaný komponent Plane Surface. Tomuto povrchu je potrebné zmeniť veľkosť. To je zabezpečené pridaním komponentu Clipped Plane Surface na objekt Surface.

V Clipped Plane Surface je následne pridaný ako parameter komponent Plane Surface, ktorý je vytvorený pre objekt Surface. Ako ďalší parameter potrebuje Clipped Plane Surface objekt do listu Clippers. To je zabezpečené pridaním komponenu Bounds Clipper. Komponentu Bound Clipper je následne zmenená veľkosť na x = 0.1, y = 0.1, z = 0.01. Tento Bound Clipper je vložený ako objekt listu Clippers pre komponent Clipped Plane Surface.

Následne je pre objekt Poke Interactable v komponente Poke Interactable ako parameter Surface Patch zvolený komponent Surface. Po týchto krokoch je v editore nad povrchom objektu viditeľný kruh a kríž, ktoré označujú začiatok a koniec stláčania tlačidla. Prisbôsobíme Quad objekt tak, aby bol vo výške ako kruh ciže na začiatku.

Pre Quad objekt je pridaný komponent Poke Interactable. Ako Poke Interactable parameter je pridaný parent objekt Poke Interactable. Pre parameter Button Base Transfrom je použitý objekt Surface.

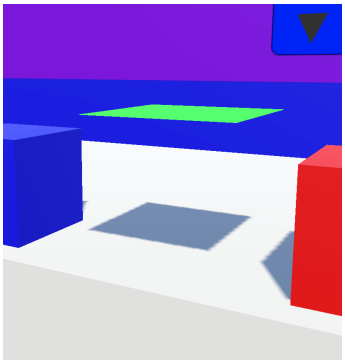
Týmto postupom je zabezpečené stláčanie objektu ako tlačidla pomocou rúk aj ovládačov. Pri stlačení zhora sa objekt začne pohybovať a keď je na úrovni konca stláčania objekt sa zastaví.



Obr. 2.48: Meta dotykový objekt Kruh a Kríž (HTtestScene)

Pre zabezpečenie funkcionality Unity Eventov je použitý nasledovný postup. Objektu Poke Insteractable je pridaný komponent Interactable Unity Event Wrapper. Ako Interactable View parameter pre tento komponent je zvolený komponent Poke Interactable rovnakého objektu. Následne je vytvorený event pri When Select (). Ako objekt eventu je pridaný objekt Cube Grabbable. Je vybraná funkcia `MeshRenderer.material`. Ako materiál je zvolený Blue_M. Pri evente When Unselect () je použitá rovnaká funkcia rovnakého objektu, ale je zvolený základný materiál Red_M, ktorý bol na objekte aj pred začiatkom eventu.

Takýmto postupom je pri stlačení tohto Poke Interactable objektu zabezpečené, že Cube Grabbable zmení farbu na modrú a po odkliknutí objekt zmení farbu späť na červenú.



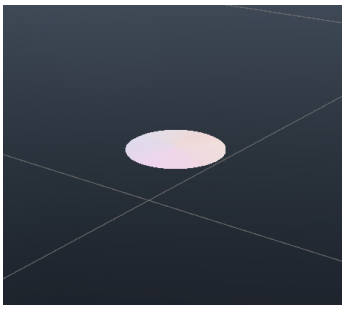
Obr. 2.49: Meta dotykový objekt (HTtestScene)

2.8.4 Teleport Hotspot a Teleport Interactable

Tento objekt je prefab Meta XR. Je uložený v priečinku Packages/Meta XR Interaction SDK Essentials/Runtime/Animations/Locomotion pod názvom Teleport-Hotspot. Zabezpečuje funkcionalitu teleportácie na miesto, na ktorom sa hotspot nachádza. V HTtestScene je hotspot umiestnený pred a na každom boku stola.

Podlaha je takisto nastavená ako teleportovacie miesto. Ak by pre podlahu nebol povolený teleport, dalo by sa teleportovať iba na hotspoty. Tak je možné nastaviť iba konkrétne miesta, na ktoré sa môže používateľ v scéne teleportovať.

Teleport na ľubovoľné miesto podlahy je zabezpečený tak, že je na Plane objekt pridaný komponent Teleport Interactable. Pre tento komponent je potrebný parameter Surface. Preto je pridaný na danú plane ďalší komponent Collider Surface. Ako Collider pre tento komponent je pridaný Mesh Collider Plane objektu. Následne je pre parameter Surface komponentu Teleport Interactable pridaný tento Collider Surface komponent. Pre Plane objekt je pridaný aj komponent Reticle Data Teleport. V Teleport Interactable komponente je tento Reticle Data Teleport vložený pre parameter Data v časti Optionals. Týmto postupom je zabezpečený teleport aj pre Teleport Hotspoty aj pre Plane teda podlahu scény.



Obr. 2.50: Meta teleportačný bod (HTtestScene)

2.8.5 Meta XR Interakcia s Canvasom (Ray Interactable)

V HTtestScene je vytvorené nové jednoduché menu pre interakciu cez Meta XR. V tomto menu sú vložené buttons cez UI - Button - TextMeshPro a UI - Button - Scrollbar. Pre interakciu s týmito buttonmi je na Canvas pridaný komponent Pointable Canvas. Ako paramter Canvas v tomto komponente je vložený Canvas objekt. Pre možnosť interakcie s Raycastmi je potrebné pridať na Canvas objekt aj komponent Ray Interactable. Ako parameter Pointable Element tohto komponentu je pridaný komponent Pointable Canvas. Ako ďalší parameter je potrebný parameter Surface. To je zabezpečené následným postupom. Je pridaný empty objekt ako child objekt objektu Canvas. Tento objekt je premenovaný na Surface.

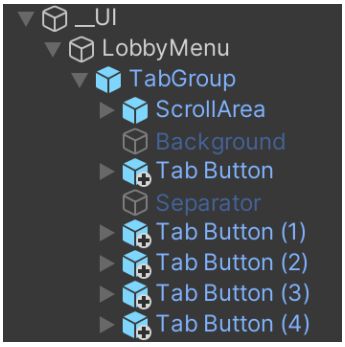
Na tento objekt je pridaný komponent Plane Surface. Následne ako ďalší komponent je pridaný Clipped Plane Surface. Ako Plane Surface parameter je priradený vytvorený komponent Plane Surface. Ďalej je pridaný na Surface objekt komponent Bounds Clipper. Size v tomto komponente je upravená na x = 1920, y = 1080, z = 1. Tento Bounds Clipper je priradený ako element pre list Clipped Plane Surface.

Následne je možné pridať v komponente Ray Interactable objektu Canvas parameter Surface a teda vytvorený a nakonfigurovaný Surface. Pri drag and drop priradení vyskočí okno s viacero možnosťami. Tu je zvolená možnosť ClippedPlaneSurface.

Ešte je potrebné v Event systéme vymazanie komponentu Standalone Input Module a priradenie komponentu Pointable Canvas Module. Týmto postupom je zabezpečená interakcia s menu pomocou Meta XR SDK. Je možné interagovať s menu aj pomocou rúk aj pomocou controllerov.

Úprava hlavného menu pre kompatibilitu s Meta XR

Hlavné menu __UI je vložené do HTtestScene. Po preskúmaní je zistené, že je náročné s Meta XR ovládaním prepínať medzi komponentmi hlavného menu, ktoré sú umiestnené pod objektmi ScrollArea v Scrollbaroch v Sliding Area. Preto sú tieto objekty presunuté z týchto miest na rovnakú hirerarchickú úroveň ako objekty ScrollArea pre dané časti menu. Príkladom sú objekty hlavných buttonov Main (Tap Button (1)), Controls (Tap Button (2)), Avatar (Tap Button (3)), Settings (Tap Button (4)) na obrázku 2.51. Pre menu je ešte potrebné pridanie a konfigurácia komponentov ako pri postupe zo sekcie Meta XR Interakcia s Canvasom (Ray Interactable).



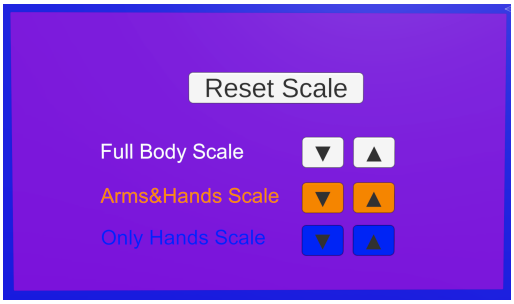
Obr. 2.51: Časť upravenej hierarchie hlavného menu

2.9 Vylepšenie scalera

V projekte už je implementovaný základný scalovací mechanizmus. Pri spustení aplikácie systém nastaví mierku avatara podľa reálnej výšky používateľa. Následne je možné mierku prepočítať. V hlavnom menu v sekcii Avatar sa nachádza tlačidlo Reset Scale. Po kliknutí na toto tlačidlo sa znova podľa výšky helmy od zeme prepočíta a priradí mierku pre avatara.

Implementované vylepšenie dopĺňa túto funkcionalitu o manuálne zväčšovanie a zmenšovanie scalu. Používateľ si tak môže detailnejšie upraviť mierku častí tela avatara. Implementované sú tri základné možnosti scalovania. Všetky tri možnosti možno nezávisle od seba ovládať šípkou hore a dole. Tieto šípky zväčšujú a zmenšujú mierku častí, ku ktorým sú priradené. Ako prvá možnosť je možnosť scalovania celého avatara. Šípkami sa scaluje veľkosť kompletne všetkých časti tela avatara.

Ako druhá možnosť je scalovanie ramien a zároveň dlaní. To je užitočné pre detailnejšie scalovanie rúk, ak majú časti tela nesprávnu mierku. Ako tretia možnosť je scalovanie iba dlaní. Táto možnosť po scalovaní ramien spolu s dlaňami pridáva ešte detailnejšie scalovanie dlaní bez scalovania ramien.



