Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Neurorehabilitácia v zdieľanej virtuálnej realite

Diplomová práca

2023 **Bc. Peter Nehila**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Neurorehabilitácia v zdieľanej virtuálnej realite

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Ing. Štefan Korečko PhD.

Košice 2023 **Bc. Peter Nehila**

Abstrakt v SJ

Virtuálna realita je v dnešnej dobe používaná na širokú škálu účelov. Je možné ju taktiež využiť aj na účely medicíny, napríklad v procese terapie a rehabilitácie pacientov. Výhodou aplikácie virtuálnej reality do procesu rehabilitácie je možnosť tvorby zaujímavých a pohlcujúcich virtuálnych prostredí a scenárov. Na základe analýzy možností využitia technológií a aktuálneho riešenia sme vo vývojom prostredí Unity vytvorili aplikáciu, ktorá umožní pacientom a terapeutom spolupracovať v zdieľanom virtuálnom prostredí. Aplikácia obsahuje niekoľko jednoduchých scenárov, ktoré je možné v budúcnosti rozširovať a spájať do zložitejších, zážitvnejších sekvencií, ako aj rozhranie na komunikáciu s externými prostrediami, ktoré sú schopné vyhodnotiť stav pacienta.

Kľúčové slová v SJ

C#, Kolaboratívne virtuálne prostredie, Neurorehabilitácia, Unity, Virtuálna realita

Abstrakt v AJ

Virtual reality is used for a wide range of applications nowadays. It can also be used for medical purposes, for example in the process of therapy and rehabilitation of patients. The advantage of applying virtual reality to the rehabilitation process is the possibility of creating interesting and immersive virtual environments and scenarios. Based on an analysis of the technology's potential uses and current solution, we have created an application in Unity, that will allow patients and therapists to collaborate in a shared virtual environment. The application includes several simple scenarios that can be expanded and combined into more complex, engaging sequences in the future, as well as interface to communicate with external environments to assess the patient's actions.

Kľúčové slová v AJ

Collaborative virtual environment, C#, Neurorehabilitation, Unity, Virtual reality

Bibliografická citácia

NEHILA, Peter. *Neurorehabilitácia v zdieľanej virtuálnej realite*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2023. 74s. Vedúci práce: Ing. Štefan Korečko PhD.

68879

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

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

Názov práce:

Neurorehabilitácia v zdieľanej virtuálnej realite

Neurorehabilitation in Shared Virtual Reality

Študent: **Bc. Peter Nehila**
Školiteľ: **Ing. Štefan Korečko, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:
1. Analyzovať súčasné prístupy k terapii vo virtuálnej realite.
2. Analyzovať riešenie pre neurorehabilitáciu vo webovej virtuálnej realite, vyvinuté na školiacom pracovisku, a požiadavky na jeho rozšírenie.
3. Na základe analýzy navrhnúť riešenie pre neurorehabilitáciu, postavené na hernom rámci Unity.
4. Riešenie navrhnúť primárne ako zdieľané a pre použitie s virtuálno-realitnými prilbami.
5. Navrhnuté riešenie implementovať a overiť.
6. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 21.04.2023
Dátum zadania diplomovej práce: 31.10.2022

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -12-

.....
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

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

Košice, 13.5.2023 .....

Vlastnoručný podpis

Poďakovanie

Veľmi rád by som poďakoval svojmu vedúcemu práce Ing. Štefanovi Korečkovi za jeho pomoc a odborné vedenie pri písaní mojej diplomovej práce. Zároveň ďakujem tímu Ing. Mgr. Romana Rosipala, DrSc. z Ústavu merania SAV za odborné rady.

Rád by som tiež poďakoval sestrám za cennú spätnú väzbu a pomoc s testovaním aplikácie. Nakoniec sa chcem poďakovať aj celej mojej rodine za trpezlivosť a podporu počas štúdia.