Obr. 2.52: Menu upravených scalerov

Pre implementáciu tohto riešenia sú upravené nasledovné skripty:

2.9.1 AvatarMenuManager.cs

V tomto skripte sú pridané tri funkcie. Každá slúži pre konkrétny scaler. Tento skript slúži ako prechod medzi tlačidlami scalera a hlavným skriptom na scalovanie. V každej funkcii sa zavolá funkcia z AvatarController.cs, ktorá priamo scaluje časti avatara. Tieto funkcie prijímajú mierku ako parameter. Tento parameter je hodnota, ktorá sa priráta k nastavenému scalu. To znamená, že ak sa priráta kladná hodnota, mierka sa zväčší a opačne ak sa priráta záporná tak sa

mierka zmenší. Vo funckcii ChangeScale, ktorá scaluje celého avatara je vložené aj prepojenie na AvatarSettings.

```
public void ChangeScale(float scale){
  avatarSettings.SetAvatarScale(scale);
  avatarController.ChangeSizeMulti(scale);
}

public void ChangeHandScale(float scale){
  avatarController.ScaleHands(scale);
}

public void ChangeArmScale(float scale){
  avatarController.ScaleArms(scale);
}
```

Zdrojový kód 2.3: Funkcie skriptu Avatar Menu Manager

2.9.2 AvatarSettings.cs

V tomto skripte je pridaná funkcia `SetAvatarScale()` (zdrojový kód 2.4), ktorá pripočíta hodnotu, o ktorú sa avatar zmení a následne vyvolá udalosť OnAvatarChange, ktorá informuje ostatné časti systému o zmene veľkosti avatara.

```
public void SetAvatarScale(float scale){
  sizeMultiplier += scale;
  OnAvatarChange?.Invoke();
}
```

Zdrojový kód 2.4: Funkcia `SetAvatarScale()` triedy AvatarSettings

2.9.3 AvatarController.cs

V tomto skripte sú vytvorené funkcie, ktoré priamo scalujú časti avatara. Funkcia `ChangeSizeMulti()` (zdrojový kód 2.5) upravuje celkovú veľkosť avatara. Pripočíta zmenu mierky k premennej `sizeMultiplier`. Ak je výsledná mierka väčšia ako 1.5 alebo menšia ako 0.5 tieto hodnoty nie sú presiahnuté a stávajú sa maximom pre daný smer scalovania. Následne sa na avatar aplikuje nova mierka a prepočíta sa IK avatara, aby boli všetky časti v súlade s novou veľkosťou.

Funkcia `RecalculateVRIK()` (zdrojový kód 2.6) počká 1 frame a následne vykoná prepočet IK pomocou vrik.solver. Tento krok je potrebný pri zmene celkovej veľkosti avatara aby sa avatar prispôsobil novému škálovaniu.

```
public void ChangeSizeMulti(float scale){
  sizeMultiplier += scale;
  if (sizeMultiplier < 0.5f){
    sizeMultiplier = 0.5f;
  }
  else if (sizeMultiplier > 0.5f){
    sizeMultiplier = 0.5f;
  }
  vrik.references.root.localScale = Vector3.one * sizeMultiplier;
  StartCoroutine(RecalculateVRIK());
}
```

Zdrojový kód 2.5: Funkcia `ChangeSizeMulti()` triedy AvatarContoller

```
public IEnumerator RecalculateVRIK(){
  yield return null;
  if (vrik != null && vrik.solver != null){
    vrik.solver.FixTransforms();
    vrik.solver.Update();
  }
}
```

Zdrojový kód 2.6: Funkcia `RecalculateVRIK()` triedy AvatarContoller

Funkcie `ScaleHands()` (zdrojový kód 2.7) a `ScaleArms()` (zdrojový kód 2.8) vykonávajú zmenu scaling (ScaleHands) a ramien s dlaňami (ScaleArms). Po stlačení tlačidiel v menu sa príslušné hodnoty mierky pripočítajú k aktuálnemu škálovaniu. Ak sa nenašli požadované referencie pre ruky alebo ramená scalovanie sa nevykoná. Funkcie scalujú obe dlane a ramená.

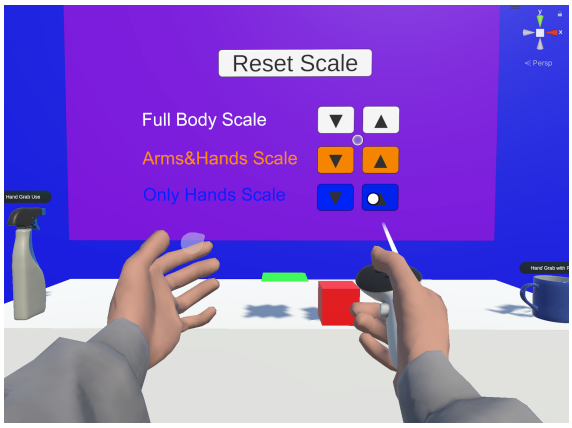
```
public void ScaleHands(float scaleIncrement){
  if (vrik == null || vrik.referces == null){
    return;
  }
  Transform leftHand = vrik.references.leftHand;
  Transform rightHand = vrik.references.rightHand;
  if (leftHand != null && rightHand != null){
    float newScaleLeft = leftHand.localScale.x + scaleIncrement;
    float newScaleRight = rightHand.localScale.x + scaleIncrement;
    leftHand.localScale = new Vector3(newScaleLeft, newScaleLeft, newScaleLeft);
    rightHand.localScale = new Vector3(newScaleRight, newScaleRight, newScaleRight);
  }
}
```

Zdrojový kód 2.7: Funkcia `ScaleHands()` triedy AvatarController

```
public void ScaleArms(float scaleIncrement){
  if (vrik == null || vrik.references == null){
    return;
  }
  Transform leftUpperArm = vrik.references.leftUpperArm;
  Transform rightUpperArm = vrik.references.rightUpperArm;
  if (leftUpperArm != null && rUpArm != null){
    float newScaleLeft = leftUpperArm.localScale.x + scaleIncrement;
    float newScaleRight = rightUpperArm.localScale.x + scaleIncrement;
    leftUpperArm.localScale = new Vector3(newScaleLeft, newScaleLeft, newScaleLeft);
    rightUpperArm.localScale = new Vector3(newScaleRight, newScaleRight, newScaleRight);
  }
}
```

Zdrojový kód 2.8: Funkcia `ScaleArms()` triedy AvatarController

V scéne HTtestScene sa nachádza konkrétne menu pre scalovanie, no funkcie sú navrhnuté tak, aby sa mohli používať z ľubovoľného menu, ktoré ma nakonfigurované tlačidlá pre pridanie a odobratie mierky.



Obr. 2.53: Zmena mierky veľkosti dlaní

2.10 Mirror networking pre hand tracking

Pre zabezpečenie správneho zobrazenia pre všetkých pripojených používateľov na serveri v Main Scene je potrebné zabezpečenie správneho networkingu. V projekte je použitý Mirror Network verzia 72.0.0. Pre controllery a prvky, ktoré už boli implementované v projekte pred touto prácou, už networking vyriešený je. Aby na serveri správne fungoval vylepšený XRRig, je potrebné upraviť niektoré handtracking objekty na avatarovi a netwroking skripty.

2.10.1 Úprava XR Rigu pre networking

V projekte sa pre network riešenie používa upravená kópia XR Rigu, konkrétne prefab RigOnlineXR. V tomto postupe sú rigu zmenené objekty ovládania rúk. Pôvodne Left Hand a Right Hand z prefabu XR Origin Hands sú nahradené samostatnými prefabmi Left Hand Tracking a Right Hand Tracking z balíka XR Hands. Prefaby poskytujú rovnakú funkcionalitu, no obsahujú menej komponentov, ktoré sú rovnako vhodné pre riešenie. Tieto prefaby sú umiestnené na rovnakom mieste na rigu ako predošlé objekty pre ľavú a pravú ruku a je im zmenený materiál na mierne transparentný, tmavosivej farby. V komponente XR Character Manager je potrebné všetky komponenty týchto objektov pridať do poľa Components To Enable Locally. Tieto objekty zobrazujú virtuálne ruky používateľa najprecíznejšie. V ďalšom kroku je každej ruke pridaný prefab NearFarInteractor z balíka XR Interaction Toolkit. Tento interaktor je vložený ako child objekt Left Hand a Right hand, ktoré sú child objekty daných prefabov. Pre správnu konfiguráciu je potrebné na rovnakej úrovni pre každú ruku vytvoriť objekty PinchPosition a AimPosition. Týmto objektom sú pridané komponenty Tracked Pose Driver (Input System). Tieto komponenty sú nakonfigurované nasledovne:

PinchPosition

- Position Input - XRI (Left/Right)/Pinch Position (Input Action Reference).
- Rotation Input - XRI (Left/Right)/Rotation (Input Action Reference).
- Tracking State Input - XRI (Left/Right)/Tracking State (Input Action Reference).

AimPosition

- Position Input - XRI (Left/Right)/Aim Position (Input Action Reference).
- Rotation Input - XRI (Left/Right)/Aim Rotation (Input Action Reference).
- Tracking State Input - XRI (Left/Right)/Tracking State (Input Action Reference).

Tieto objekty sú následne pridané v komponentoch NearFarInteractora každej ruky:

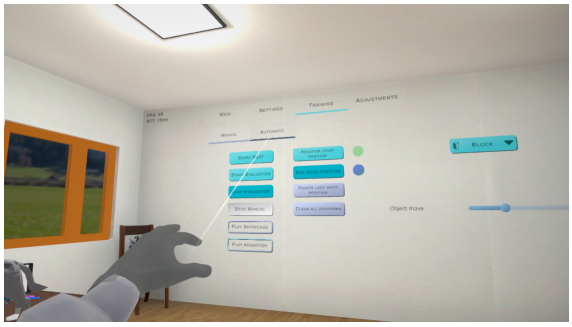
- Interaction Attach Controller -> Transorm To Follow - PinchPosition.
- Sphere Interaction Caster -> Cast Origin - PinchPosition.

- Curve Interaction Caster -> Cast Origin AimPosition.
- Objekt LineVisual (Child objekt interaktora) - Curve Visual Controller -> Line Origin Transform - AimPosition.