Obsah

Úvod **1**

1 Virtuálna realita a jej využitie v terapii **3**
1.1 Kolaboratívne virtuálne prostredia 5
1.2 Rehabilitácia vo VR 6
1.3 Existujúce projekty 8

2 Aktuálny stav projektu **11**
2.1 Aktuálna implementácia 11
2.1.1 Používateľské rozhrania 11
2.1.2 Komunikácia a zabezpečenie sieťového spojenia 13
2.1.3 Animácie pohybu 14
2.2 Analýza implementácie 14
2.2.1 Stav kódu 15
2.2.2 Komunikácia modulov 16
2.2.3 Animácie pohybu 16

3 Prototyp riešenia **17**
3.1 Unity XR 17
3.2 Požiadavky na minimálny prototyp 18
3.2.1 Sieťová infraštruktúra 19
3.2.2 Komunikácia s OpenVibe prostredím 19
3.2.3 Synchronizácia klientov 20
3.2.4 Viditeľnosť objektov z pohľadu jednotlivých používateľov . 21
3.3 Testovanie prototypu 22
3.4 Porovnanie Unity a A-Frame 24

4 Návrh a implementácia riešenia **27**
4.1 Návrh systému a procesov 27
4.1.1 Podporované platformy 28

4.1.2 Proces animovania počas rehabilitácie 28
4.1.3 Proces rehabilitácie 28
4.1.4 Zdieľané časti objektov používateľa 29
4.1.5 Optimalizácia pri práci s VR 30
4.2 Architektúra systému 31
4.3 Podsystém používateľského rozhrania 32
4.3.1 Úvodné menu 32
4.3.2 Menu pre terapeuta 33
4.4 Funkcie avatarov 36
4.4.1 Animácie avatarov 39
4.4.2 Zmena avatara 43
4.5 Dynamické animácie rúk pacienta 44
4.5.1 Počítanie inverznej kinematiky 44
4.5.2 Ukážková a reálna animácia 45
4.5.3 Pozície pre pohyby animácie 47
4.5.4 Rozdielne typy animácií 50
4.6 Prispôsobenie nastavení spojených s VR 52
4.7 3D prostredie 56
4.8 Desktopový klient 58
4.9 Špecifické nastavenia servera 59
4.9.1 Vytváranie objektov na serveri 59
4.9.2 Komunikácia so serverom 61

5 Vyhodnotenie implementácie **62**

6 Záver **68**

Literatúra **71**

Zoznam skratiek **75**

Slovník **76**

Zoznam príloh **77**

Systémová príručka **78**

Používateľská príručka **89**

Zoznam obrázkov

1.1 Ukážka systému BRAVEMIND [17] 9
1.2 Hubs - webové rozhranie 9

2.1 Možnosti zobrazenia rozhrania 12
2.2 Rozhranie pacienta 12
2.3 Rozhranie pacienta - ukážka pohybu chytenia kocky 13
2.4 Rozhranie terapeuta s ovládaním nastavení 14
2.5 Rozhranie terapeuta bez nastavení 15
2.6 Ukážka komunikácie medzi modulmi 16

3.1 Komunikácia medzi našim systémom a prostredím OpenVibe . . . 20
3.2 Výber roli používateľa 22
3.3 Rozdielne UI 22
3.4 Využitie rozdielnych avatarov 23
3.5 Pohľad 2 hráčov na seba 23
3.6 Pohľad 2 hráčov na seba so synchronizovaným pohybom rúk 24

4.1 Proces zobrazenia animácie v rámci rehabilitácie 29
4.2 Proces nastavenia animácie z pohľadu terapeuta 30
4.3 Ukážka hlavného menu v offline scéne 33
4.4 Menu terapeuta určené na ovládanie animácií 34
4.5 Ukážka menu na zápästí 34
4.6 Ukážka statického menu v 3D prostredí 35
4.7 Menu na prispôsobovanie pozícií 37
4.8 Ukážka avatara 38
4.9 Štruktúra modelov avatarov 39
4.10 Ukážka použitia starého modelu ruky 45
4.11 Ukážka použitia modelu ruky, ktorý je súčasťou avatara 45
4.12 Ukážka animácií z pohľadu terapeuta 46
4.13 Priebeh zmeny hodnoty pomocou krivky [28] 47

4.14 Ukazovatele pozícií 49
4.15 Ukazovateľ dosahu ruky 50
4.16 Ukážka rôznych typov animácií 52
4.17 Komponenty trackovania headsetu 54
4.18 Pohľad na stôl 57
4.19 Pohľad na miestnosť 57
4.20 Menu na desktopovom klientovi 59

5.1 Vizualizácia počtu snímkov - desktop 64
5.2 Vizualizácia počtu snímkov - desktop + simulovaný VR headset . . 64
5.3 Vizualizácia počtu snímkov - laptop 65
5.4 Vizualizácia počtu snímkov - laptop + simulovaný VR headset . . . 65
5.5 Vizualizácia počtu snímkov - laptop (eko režim) 66
5.6 Vizualizácia počtu snímkov - laptop (eko režim) + simulovaný VR headset 66
5.7 Vizualizácia počtu snímkov - Oculus Quest 2 67
5.8 Vizualizácia počtu snímkov - Oculus Quest 2 opakovaný test 67

Zoznam tabuliek

3.1 Porovnanie Unity a A-Frame 25

5.1 Hardvérové špecifikácie zariadení použitých pri testovaní 63
5.2 Namerané hodnoty počtu snímkov za sekundu 67

Zoznam kódov

4.1 Ukážka animovania chôdze v prípade pohybu hlavy 41
4.2 Ukážka posunu objektu 48
4.3 Kontrola pozície objektu, či je v dosahu ruky a zároveň nad stolom 51
4.4 Odobratie autority nad objektami pri odpojení klienta 60

Úvod

V oblasti terapie sa v dnešnej dobe neustále skúšajú nové metódy, ktoré by mohli zjednodušiť prácu ako pre terapeutov, tak aj pre samotných pacientov. Medzi hlavné ciele patrí okrem toho aj zlepšenie výsledkov a zefektívnenie liečby.

Virtuálna realita začala v poslednej dobe rýchlo napredovať a to, čo sme si ešte pred zopár rokmi nevedeli predstaviť, máme teraz priamo na dosah ruky. Pokrok bol zaznamenaný taktiež v rámci zobrazovacích technológií. Prešli sme z pomerne pomalých a nepraktických zariadení k zariadeniam, ktoré dokážu bez potreby pripojenia k počítaču fungovať samostatne s dostatočnou kvalitou obrazu a s vysokou presnosťou snímačov pohybu. Virtuálna realita zatiaľ ešte nie je niečo, s čím by sa ľudia stretávali na dennej báze, no tento trend sa možno bude v budúcnosti postupne meniť. Systémy virtuálnej reality umožňujú jej využitie aj v zdravotníctve, napríklad pri rehabilitácii alebo terapii.

V dnešnej dobe už existuje niekoľko projektov, ktoré sa zameriavajú na terapiu a rehabilitáciu v kombinácii s použitím virtuálnej reality. Charakter týchto systémov sa líši v závislosti od toho, na čo konkrétne sa zameriavajú. Existujú systémy špecializované primárne na liečenie rôznych fóbií alebo na terapiu po nepríjemnej skúsenosti. Tieto systémy sú charakteristické tým, že skôr ide o niečo ako 3D virtuálne prostredie, kde samotný pacient nie je schopný veľmi zasahovať. Existujú aj systémy, pre ktoré boli vytvorené rôzne špecializované rukavice alebo iné pomocné zariadenia, ktorých účelom je zlepšenie výsledkov terapie.