Toto nastavenie zabezpečí správnu polohu a rotáciu interaktorov v scéne pre každého používateľa. Interaktory tak sledujú polohu a rotáciu dlaní používateľa a otáčajú sa spoločne s rukami používateľa. Pre správnu konfiguráciu interakcií interaktora je potrebné nastaviť pre každý interaktor v komponente Near-Far Interactor tieto parametre:

- UI Press Input - XRI (Left/Right) Interaction/UI Press, XRI (Left/Right) Interaction/UI Value.
- UI Scroll Input - XRI (Left/Right) Interaction/UI Scroll.
- Select Input - XRI (Left/Right) Interaction/Select, XRI (Left/Right) Interaction/Select Value.
- Activate Input - XRI (Left/Right) Interaction/Activate, XRI (Left/Right) Interaction/Activate Value.

Následne je potrebné nastavenie parametrov Physics Layer Mask (v komponente Spehere Interaction Caster) a Raycast Mask (v komponente Curve Interaction Caster) na hodnoty Defalut, UI a Target. Ako ďalší krok je potrebné na objekte RigOnlineXR v komponente Input Action Manager pridanie nového elementu XRI Default Input Actions (Input Action Asset) z balíka XR Interaction Toolkit 3.0.6. V komponente XR Character Manager je potrebné do poľa Objects To Disable pridanie objektov interaktorov spolu s pinch a aim position objektmi. Do poľa Components To Enable Locally je potrebné pridanie RigOnlineXR (Input Action Manager).



Obr. 2.54: XRI Interakcia lúču interaktora s hlavným menu

2.10.2 XRDragAndDrop.cs

Tento skript je upravený tak, aby obsahoval pole `[SerializeField] private NearFarInteractor[] xrHandInteractors`, do ktorého sú následne NearFar interaktory pravej a ľavej ruky pridané. Vo funkciách skriptu `onEnable()` a `onDisable()` je upravená logika tak, aby vykonala funkcie `AddListener` a `RemoveListener` aj pre NearFar interaktory.

2.10.3 _Enums.cs

Do `enum ControllerType` je pridaný nový typ ovládača - HandTracking. Je nakonfigurovaný tak, že sa v menu neukazuje, ale je možné sa z každého ovládača naň prepnúť pomocou aktivovania hand trackingu na HMD zariadení.

2.10.4 XRIKHandTracking.cs

V tomto skripte je vytvorená funkcia `Switch()`, ktorá prepína IK ciele pre zápästia avatara medzi sledovaním ovládačov a sledovaním rúk. Tento skript je odvodený od skriptu HandTrackingSwitch.cs, ktorý zabezpečuje túto funkcionalitu pre offline režim. Je navrhnutý tak, že miesto sledovania globálneho stavu ovládačov, berie ako parameter aktuálny typ ovládača z poľa `ControllerType` zo súboru _Enums.cs, čo zabezpečuje jeho funkcionalitu v online režime.

2.10.5 CharacterManager.cs

V tomto skripte sú pridané nasledovné polia:

- `private GameObject[] leftHandAvatarModels` a `rightHandAvatarModels` - Obsahujú objekty modelov dlaní avatara.
- `private GameObject[] objectsToDisable` - Obsahuje objekty, ktoré sa majú deaktivovať mimo lokálneho prostredia (observers) = Objekty hand interaktorov.

Tieto polia sú použité v nasledovných funkciách:

- `public void SetLeftHandAvatarModelActive(bool active)` - Aktivuje alebo deaktivuje objekty modelov dlaní avatara definované v poli leftHandAvatarModels.
- `public void SetRightHandAvatarModelActive(bool active)` - Aktivuje alebo deaktivuje objekty modelov dlaní avatara definované v poli rightHandAvatarModels.

2.10.6 XRCharacterManager.cs

V tomto skripte sú pridané nasledovné premenné:

- `private ControllerType previousControllerType` - Obsahuje predchádzajúci zvolený typ ovládača.
- `private XRHandTrackingEvents[] XRHandTrackingControllers` - Obsahuje komponenty ovládania XR-Hand = Right a Left Hand Tracking objekty.
- `private XRIKHandTracking[] XRIKHandTracking` - Obsahuje komponenty XRIKHandTracking, ktoré zabezpečujú prepínanie IK cieľov pre ruky.
- `private ControllerType _localControllerType` - Obsahuje aktuálny typ ovládača lokálneho používateľa.

Tieto premenné sú použité v nasledovných funkciách:

- `private bool IsControllerTracking(XRNode controllerNode)` - Kontroluje, či sú ovládače aktívne.
- `private void Update()` - Na základe stavu ovládačov mení typ ovládača. Obsahuje špeciálnu podmienku pre prepínanie na hand tracking.
- `private void CmdChangeControllerType(ControllerType controllerType)` - Mení typ ovládača pre všetkých používateľov. Je to serverová funkcia.
- `public void ChangeControllerType(ControllerType _old, ControllerType _new)` - V tejto funkcii je doplnená podmienka pre prepínanie na hand tracking, pri ktorom sa vypnú modely controllerov a modely rúk avatara.
- `public override void ChangeAnimatedArm(bool _old, bool _new)` - Doplnená logika pre ruky na základe existujúcej funkcie.
- `public override void FixArms(bool _old, bool _new)` - Doplnená logika pre ruky na základe existujúcej funkcie.

2.10.7 HandTrackingNetworkSync.cs

Tento skript je v projekte nový. Zabezpečuje synchronizáciu pozície a rotácie rúk avatara medzi používateľmi. Je vložený dvakrát ako komponent pre RigOnlineXR a pripojený na L_Wrist a R_Wrist avatara. Používa Unreliable channel pre synchronizáciu (UDP).

V tomto skripte sú pridané nasledovné premenné:

- `private Transform handTrackingRoot` - Obsahuje root objekt rúk avatara.
- `private Transform[] handTrackingObjects` - Obsahuje objekty kostí rúk avatara. Inicializuje sa dynamicky vo funkcii Awake.
- `private Vector3[] positions` - Obsahuje pozície kosti rúk avatara.
- `private Quaternion[] rotations` - Obsahuje rotácie kosti rúk avatara.

Tieto premenné sú použité v nasledovných funkciách:

- `private void Awake()` - Inicializuje polia handTrackingObjects, positions a rotations.
- `private void CmdUpdateTransforms(Vector3[] positions, Quaternion[] rotations)` - Synchronizuje pozície a rotácie rúk avatara medzi používateľmi. Je to serverová funkcia.
- `private void RpcUpdateClients(Vector3[] positions, Quaternion[] rotations)` - Aktualizuje pozície a rotácie rúk avatara lokálneho používateľa. Je to client funkcia.
- `private void Update()` - Posiela pozície a rotácie rúk avatara lokálneho používateľa ostatným používateľom.

2.10.8 Pridanie NetworkTransform komponentov

Pre správne fungovanie synchronizácie na serveri sú pridané Network Transformy. Network Transformy používajú UDP protokol, ktorý zabezpečuje synchronizáciu v reálnom čase. Objekty rúk sú vždy na aktuálnej pozícii, no občas môže dôjsť k strate paketov. To môže spôsobiť krátkodobé zaseknutie rúk na pozícii, kým neprebehne ďalšia úspešná synchronizácia. Zoznam objektov, pre ktoré sú pridané a nakonfigurované komponenty NetworkTransform na objekte RigOnlineXR: L_Wrist, R_Wrist, L_Wrist_Target, R_Wrist_Target, VRIK_L_Target, VRIK_R_Target, Left Hand Tracking, Right Hand Tracking.

Obr. 2.55: Synchronizácia dlaní na serveri

Tieto postupy a úpravy skriptov zabezpečujú plnú funkcionalitu sycnrhonizácie a interakcie rúk na serveri. Používateľ vidí z prvej osoby svoje raycasty, no pre ostatných používateľov sú tieto raycasty skryté. Používatelia na serveri vidia synchronizované pohyby dlaní a prstov všetkých používateľov v reálnom čase, ktoré sú zároveň aj prepojené s ich avatarmi.

Obr. 2.56: Pohľad 1. osoby

Obr. 2.57: Pohľad 2. osoby

3 Vyhodnotenie riešenia

Počas testovania bola využívaná kombinácia zariadení Oculus - Oculus. Konkrétne sa na testovanie používali zariadenia Meta Quest 2 a Meta Quest 3. Špecifikácia týchto zariadení sa nachádza v tabuľke 3.1.

Parameter	Meta Quest 2	Meta Quest 3
CPU	Qualcomm XR2 Gen1	Qualcomm XR2 Gen2
GPU	Adreno 650 @ 587 MHz	Adreno 740 @ 599 MHz
RAM	6 GB LPDDR4X	8 GB LPDDR5
OS	Meta Horizon OS	Meta Horizon OS
Displej	2x LCD (RGB) 1832 x 1920	2× LCD (RGB) 2064×2208
Frekvencia	72 - 120 Hz	90 - 120 Hz
Hand tracking	Áno	Áno

Tabuľka 3.1: Špecifikácia Desktop PC zariadenia

Pre testovanie bolo s pripojením Meta Quest 3 používané aj Desktop PC zariadenie. Špecifikácia tohto zariadenia sa nachádza v nasledovnej tabuľke 3.2.

Parameter	Desktop PC
RAM	128 GB DDR5
CPU	13th Gen Intel(R) Core(TM) i9-13900K
GPU	NVIDIA GeForce RTX 4090
OS	Microsoft Windows 11 Pro
Základná doska	Z790 AORUS ELITE AX

Tabuľka 3.2: Špecifikácia zariadení Meta Quest 2 a Meta Quest 3

Aplikácia bola testovaná troma spôsobmi. Hlavným cieľom testovania bolo overiť a vyhodnotiť správnosť a presnosť funkcionality všetkých implementovaných vylepšení tejto práce. Tie zahŕňajú správne fungovanie rigov pri sledovaní rúk, interaktorov, interactables, scalera a networking riešenia pre serverovú časť.

Ďalšia časť testovania bola zameraná na technickú stránku, ktorá bola odmeraná počas používania aplikácie. Časť vyhodnotenia sa zameriava aj na porovnanie Open XR Rigu s Meta XR Rigom a na ich interaktory a interactables.

3.1 Testovanie Meta XR Rigu

Testovanie prebiehalo na zariadeniach po nahraní buildnutej aplikácie. Pre toto testovanie boli používané dve zariadenia (Meta Quest 3 a Meta Quest 2). Na každom zariadení bola aplikácia testovaná osobitne. Pri tomto testovaní nebolo potrebné vytvorenie a spustenie dedicated servera.