Jeden takýto systém je vyvíjaný aj v rámci projektu APVV-21-0105 s názvom "Dôveryhodná interakcia človek–robot a terapeut–pacient vo virtuálnej realite", ktorého súčasťou je aj táto diplomová práca. Systém obsahuje kolaboratívne prostredie, čo je prostredie umožňujúce interakciu viacerých používateľov navzájom v reálnom čase. Systém je schopný komunikovať s externým zariadením, ktoré slúži na snímanie aktivity pacienta a zistenie, či sa pokúša vykonať pohyb, ktorý sa od neho požaduje. Následne sa vo virtuálnej realite vyobrazí animácia pohybu ruky.

Formulácia úlohy

Cieľom tejto práce je návrh a implementácia systému určeného na asistenciu pri neurorehabilitácii pacientov. Medzi hlavné vlastnosti aplikácie, ktoré je potrebné splniť, je použitie rámca Unity, implementácia rozhrania na použitie virtuálnej reality a zabezpečenie možnosti spolupráce v zdieľanom virtuálnom prostredí. Taktiež je potrebné implementovať funkcionalitu z už existujúceho riešenia vytvoreného použitím jazyka JavaScript.

V kapitole 1 je v prvom kroku potrebné vykonať analýzu možností virtuálnej reality a jej použitia na účely terapie a rehabilitácie. Detailnejšie sa zameriame na aspekty použitia heliem pre virtuálnu realitu a ich obmedzenia, ktoré vznikajú počas ich používania. Zároveň sa zameriame aj na už existujúce projekty. V rámci tohto kroku sa zameriame na spôsob fungovania týchto projektov a taktiež aj na prípadné nevýhody. Vzhľadom na to, že už existuje riešenie vytvárané v rámci tohto projektu v jazyku JavaScript, je nutné analyzovať túto implementáciu. Aktuálnemu stavu implementácie v jazyku JavaScript sa venujeme v kapitole 2. Našim hlavným cieľom bude analýza funkcií ako aj kódu aplikácie, výhod, prípadne nevýhod aktuálnej implementácie, ako aj možností a obmedzení použitých technológií.

V ďalšom kroku sa v kapitole 3 venujeme vytvoreniu prototypu za použitia herného rámca Unity. Vďaka prototypu dokážeme za pomerne krátky čas určiť, či je možné splniť naše požiadavky na funkcionalitu. Zároveň je počas tvorby prototypu potrebné zamerať sa na nedostatky technológie, ktoré by nás mohli obmedzovať, prípadne úplne znemožniť implementáciu plnej funkcionality systému.

Hlavným cieľom práce je následný návrh a implementácia systému určeného na neurorehabilitáciu pacientov vo virtuálnej realite. Tejto časti je venovaná kapitola 4. Počas implementácie sa budeme usilovať o vytvorenie systému, ktorý bude príjemný na použitie pre pacientov a zároveň aj terapeutov. Súčasne bude našim cieľom vytvorenie systému nielen na prácu s helmou pre virtuálnu realitu, ale taktiež aj vytvorenie klientskej aplikácie, ktorá bude použiteľná na osobnom počítači. V rámci riešenia taktiež navrhneme a implementujeme nové funkcie, ako aj zmeny oproti už existujúcemu systému.

Nakoniec v kapitole 5 vyhodnotíme nami vytvorenú implementáciu. Rozoberieme funkcionalitu, ktorú sa podarilo implementovať, prípadne čo sa implementovať nepodarilo a z akého dôvodu. Taktiež vyhodnotíme plynulosť behu aplikácie na rôznych systémoch.