3.1.1 Testovanie funkčnosti

Cieľom bolo overiť funkčnosť aplikácie obsahujúcej Meta XR Rig a jeho interakcií v rámci singleplayer scény. Testované aspekty boli hlavne správanie sa rigu a rúk avatara, interakcie Meta interaktorov s interactables, scalovanie avatara pomocou všetkých typov scalerov a interakcia s upraveným hlavným menu aplikácie.

Postup testovania:

1. Spustenie scény HTtestScene pomocou tlačidla play v Unity Editore (zariadenie Meta Quest 3).
2. Overenie správnej detekcie, vizualizácie a prepínania medzi rukami a ovládačmi v reálnom čase.
3. Testovanie funkcionality gest pre Teleport a Turner interaktory.
4. Testovanie správania Teleport a Turner interaktorov.
5. Testovanie interakcie s každým typom Meta XR Interactable.
6. Testovanie každého scalera.
7. Testovanie interakcie s upraveným hlavným menu.
8. Kontrola synchronizácie avatara s rukami a ovládačmi (zrkadlo v scéne).

Po testovaní bolo zistené, že Avatar sa správal podľa očakávania. Scéna sa spustila pri správnej konfigurácii bez problémov. Sledovanie rúk a ovládačov bežalo bezchybne, rovnako ako aj prepínanie medzi nimi. Tento rig poskytoval aj hybridné ovládanie. To znamená, že používateľ mohol používať jeden ovládač a jednu ruku zároveň. Aj pri tomto ovládaní všetko prebehlo bezchybne.

Pri ukázaní správneho gesta pomocou ruky sa objavili príslušné Teleport a Turner interaktory (obrázky 2.38 a 2.39) a poskytli správnu funkcionalitu podľa konfigurácie. Teleport Interactor poskytol teleport aj na Teleport Hotstpoty a zároveň aj na ľubovoľné miesto na plane, ktorá je taktiež nastavená pre možnosť teleportovania sa na ľubovoľné miesto v rámci konkrétnej plane. Turner Interactor používaný pravou rukou otáčal avatara plynulo a ľavou rukou prichytením na určité uhly. Ray Interactor zabezpečil správnu funkcionalitu interakcie s menu scalerov aj s hlavným menu. Pri tejto interakcii boli otáčanie aj teleport vypnuté. Znova boli aktivované keď Ray Interactor nemieril na UI.

Testovanie Grab Interactable malo nasledovný výstup. Pri grab geste ak bola ruka dostatočne blízko sa Grab Interactable prichytila k ruke (obrázok 2.36). Keď bolo grab gesto zrušené Grab Interactable sa odpútala od ruky. Táto interakcia nefungovala na diaľku. Avšak to poskytovala ďalšia interactable - Distance Grab Interactable (obrázok 2.40). Táto vykonala rovnakú funkcionalitu, ale s možnosťou interakcie aj na diaľku. Pri stlačení Poke Interactable zhora (obrázok 2.34) sa posunula a Grab Interactable zmenila farbu na modrú. Po uvoľnení Poke Interactable sa jej farba zmenila naspäť na červenú (obrázok 2.35).

Pri testovaní každého scalera (obrázok 2.52) bol taktiež dosiahnutý očakávaný výstup. Príslušné časti alebo celý avatar sa scalovali podľa nastavenia cez šípky v menu scalerov. Pri interakcii s hlavným menu pomocou raycastov sa bolo možné prepínať bez problémov cez ľubovoľne zvolené tlačidlá.

Pri kontrole synchronizácie bolo zistené, že ruky avatara sledovali ovládače aj ruky používateľa a hýbali sa podľa nich, zatiaľ čo bola poloha a výzor avatara zachovaná v reálnom tvare (obrázok 2.32). Občas sa však stalo, že sa nejaká ruka posunula na chvíľu na náhodné miesto v okolí avatara. To bolo spôsobené tým, že zariadenie na chvíľu nevedelo rozoznať ani jeden aktívny vstup.

Avšak je potrebné dávať pozor na nastavenia v Settings pre Meta Quest zariadenia. V týchto nastaveniach je možnosť upraviť výšku pre používanie v sede. Toto nastavenie pri sedení poskytuje aplikácii rovnakú výšku ako zariadenie poskytuje s vypnutým nastavením pri státi. Toto nastavenie môže občas spôsobiť nesprávnu mierku avatara, ktorá nejde tak ľahko vyscalovať. Pri tomto probléme sa odporúča znova nastaviť typ nastavenia (sedenie, státie) a reštart zariadenia aj aplikácie. Nastavenie je možné nájsť v hlavnom menu pre Meta zariadenie Settings -> Accessibility (Scroll panel na ľavej strane) -> Mobility -> Adjust height.

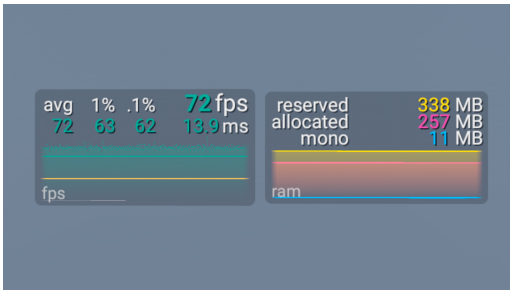
3.1.2 Testovanie výkonu

Cieľom tohto testovania bolo overiť výkon aplikácie. Pri tomto testovaní bol stanovený limit 5 minút, počas ktorého sa v aplikácii vykonávali rôzne akcie. To zabezpečilo, že testovanie výkonu sa nesústredilo iba na jeden scenár, ale na výstup výkonu rôznych funkcionalít aplikácie.

Meta Quest 3

Výstup pre zariadenie Meta Quest 3 bol nasledovný:

- Plynulosť pre čas vytvorenia snímku 3.1 - 72fps.
- Doba načítania framu pre čas vytvorenia snímku 3.1 - 13.9ms.
- Priemerné FPS - 72.
- Najnižšia hodnota 1% FPS - 63.
- Najnižšia hodnota .1% FPS - 62.
- Rezervovaná pamäť - 338 MB.
- Alokovaná pamäť - 257 MB.
- Mono pamäť - 11 MB.



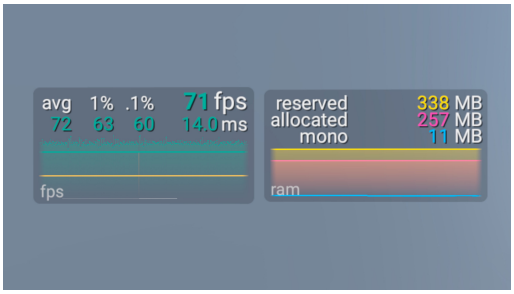
Obr. 3.1: Meta Quest 3 Meta XR Rig FPS a RAM Test

Meta Quest 2

Výstup pre zariadenie Meta Quest 2 bol nasledovný:

- Plynulosť pre čas vytvorenia snímku 3.2 - 71fps.
- Doba načítania framu pre čas vytvorenia snímku 3.2 - 14.0ms.

- Priemerné FPS - 72.
- Najnižšia hodnota 1% FPS - 63.
- Najnižšia hodnota .1% FPS - 60.
- Rezervovaná pamäť - 338 MB.
- Alokovaná pamäť - 257 MB.
- Mono pamäť - 11 MB.



Obr. 3.2: Meta Quest 2 Meta XR Rig FPS a RAM Test

3.2 Testovanie singleplayer Open XR Rigu

Testovanie prebiehalo na zariadeniach po nahraní buildnutej aplikácie. Pre toto testovanie boli používané dve zariadenia (Meta Quest 2 a Meta Quest 3). Na každom zariadení bola aplikácia testovaná osobitne. Pri tomto testovaní nebolo potrebné vytvorenie a spustenie dedicated servera.

3.2.1 Testovanie funkčnosti

Cieľom bolo overiť funčnosť Open XR Rigu a jeho interakcie v rámci singleplayer scény. Testované aspekty boli správanie sa rigu aj rúk avatara, vizualizácia a synchronizácia avatara s rukami a interakcie XRI interaktorov s interactables.

Postup testovania:

1. Overenie scény MenuScene (Prvá scéna, ktorá sa načíta pri zapnutí aplikácie).
2. Overenie správnej detekcie, vizualizácie a prepínania medzi rukami a ovládačmi v reálnom čase.

3. Testovanie Poke a NearFar interaktorov.

4. Testovanie interakcie s každým typom XRI Interactable a upravenými objektmi v projekte.

5. Kontrola synchronizácie avatara s rukami a ovládačmi (zrkadlo v scéne).

Pri zapnutí aplikácie sa avatar nainicializoval podľa očakávania, no v zopár prípadoch sa počas vývoja stalo, že sa avatar načítal s pohľadom zaseknutým v strope alebo jeho mierka voči miestosti nebola správna. Na to je už spomínané riešenie v sekcii 3.1 v tejto kapitole, kde je spomenutý postup nastavenia typu výšky na Meta Quest zariadení. Po spustení sa avatar nachádzal v MenuScene.

Pri zariadení Meta Quest 2 je odporúčané prepínať na ruky pomocou dvojitého klepnutia joystickov o seba. Po dvojitom klepnutí joystickov o seba sa zobrazili ruky (obrázok 2.12) a ich funkčnosť, detekcia a synchronizácia s avatarom bola správna. Na zariadení Meta Quest 3 je prepínanie medzi rukami a joystickmi vylepšené. Stačí len odložiť ovládače a ruky sa automaticky zobrazia. V niektorých prípadoch je to možné aj pre Meta Quest 2, ale nie je to až tak komfortné, ako pri Meta Quest 3. Pri tomto zariadení funguje rovnako aj dvojité klepnutie joystickov o seba, takže si používateľ môže zvoliť prepínanie podľa seba. Čo sa týka prechodu z rúk na ovládače, po úchope ovládačov zariadenia automaticky detekujú aktívne joysticky a ovládanie sa prepne späť.