1 Virtuálna realita a jej využitie v terapii

V spojení s virtuálnymi prostrediami existujú tri základné rozdelenia na základe technológií, s ktorými pracujú:

- **Virtuálna realita** (VR) je systém, ktorý poskytuje vysokú mieru pohltenia do virtuálneho sveta [1]. Svet virtuálnej reality je plne virtuálny, nemieša sa s prvkami reálneho sveta,
- **Zmiešaná realita** (MR) poskytuje spojenie reálneho sveta s virtuálnym, na úrovni zobrazovania počítačovo vytvorených objektov do reálneho sveta. Súčasťou takýchto systémov je aj zároveň schopnosť interakcie s takýmito objektami [2],
- **Augmentovaná realita** (AR) predstavuje taktiež spojenie reálneho sveta s virtuálnym, narozdiel od zmiešanej reality sa skôr snaží len o doplnenie skutočného sveta o doplňujúce informácie a iné počítačovo generované vylepšenia [1],
- **Rozšírená realita** (XR) je pojem používaný na pomenovanie zmiešanej reality, virtuálnej reality a augmentovanej reality. Táto kategória sa používa na pomenovanie všetkých foriem počítačom vytvorenej reality.

Všetky tieto virtuálne prostredia sa okrem iného vyznačujú aj tým, že sú spracovávané v reálnom čase.

Na vysvetlenie pojmu virtuálnej reality (VR) existuje niekoľko definícií. V zásade sa v prípade virtuálnej reality dá hovoriť o systéme, ktorý predstavuje simulovaný svet, s ktorým používateľ interaguje prostredníctvom dátových rukavíc alebo ovládačov a okuliarov na zobrazovanie virtuálneho prostredia [3]. Obraz je možné zobrazovať buď na monitore počítača alebo priamo na okuliaroch určených na stereoskopické zobrazenie. Virtuálna realita sa usiluje o čo najvernejšie

modelovanie sveta a zabezpečenie čo najlepšej interakcie medzi človekom a simulovaným svetom.

Prvé zmienky o technológiách virtuálnej reality, ako ju poznáme dnes, sa začali objavovať v 80-tých rokoch minulého storočia. Jaron Lanier a Thomas Zimmerman ako prví vytvorili koncept nástrojov na prácu s virtuálnou realitou. Lanierovi sa taktiež pripisuje vytvorenie pojmu virtuálna realita, ktorý bol neskôr spopularizovaný. Prvé nástroje na prácu s VR sa začali objavovať okolo roku 1990[4], tieto zariadenia boli ale často nepraktické a poskytovali len veľmi nízku kvalitu zážitku.

V dnešnej dobe sú systémy VR rozšírené v rôznych oblastiach. Medzi najčastejšie patria:

- počítačové hry,
- edukácia,
- modelovanie, dizajn, architektúra,
- medicína,
- vzdialená komunikácia.

V roku 2021 spoločnosť Meta Platforms, Inc. oznámila prácu na projekte Metaverse [1]. Tento projekt by mal predstavovať platformu, kde by ľudia mohli vykonávať rôzne aktivity počínajúc zábavnými hrami až po vykonávanie rôznej práce. Tento projekt bol prijatý verejnosťou so zmiešanými postojmi. Medzi najväčšie obavy patrí speňaženie projektu a taktiež tu vzniká otázka, či je takýto plne virtuálny svet, ktorý by mal úplne nahradiť reálny svet potrebný a či to je smer, ktorým by sa tieto technológie mali uberať.

V oblasti medicíny vzniklo viacero projektov. Veľká časť z nich ukázala sľubné výsledky pri práci s pacientmi počas klinických testov. Stručný prehľad tém nám preto môže pomôcť nasmerovať nás správnym smerom a pomôcť nám pri rozhodnutiach.

Autormi v [5] bol vytvorený projekt na rehabilitáciu pacientov po mŕtvici. K výsledkom, ktoré dosiahli, patrí napríklad zlepšenie pohybových funkcií pacientov. Projekt sa zameriaval na zlepšenie pohybu svalov dlaní. Rehabilitácia bola vykonávaná za pomoci dátových rukavíc, rovnako boli jednotlivé cvičenia prispôsobené pacientom podľa ich možností.