Pri testovaní NearFar interaktora (obrázok 2.18) bola dosiahnutá očakávaná funcionalita. Po namierení na Grab Interactable (obrázok 2.19) sa konkrétny interaktor predĺžil smerom k tejto interactable a po namierení mimo sa interaktor skrátil smerom k ruke avatara. Pri mierení a následnom pinch geste na Grab Interactable sa objekt dostal do stavu úchopu. Pri pustení sa objekt od interaktora odviazal a spadol späť na plochu pod ním. Ak bol objekt uchopený interaktorom a následne bola ruka pritiahnutá k telu, objekt bol pritiahnutý do ruky. V tej chvíli interaktor zmizol a objekt bol zafixovaný na koniec ruky. Manipulácia bola zachovaná a po zrušení pinch gesta sa objekt odpútal od ruky. Čo sa týka upravenej interactable pohára (obrázok 2.22), ktorá bola v projekte, tá fungovala rovnako správne ako Grab Interactable.

Testovanie Poke Interactora (obrázky 2.17 a 2.16) poskytlo rovnako správne výsledky. Pri evente Poke Interactable (obrázok 2.21) Hover funkcionalita zmeny farby bola vykonaná a pri dostatočnej vzdialenosti od plochy sa farba prepla späť na základnú. Avšak treba si uvedomiť, že táto interakcia počíta aj interaktory, takže sa farba mení už pri dotyku interaktora. To znamená, že aj keď v blízkosti objektu interaktor zmizne, v priestore interaktor stále existuje, takže objekt ho

detekuje a zmení farbu pri dotyku miesta, na ktorom je interaktor skrytý. Počas celého testovania sa ruky avatara synchronizovali a pohybovali správne aj pri prepnutí na ruky aj pri ovládaní ovládačmi.

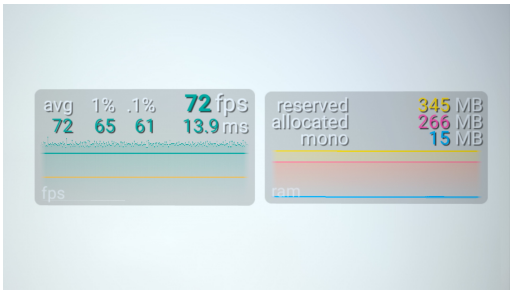
3.2.2 Testovanie výkonu

Cieľom tohto testovania bolo overiť výkon aplikácie. Pri tomto testovaní bol stanovený limit 5 minút, počas ktorého sa v aplikácii vykonávali rôzne akcie. To zabezpečilo, že testovanie výkonu sa nesústredilo iba na jeden scenár, ale na výstup výkonu rôznych funkcionalít aplikácie.

Meta Quest 3

Výstup pre zariadenie Meta Quest 3 bol nasledovný:

- Plynulosť pre čas vytvorenia snímku 3.3 - 72fps.
- Doba načítania framu pre čas vytvorenia snímku 3.3 - 13.9ms.
- Priemerné FPS - 72.
- Najnižšia hodnota 1% FPS - 65.
- Najnižšia hodnota .1% FPS - 61.
- Rezervovaná pamäť - 345 MB.
- Alokovaná pamäť - 266 MB.
- Mono pamäť - 15 MB.



Obr. 3.3: Meta Quest 3 Open XR Rig FPS a RAM Test

Meta Quest 2

Výstup pre zariadenie Meta Quest 2 bol nasledovný:

- Plynulosť pre čas vytvorenia snímku 3.4 - 71fps.
- Doba načítania framu pre čas vytvorenia snímku 3.4 - 14.1ms.
- Priemerné FPS - 72.
- Najnižšia hodnota 1% FPS - 62.
- Najnižšia hodnota .1% FPS - 59.
- Rezervovaná pamäť - 344 MB.
- Alokovaná pamäť - 265 MB.
- Mono pamäť - 15 MB.

Obr. 3.4: Meta Quest 2 Open XR Rig FPS a RAM Test

3.3 Testovanie multiplayer Open XR Rigu

Testovanie prebiehalo na zariadeniach po nahraní buildnutej aplikácie a spustení dedicated servera. Pre toto testovanie boli používané tri zariadenia (Meta Quest 2, Meta Quest 3 a Dekstop PC). Na každom zariadení bola aplikácia spustená a testovaná súčasne. Pri tomto testovaní bolo potrebné vytvorenie a spustenie dedicated servera (v tomto prípade na Desktop PC).

3.3.1 Testovanie funkčnosti

Cieľom bolo overiť funkčnosť Open XR Rigu, jeho interakcie v rámci multiplayer scény a vzájomná viditeľnosť všetkých používateľov pripojených na serveri. Ako prvé sa vykonal dedicated server build pre Desktop PC. Tento server bol spustený

v rovnakej sieti, v akej boli pripojené Meta Quest zariadenia. To sprístupnilo Meta Quest zariadeniam pripojenie na server. Následne sa spustila aplikácia na oboch zariadeniach naraz a zadali sa IP adresy pre pripojenie na server. Následne sa obe zariadenia pripojili na server.

Postup testovania:

1. Pripojenie používateľov na server.
2. Overenie vzájomnej viditeľnosti avatarov.
3. Overenie synchronizácie avatarov.
4. Overenie prepínania rúk používateľmi.
5. Overenie synchronizácie a vzájomnej viditeľnosti rúk aj pohybov prstov avatarov.
6. Overenie viditeľnosti NearFar interaktorov každého používateľa.
7. Overenie interakcie používateľov s objektmi na serveri.

Pri zapnutí aplikácie sa avatary nainicializovali podľa očakávania. Po spustení sa obaja používatelia s avatarmi nachádzali v MenuScene. Po kliknutí na input button pre server sa zobrazila klávesnica, každý používateľ zadal server a ten bol pridaný do listu. Obe zariadenia sa úspešne pripojili na server.

Používatelia videli navzájom svojich avatarov a rovnako aj ich pohyb a synchronizáciu. V nejakých prípadoch sa stalo, že avatarovi sa zle nainicializovali objekty rúk, a to znamenalo, že ruky smerovali do strany avatara alebo sa avatar zasekol v zemi. Avšak tento bug má jednoduché riešenie. Tým je odpojenie a následné pripojenie sa na server. Po prepojení sa to už neopakovalo.

Prepínanie medzi rukami a ovládačmi fungovalo u každého používateľa ako malo. Na Meta Quest 2 sa používateľ prepínal cez dvojité klepnutie ovládačov a na Meta Quest 3 kombinovane. Ovládače boli každému používateľovi osobitne podľa jeho nastavenia prepnuté na ruky a opačne ruky na ovládače. Ruky avatarov vždy správne sledovali polohu ovládačov aj rúk a pohybovali sa správne.

Používatelia si navzájom videli správne pohyby rúk aj prstov (obrázok 2.55). Občas sa stalo, že sa ruky alebo ovládače na veľmi krátku chvíľu zasekli, no to bolo spôsobené pripojením. Po tejto chvíli sa hneď ruky a ovládače späť zosynchronizovali. Táto krátka chvíľa netrvala viac než pár framov.

Overenie viditeľnosti NearFar interaktorov bolo rovnako správne (obrázky 2.56 a 2.57). Každý používateľ videl NearFar interactory pre svoje ruky a nevi-

del NearFar interaktory rúk druhého používateľa. Interakcie týchto NearFar interaktorov boli tiež správne. Ak jeden používateľ uchopil objekt pomocou svojho interaktora, prevzal nad ním autoritu a druhý používateľ objekt uchopiť nevedel. Keď používateľ objekt pustil, autorita bola zrušená a druhý používateľ mohol objekt uchopiť.

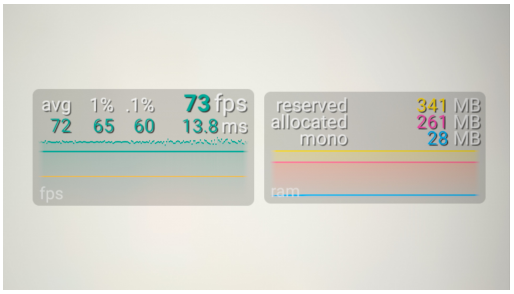
3.3.2 Testovanie výkonu

Cieľom tohto testovania bolo overiť výkon aplikácie. Pri tomto testovaní bol stanovený limit 5 minút, počas ktorého sa v aplikácii vykonávali rôzne akcie. To zabezpečilo, že testovanie výkonu sa nesústredilo iba na jeden scenár, ale na výstup výkonu rôznych funkcionalít aplikácie.

Meta Quest 3

Výstup pre zariadenie Meta Quest 3 bol nasledovný:

- Plynulosť pre čas vytvorenia snímku 3.5 - 73fps.
- Doba načítania framu pre čas vytvorenia snímku 3.5 - 13.8ms.
- Priemerné FPS - 72.
- Najnižšia hodnota 1% FPS - 65.
- Najnižšia hodnota .1% FPS - 60.
- Rezervovaná pamäť - 341 MB.
- Alokovaná pamäť - 261 MB.
- Mono pamäť - 28 MB.

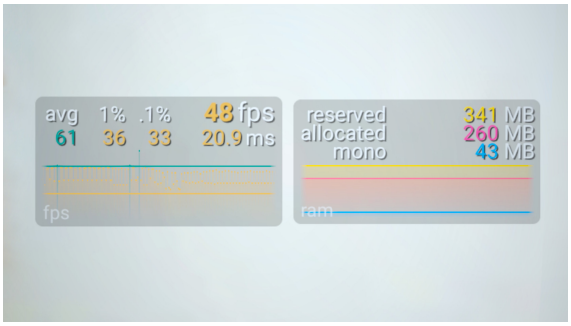


Obr. 3.5: Meta Quest 3 Open XR Rig server FPS a RAM Test

Meta Quest 2

Výstup pre zariadenie Meta Quest 2 bol nasledovný:

- Plynulosť pre čas vytvorenia snímku 3.6 - 48fps.
- Doba načítania framu pre čas vytvorenia snímku 3.6 - 20.9ms.
- Priemerné FPS - 61.
- Najnižšia hodnota 1% FPS - 36.
- Najnižšia hodnota .1% FPS - 33.
- Rezervovaná pamäť - 341 MB.
- Alokovaná pamäť - 260 MB.
- Mono pamäť - 43 MB.



Obr. 3.6: Meta Quest 2 Open XR Rig server FPS a RAM Test

3.4 Vyhodnotenie vykonaných testovaní

Z vykonaných testovaní funkčnosti vyplýva, že všetky typy Rigov spĺňajú očakávané požiadavky na oboch zariadeniach (Meta Quest 2 a Meta Quest 3) ako osobitne, tak aj spoločne. Čo sa týka výkonu, podľa výsledkov priemerná hodnota FPS pri všetkých testovaniach (s výnimkou Meta Quest 2 Server Rig testu) bola 72. Ak nepočítame spomínaný test, pri ktorom hodnoty fps boli nižšie, Meta Quest 3 poskytuje vo väčšine prípadov mierne lepší výkon. Avšak aj Meta Quest 2 poskytuje takmer rovnako plynulý výkon. Pri teste Open XR Rigu na serveri je možné, že fps výsledky Meta Quest 2 zariadenia klesli preto, lebo prenos každej kosti prsta môže byť pre toto zariadenie náročnejší ako pre Meta Quest 3, ktoré poskytlo takmer rovnaké výsledky ako pre offline, tak aj pre serverový test.

3.5 Vyhodnotenie sieťového prenosu

Pri implementácii bolo pridaných 8 Network Transformov, ktoré sú nakonfigurované rovnako ako už existujúce Network Transformy v projekte. Tieto Network Transformy zabezpečujú prenos údajov pre virtuálne ruky, zápästia a sledované cieľové body zápästí a ovládačov. Tieto tansformy posielajú rotáciu a pozíciu každého objektu, pre ktorý je Network Transform nakonfigurovaný. To znamená 7 float hodnôt pre jeden objekt. V tomto prípade je to 8-krát 7 float hodnôt čo sa rovná 56 float hodnotám. 1 float má 4 bajty. To sa rovná hodnote 0,224kB. Pri aktuálnej send rate hodnote 45Hz to môže navýšiť prenos o približne 10,08kB/s pre jedného používateľa. Pri prijímaní hodnôt klientmi platí vzorec p*10,08kB/s, pričom p označuje počet nelokálnych klientov voči odoslaným dátam.

Pre synchronizáciu prstov bol vytvorený nový skript, ktorý používa na prenos údajov Cmd a Rpc funkcie. Funkcia `CmdUpdateFransforms()` posiela údaje používateľa na server. Funkcia `RPCUpdateClients()` synchronizuje údaje s ostatnými klientmi. Úlohou a výhodou tohto skriptu je možnosť prenosu údajov pre všetky prsty rúk naraz. Každá kosť ruky má pozíciu a rotáciu, čo znamená 7 float hodnôt pre každú kosť. Jedna ruka obsahuje 25 kostí plus 1 cieľový bod. To znamená 26-krát 7 float hodnôt. Vo výsledku to je pre jednu ruku 182 float hodnôt pri každom prenose. Pre dve ruky to je 364 float hodnôt. 1 float má 4 bajty. To sa rovná hodnote 1,456kB na jeden Update pri aktívnom hand trackingu. Pri aktuálnej send rate hodnote 45Hz to môže navýšiť prenos o približne 65,52kB/s pre jedného používateľa pri aktívnom hand trackingu. Pri prijímaní hodnôt klientmi platí vzorec p*65,52kB/s pri aktivnom hand trackingu, pričom p označuje počet nelokálnych klientov voči odoslaným dátam.

V skripte XRCharacterManager.cs bola vytvorená Command funkcia s názvom `CmdChangeControllerType()`, ktorá zabezpečuje odovzdávanie informácií o prepínaní medzi ovládačmi a hand trackingom na server. Server následne informuje všetkých nelokálnych klientov o zmene ovládania lokálneho používateľa. Pri tejto funkcii je z klienta na server odoslaná hodnota 1 int. Server následne pošle hodnotu 1 int každému nelokálnemu klientovi. 1 int má 4 bajty. To znamená, že odoslanie tejto hodnoty na server zvýši prenos o 0,004kB na jedno prepnutie ovládania. Pri prijímaní hodnôt klientmi platí vzorec p*0,004kB, pričom p označuje počet nelokálnych klientov voči odoslaným dátam. Stručné výsledky týchto výpočtov na jedného používateľa sú zapísané v tabuľke 3.3.

Prenášané údaje	Počet hodnôt	Veľkosť prenosu	Rýchlosť prenosu	Poznámka k prenosu
Network Transform (8x)	56 float	0,224 kB	10,08 kB/s	-
Synchronizácia prstov (2 ruky)	364 float	1,456 kB	65,52 kB/s	Pri aktívnom hand trackingu
Zmena ovládania (odosielanie)	1 int	0,004 kB	-	Raz pri každej zmene ovládania
Príjem údajov klientom	Rovné príslušnej hodnote a typu	Rovné príslušnej veľkosti	Rovné príslušnej rýchlosti	Rovné príslušným poznámkam

Tabuľka 3.3: Špecifikácia sieťového prenosu údajov na jedného používateľa

3.6 Porovnanie Open a Meta XR rigov v aplikácii

Tieto rigy poskytujú podobnú funkcionalitu, no ich konfigurácia je rozdielna. V Open XR rigu je ovládanie implementované pre obe ruky naraz. To znamená, že ak sa zmení ovládanie z rúk na ovládače a opačne, zmenia sa obe ruky. V Meta rigu je naopak zabezpečená aj kombinácia ruka a ovládač.

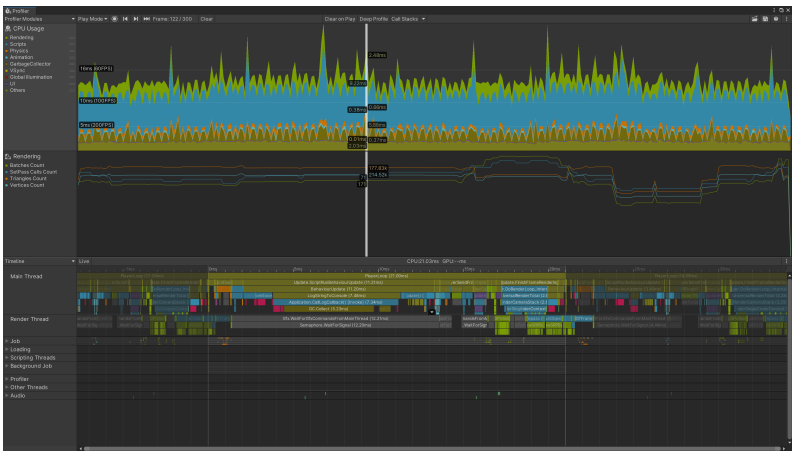
Ďalším rozdielom je pohyb. Pri Open XR rigu je zachovaný pohyb pomocou controllerov a ruky neobsahujú podporu teleportu. Meta XR rig obsahuje Turner a Teleport Meta interaktory, ktoré zabezpečujú otáčanie a teleportovanie pomocou konkrétnych gest. Avšak pri tomto rigu nie je zabezpečený plynulý pohyb pomocou controllerov ako pri Open XR rigu.

Open XR rig obsahuje NearFar a Poke interaktory pre interakcie. Meta XR rig obsahuje viacero interaktorov, ako sú Hand Poke Interactor, Hand Grab Interactor, Teleport a Turner interaktory, Distance Hand Grab Interactor, Touch Hand Grab Interactor a Hand Grab Use Interactor. Čo sa týka interactables Open XR, riešenie obsahuje dva základné typy - XRI Grab Interactable a XRI Poke Interactable. Meta XR obsahuje Grab Interactable, Distance Grab Interactable, Poke Interactable, Teleport Hotspot a Ray Interactable. Hierarchia oboch rigov je postavená na rovnakom princípe, ale Meta XR hierarchia ponúka viacero pomocných child

objektov. Sú to objekty ako napríklad Anchor pointy Left Eye, Right Eye, Center Eye, Tracker Anchor, Left Hand a Right Hand. Čo sa týka objektov Hand Tracking rúk, Open XR rig obsahuje základné objekty, ktoré obsahujú tri hlavné komponenty a objekty pre všetky kosti rúk a interaktory. Pri Meta XR rigu objekty rúk obsahujú aj HandData, HandFeatures a OVRHandDataSource objekty.

3.7 Výsledky získané pomocou Unity profilera

Aplikácia bola testovaná aj s pripojením na profiler. Každé zariadenie bolo pri taktomto testovaní pripojené pomocou kábla na Desktop PC pre zber údajov a výkon poskytovalo zariadenie samotné. Ako ukážka (obrázok 3.7) bolo vybrané testovanie multiplayer scény v aplikácii na zariadení Meta Quest 3. Zo všetkých meraní boli z profileru uložené .data súbory, ktoré sú posktynuté v prílohe.

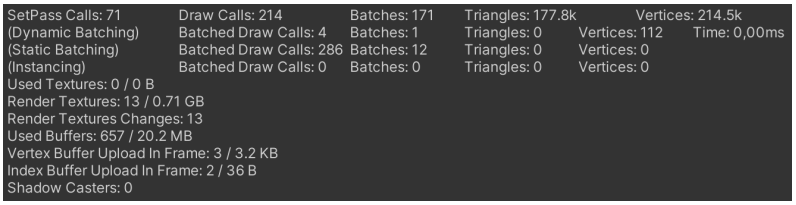


Obr. 3.7: Meta Quest 3 výsledky profilera

Popísané výsledky sa týkajú konkrétneho framu (CPU frame time = 21.03ms) na obrázku 3.7. Najväčšiu záťaž CPU (podmienka = 1% a viac) pre systém spôsobovali v PlayerLoope, ktorý trval 21.00ms nasledovné zložky (pri zložkách, ktorých vykonávanie trvalo viac ako primeraný čas je informácia v liste označená):

- `GC.Collect` (27.9% = 5.88ms) - Uvoľnenie pamäte pre objekty, ktoré sa už nepoužívajú (veľmi vysoká hodnota).
- `Application.CallLogCallback() [Invoke]` (6.8% = 1.45ms) - Logovanie (vysoká hodnota).
- `UIInputModule.Update() [Invoke]` (4.5% = 0.96ms) - Aktualizácia UI vstupov (vysoká hodnota).

- `StackTraceUtility.ExtractStringFromExceptionInternal() [Invoke]` (4.5% = 0.95ms) - Extrahovanie a konvertovanie zásobníkovej stopy z výnimiek na reťazec (vysoká hodnota).
- `XRBaseController.Update() [Invoke] (3.9% = 0.83ms)` - Aktualizácia VR ovládačov.
- `Semaphore.WaitForSignal(PostUpdate) (2.5% = 0.45ms)` - Čakanie na signál po aktualizácii PostUpdate (mierne vysoká hodnota).
- `InputUpdate (2.2% = 0.47ms)` - Aktualizácia stavu vstupných zariadení.
- `ClipperRegirsty.Cull` (2% = 0.44ms) - Selektívne vykresľovanie objektov -> Zobrazenie iba viditeľných pre používateľa.
- `SolverManager.LateUpdate() [Invoke]` (1.9% = 0.40MS) - Late Update pre VR IK solver.
- `Inl_ScriptableRenderContext.Submit` (1.7% = 0.36ms) - Odosielanie renderovacích príkazov na GPU.
- `Semaphore.WaitForSignal(Physics)` (1.5% = 0.32ms) - Čakanie na signál po aktualizácii fyziky.
- `Application.InvokeOnBeforeRender() [Invoke]` (1.3% = 0.29ms) - Callback poslednej aktualizácie pred vykreslením framu.
- `EarlyUpdate.XRUpdate` (1.3% = 0.27ms) - Aktualizácia polohy a orientácie HMD zariadení.
- `Profiler.FlushMemoryCounters` (1.1% = 0.25ms) - Aktualizácia a čítanie v profileri (mierne vyššia hodnota).
- `UGUI.Rendering.EmitWorldScreenspaceCameraGeometry` (1.1% = 0.23ms) - Generovanie geometrie pre UI prvky (mierne vyššia hodnota).
- `GUIUtility.BeginTUI() [Invoke]` (1.0% = 0.22ms) - Inicializácia a príprava systému pre vykreslenie IMGUI (mierne vyššia hodnota).



Obr. 3.8: Meta Quest 3 výsledky profilera

Čo sa týka renderovania framu výsledky (obrázok 3.7) boli nasledovné 3.8:

- SetPass Calls - 71 (zmeny shader passov).
- Draw Calls - 214 (individuálne rendering príkazy).
- Batches - 171 (zoskupené rendering príkazy).
- Triangles - 177.8k (počet spracovaných trojuholníkov).
- Vertices - 214.5k (počet spracovaných vrcholov).
- (Dynamtic Batching) Batched Draw Calls - 4.
- (Dynamic Batching) Batches - 1.
- (Dynamic Batching) Vertices - 112.
- (Static Batching) Batched Draw Calls - 286.
- (Static Batching) Batches - 12.
- Render Textures - 13 (veľkosť 0.71 GB).
- Used Buffers - 657 (veľkosť 20.2 MB).
- Vertex Buffer Upload In Frame - 3 (veľkosť 3.2 KB).
- Index Buffer Upload In Frame - 2 (veľkosť 36 B).

4 Záver

Hlavným cieľom práce bolo implementovať do systému hand tracking. Systém fungoval na ovládaní iba pomocou ovládačov a to bolo vo výsledku upravené tak, aby sa dalo medzi ovládačmi a rukami prepínať v ľubovoľnom čase. Návrh tohto riešenia je detailne popísaný v diagramoch 2.1 a 2.2. Implementácie týchto vylepšení je možné nájsť v sekciách 2.3 a 2.6 kapitoly 2. Pri Open XR riešení je možné prepnúť sa medzi oboma ovládačmi alebo rukami. Pri Meta XR riešení je možnosť používať aj hybridné ovládanie, čo v tomto prípade znamená ovládanie aplikácie pomocou jedného ovládača a jednej ruky. Prepnutie medzi rukami a ovládačmi je aktivované tak, že používateľ o seba dvakrát ťukne ovládače od zariadenia Meta Quest 2, Meta Quest 3 a ďalších. Pri tejto akcii zariadenie rozpozná, že používateľ chce prepnúť ovládanie. Ak chce používateľ prejsť z rúk späť na ovládače, stačí len zobrať ovládače do rúk a na tie sa aplikácia automaticky prepne. Táto možnosť prináša prirodzenejšie ovládanie. Používateľ v ruke nemusí držať ovládač a avatar je synchronizovaný s jeho prstami na ruke. Pri vyhodnotení bolo zistené, že najprecíznejšie zobrazenie rúk je dosiahnuté pomocou Left Hand Tracking a Right Hand Tracking modelov. Tieto modely sú prepojené so zápästiami avatara, ktoré ich dynamicky nasledujú. Synchronizácia prstov prináša nový rozmer pre rôzne druhy terapií. Pri terapiách je tak možné sústrediť sa na presné detailné pohyby prstov. Hand tracking pacienta odbremení od držania ovládačov a terapia tak môže byť efektívnejšia.

Ďalším cieľom práce bolo pridať k hand trackingu aj interakčné prvky. Návrh interakcií je popísaný v diagrame 2.1 a implementáciu je možné nájsť v sekciách 2.4, 2.5, 2.7 a 2.8 v kapitole 2. Riešenie zabezpečilo plné využitie hand trackingu. Nakonfigurované interaktory poskytujú interakciu používateľa s objektmi. Intractable objekty v aplikácii interagujú s interaktormi podľa konfigurácie. Pri Open XR riešení bol implementovaný Poke Interactor, ktorý zabezpečuje interakciu pomocou dotyku objektu a NearFar Interactor, ktorý zabezpečuje manipuláciu s objektom pomocou lúča. Pre tieto interaktory sú v tomto riešení nakonfigurované Poke a Grab interactables. Objekty, ktoré boli v projekte už vložené

sú rovnako kompatibilné s NearFar interacotrom. Pri Meta riešení je viac typov interaktorov a interactables. Tie zabezpečujú rovnakú funkcionalitu a umožňujú aj interakcie ako teleportovanie na hotspoty alebo ľubovoľné miesto, otáčanie sa a interakciu s novým typom Ray Interactable menu. Tieto interakcie dopĺňajú hand tracking a používateľ tak môže ovládať aplikáciu iba pomocou rúk, čo mu umožní sa viac sústrediť na terapiu než na ovládanie. Výsledky a vyhodnotenie testovania hand trackingu a interakcií s objektmi sú dostupné v sekciách 3.1, 3.2 a 3.6 v kapitole 3.

Ako ďalšie bolo v aplikácii implementované serverové riešenie pre Open XR rig. Implementácia je detailne popísaná v sekcii 2.10 v kapitole 2 a výsledky testovaní sú dostupné v sekciách 3.3, 3.5 a 3.7 v kapitole 3. To zabezpečilo, že Open XR vylepšenia sú správne viditeľné a synchronizované pre všetkých používateľov. Používatelia si tak navzájom v reálnom čase vidia zosynchronizovaných avatarov prepojených s ich rukami a prstami. Pomocou nich môžu interagovať s prostredím a tak vykonávať online terapie.

Táto práca obsahuje aj vylepšenie scalera avatara. Návrh popisujú diagramy 2.3 a 2.4, implementácia je popísaná v sekcii 2.9 v kapitole 2. Testovanie vykonané v sekcii 3.1 v kapitole 3 zahŕňalo aj testovanie tohto vylepšenia. Na začiatku implementácie tento scaler poskytoval iba automatické nastavenie mierky celého avatara pomocou tlačidla Reset Scale. Tento scaler bol vylepšený tak, že si používateľ môže pomocou šípok pridať alebo ubrať veľkosť celého avatara, jeho rúk s dlaňami alebo iba dlaní. To je pre používateľov veľkým prínosom, pretože takéto detailné nastavenie umožní lepšie prepojenie s vitruálnym avatarom. Polohy a pohyby rúk tak budú oveľa precíznejšie premietané z reality na virtuálneho avatara a to zabezpečí aj lepšie výsledky terapie.

Na základe súčasného stavu aplikácie existujú možné vylepšenia do budúcna. Veľkým prínosom pre túto aplikáciu by bolo vytvorenie nových scén a aktivít, ktoré môže pacient vykonávať. Tieto scény môžu byť aj dynamické a tak by pacient mohol sedieť alebo stáť na mieste a prostredie by sa mu prispôsobovalo podľa jeho akcií. To by sa efektívne dopĺňalo s hand trackingom. Každá scéna by tak pacienta previedla terapiou s rôznou tématikou. Po ukončení scény by boli sledované výsledky zaslané terapeutovi. To by pacientovi dodalo ďalšiu motiváciu pre ďalšie takéto terapie.

Ako technický prínos pre aplikáciu by bola vhodná optimalizácia sieťového prenosu údajov prstov na ruke, ktorého výsledky je možné nájsť v sekcii 3.5 v kapitole 3. Dobrým riešením pre tento prípad je implementácia techník, ako je napríklad interpolácia. Pri tejto funkcionalite by zároveň bolo vhodné vymys-

lieť alternatívu k zisťovaniu aktívnych vstupov mimo `Update()`. Keďže hodnota GC.Collect predstavovala až 27.9% práce procesora na frame z obrázka 3.7, bolo by vhodné optimalizovať spôsob, akým aplikácia vytvára, používa a zbavuje sa objektov. Vysokú hodnotu ukazuje aj logovanie, ktoré vo finálnej aplikácii nie je až tak potrebné. Rovnako vhodné je zváženie alternatívnych prístupov k ošetreniu chybových stavov, ktoré by nevyžadovali volanie výnimiek. Mierne vysoké hodnoty boli zistené pri generácii geometrie pre UI prvky v priestore kamery a pri inicializácii systému pre vykreslenie IMGUI (Immediate Mode GUI). Tieto hodnoty nie sú kritické, avšak naznačujú priestor pre potenciálnu optimalizáciu používateľského rozhrania, ako je napríklad zníženie frekvencie prekresľovania UI alebo nahradenie IMGUI systémom založeným na Unity UI (uGUI).

Literatúra

1. META. *Meta Quest*. 2025. Dostupné tiež z: https://about.meta.com/technologies/meta-quest/.
2. BEZMALINOVIC, Tomislav. *Meta Quest 3 - Price, Specs...* 2023. Dostupné tiež z: https://mixed-news.com/en/meta-quest-3-roundup-specs-release-price/.
3. MICROSOFT. *Hololens 2*. 2025. Dostupné tiež z: https://learn.microsoft.com/sk-sk/hololens/hololens2-options.
4. MICROSOFT. *Remote Rendering*. 2025. Dostupné tiež z: https://azure.microsoft.com/en-us/products/remote-rendering/.
5. COMPANY, OxfordVR 2019. *OxfordVR*. 2019. Dostupné tiež z: https://oxfordvr.co/.
6. OSSO VR, Inc. *Operate at a higher level*. 2024. Dostupné tiež z: https://www.ossovr.com/.
7. BABA, Saiful Ariffin; HUSSAIN, Hanafizan; EMBI, Zarina Che. An overview of parameters of game engine. *IEEE Multidisciplinary Engineering Education Magazine*. 2007, roč. 2, č. 3, s. 10–12.
8. DRAGONFLY. *Top 56 3D Game Engines Compared*. 2024. Dostupné tiež z: https://www.dragonflydb.io/game-dev/engines/3d.
9. UNITYTECHNOLOGIES. *Unity Documentation*. 2025. Dostupné tiež z: https://docs.unity.com/.
10. HAAS, John K. A history of the unity game engine. *Diss. Worcester Polytechnic Institute*. 2014, roč. 483, č. 2014, s. 484.
11. UNITYTECHNOLOGIES. *Hand data model*. 2024. Dostupné tiež z: https://docs.unity3d.com/Packages/com.unity.xr.hands@1.5/manual/hand-data/xr-hand-data-model.html.
12. MIRRORNETWORKING. *User Manual*. 2024. Dostupné tiež z: https://mirror-networking.gitbook.io/docs/manual/general.

13. META. *Meta XR All-in-One SDK*. 2025. Dostupné tiež z: `https://developers.meta.com/horizon/downloads/package/meta-xr-sdk-all-in-one-upm/`.

14. META. *Developing extended reality apps for Horizon OS in Unity*. 2025. Dostupné tiež z: `https://developers.meta.com/horizon/documentation/unity/unity-development-overview/`.

15. NEHILA, Peter. *Neurorehabilitácia v zdieľanej virtuálnej realite*. 2023. Dostupné tiež z: `https://hron.fei.tuke.sk/~korecko/zavPr/DP_Nehila.pdf`.

Zoznam príloh

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Terapeutický systém na báze technológií eXtended (XR) reality

Rozšírenie základných príručiek systému

2025 **Bc. Adam Mikael**

Úvod

Rozšírenie je doplnkom k základným príručkám systému, ktoré je možné nájsť v bibliografii ako prílohy práce s názvom „Neurorehabilitácia v zdieľanej virtuálnej realite" [15]. V citovanej práci je implementovaná základná verzia tohto systému a nachádzajú sa v nej aj systémová a používateľská príručka. Tie obsahujú všetky základné informácie od prekladu aplikácie pre zariadenia až po ovládanie aplikácie pomocou ovládačov. Tento doplnok nadväzuje na zmienené príručky a obsahuje informácie k vylepšeniam systému implementovaným v tejto práci. Informácie sa týkajú používania hand trackingu, interakcií s objektmi (Inetractables) pomocou virtuálnych rúk a ich prvkov (Interactors) a používania vylepšeného scalera. Obsah doplnku je podrobne rozpracovaný aj v hlavnom texte tejto práce. Doplňujúce informácie je možné nájsť v tomto rozšírení.

Preklad aplikácie pre Meta Quest 3

Tento systém bol testovaný na zariadeniach Meta Quest 2 a Meta Quest 3. Spomínané príručky obsahujú informácie o preklade aplikácie pre zariadenie Meta Quest 2 (Oculus Quest 2). Preklad pre zariedenie Meta Quest 3 je rovnaký. Pri buildovaní je potrebné nastaviť Run Device na Meta Quest 3 zariadenie.

Aktivácia a deaktivácia hand trackingu

Pri tomto ovládaní je potrebné mať zapnutý hand tracking v nastaveniach Meta Quest zariadenia, ktorý je možné nájsť v Settings -> Movement Tracking -> Hand Tracking. V týchto nastaveniach sa dá hand tracking detailne nakonfigurovať. Hand tracking je možné aktivovať dvoma spôsobmi. Prvý je odloženie ovládačov na miesto, kde ich nebude zariadenie detekovať a počkať kým sa prepne na ruky. Takýto spôsob je kompatibilnejší so zariadením Meta Quest 3. Druhý spôsob je plne kompatibilný s oboba zariadeniami a je odporúčaný hlavne pre Meta Quest 2. Pri tomto spôsobe je potrebné dvojité klepnutie ovládačov headsetu o seba. Tým Meta Quest zariadenie detekuje aktiváciu hand trackingu. Po aktivácii sa systém automatický prispôsobí aktívnemu ovládaniu. Pre deaktiváciu hand trackingu je potrebné zobrať ovládače headsetu do rúk. Headset detekuje aktivitu ovládačov a systém sa automaticky prispôsobí. Ak detekcia neprebehne správne, je možné použiť rovnaký postup dvojitým klepnutím ovládačov headsetu o seba, ako pri aktivácii hand trackingu.

Interakcia s objektmi

Systém obsahuje viacero špecifických interakcií s objektmi. Každá z nich zahŕňa príslušné interaktory (prvky, ktoré vykonávajú interakciu) a Interactables (objekty, s ktorými sa dá interagovať). Následovný zoznam obsahuje opis interaktorov a interatables v systéme a informácie o ich ovládaní:

- (Open XR Rig) NearFar Interactor - Pracuje pomocou lúča (Raycast). Pri namierení lúča na objekt (XRI Grab Interactable) a následnom spojení ukazováka a palca je objekt uchopený. Po oddialení ukazováka od palca je objekt pustený. S objektom je možné interagovať na blízko aj na diaľku. Ak je objekt blízko, interaktor je skrytý no funcionalita ostáva zachovaná.

- (Open XR Rig) Poke Interactor - Hranice tohto interaktora sú ruky a lúče pred nimi. Ak hranice preniknú do objektu (XRI Poke Interactable), podľa gesta je vykonaná príslušná akcia. Objekt napríklad pri vniknutí prstom zmení farbu a pri následnom spojení prsta a ukazováka pri prieniku zmení farbu na inú.

- (Meta XR Rig) Hand Poke Interactor - Tento interaktor pracuje s objektom (Meta Poke Interactable) na princípe tlačidla. To znamená, že keď je stlačená plocha objektu tak sa automaticky pohne podľa prsta.

- (Meta XR Rig) Hand Grab Interactor - Pri úchope rukou je objekt (Meta Grabbable) uchopený. Po pustení je pustený aj objekt. Pre funkčnosť je potrebný priamy kontakt s objektom.

- (Meta XR Rig) Distance Hand Grab Interactor - Funguje rovnako ako Meta Hand Grab Interactor, no poskytuje aj interakcie na diaľku. Po namierení na objekt (Meta Distance Grabbable) a následnom úchope je objekt pritiahnutý do dlane.

- (Meta XR Rig) Hand Ray Interactor - Tento interaktor pracuje s Canvas (Menu) objektom (Meta Ray Interactable). Pri namierení ruky sa na plátne zobrazí kurzor v podobe kruhu. Pred rukou je zároveň zobrazený objekt v tvare šípky, ktorý ukazuje smer interakcie. Potvrdenie interakcie sa vykoná spojením ukazováka a palca.

- (Meta XR Rig) Turner Interactor - Poskytuje možnosť otáčať sa do strán. Ruka musí byť otočená vetrikálne. Následne musia byť malíček, prsteník a prostredník plne zohnuté. Ukazovák a palec musia byť plne vystreté. Takéto

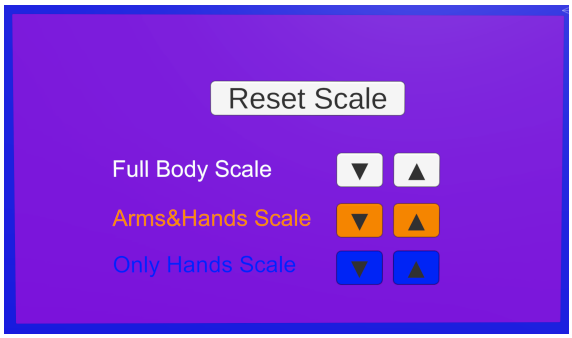
gesto aktivuje interaktor. Po aktivácii sa zobrazia dve šípky. Po namierení na jednu z nich a spojení palca a ukazováka sa vykoná rotácia do príslušnej strany.

- (Meta XR Rig) Teleport Interactor - Posktuje teleport. Tento interaktor je aktivovaný tak, že najprv je potrebné aktivovať Meta Turner Interactor. Následne je potrebné otočiť ruku s daným gestom z vertikálneho otočenia do horizontálneho. Takto sa aktivuje interaktor. Po spojení ukazováka a palca je vykonaný teleport na miesto (Meta Teleport Hotspot alebo Meta Teleport Interactable), kde je prstami namierené.
- (Meta XR Rig) Touch Hand Grab Interactor - Funguje ako Meta Hand Grab Interactor a poskytuje dynamické prispôsobenie prstov k povrchu uchopeného objektu.
- (Meta XR Rig) Hand Grab Use Interactor - Objekt je možné uchopiť a následne pomocou ukazováka pri úchope stláčať spúšť pre aktiváciu objektu.

Funkcionalita menu scalerov

Pre scalovanie avatara pomocou menu na obrázku 1 príslušná šípka hore zvýši konkrétnu mierku o 0,02 a príslušná šípka dole zníži konkrétnu mierku o 0,02.

- Reset Scale - Automatické scalovanie avatara podľa výšky.
- Full Body Scale - Scalovanie celého avatara.
- Arms & Hands Scale - Scalovanie celých rúk avatara.
- Only Hands Scale - Scalovanie dlaní avatara.



Obr. 1: Menu upravených scalerov