[1] https://about.facebook.com/meta/

V ďalšom projekte [6] autori vytvorili systém, využívajúci virtuálnu realitu na rehabilitáciu končatín. Systém obsahuje aj robotického asistenčného robota určeného na pomoc pri vykonávaní pohybov. Robot umožňuje pohyb vo viacerých stupňoch voľnosti naraz (celkovo obsahuje 18 stupňov voľnosti na viacerých kĺboch dokopy). Tréningové scenáre sú sprevádzané audiovizuálnymi pokynmi vo virtuálnej realite.

Ďalší systém[7] bol vyvinutý na terapiu a rehabilitáciu pacientov po mŕtvici. Autori okrem virtuálneho prostredia vyvinuli aj dátovú rukavicu, ktorá pomáhala pri pohyboch rúk. Testy vykonali na dvoch vzorkách pacientov - s použitím rukavice a bez. Pacienti, ktorí využívali kombináciu rukavice a VR preukázali lepšie výsledky než tí, ktorí využívali len VR na terapiu.

1.1 Kolaboratívne virtuálne prostredia

Pod pojmom kolaboratívne virtuálne prostredie (anglicky Collaborative Virtual Environment, CVE) si môžeme predstaviť systém, ktorý môže využívať viacero používateľov naraz, a tak môžu spoločne využívať jeho možnosti. Teda môžu spolupracovať a interagovať vo virtuálnom prostredí. Najdôležitejšou súčasťou CVE je zdieľanie informácií. Kolaboratívne virtuálne prostredia je taktiež možné definovať ako distribuované systémy virtuálnej reality, ktoré ponúkajú digitálny svet určený na zdieľanie informácií a interakciu medzi používateľmi za pomoci prostriedkov, ktoré daný systém ponúka [8]. Pri použití takejto definície je možné za CVE považovať aj systém, ktorý nie je nutne reprezentovaný 3D priestorom. Keď budeme ďalej v tejto práci rozprávať o týchto systémoch, budeme mať však na mysli tie CVE, ktoré sú reprezentované 3D virtuálnym prostredím.

V kolaboratívnych virtuálnych prostrediach je dôležitá reprezentácia používateľa, pomocou tzv. avatara. Avatar môže byť vyobrazený rôznymi spôsobmi, taktiež môže mať rôzne vlastnosti ako napríklad schopnosť vznášať sa alebo vykonávať iné činnosti, ktoré nie sú v reálnom svete možné. Taktiež je tu potrebné brať do úvahy aj prostredie, v ktorom budú používatelia spolupracovať. Podľa [9] je potrebné zvoliť vyobrazenie používateľa tak, aby bola identita hneď jasná a bolo možné jednoznačne rozlíšiť, o koho ide. Efektívnym využitím priestoru a pocitu prítomnosti je schopné navodiť príjemné pocity z používania systému a zlepšiť tak používanie systému. Gestá a mimika je samostatná kategória, ktorou je potrebné sa zaoberať. Možnosť vyobrazenia je v takýchto systémoch obmedzená, no aj tu sú určité možnosti, ako jednotlivé aspekty do systému zaviesť. Jedným z nich by mohli byť napríklad ukazovatele toho, či niekto momentálne rozpráva vo

forme animácií úst [9].

1.2 Rehabilitácia vo VR

V rámci oblasti rehabilitácie vo VR musíme uvažovať nad niekoľkými otázkami, hlavnou z nich je kvalita rehabilitácie a jej reálne výsledky. Zlepšenie výsledkov je podľa [10] možné dosiahnuť vylepšením hneď viacerých aspektov. Medzi tie patrí:

- prirodzenosť,
- angažovanosť,
- pocit prítomnosti,
- náklonnosť voči samotnej VR aplikácii.

Dôležitým je aj pocit pohltenia (anglicky immersion), ktorý so sebou nesie ďalšie pozitívne pocity z práce vo virtuálnom prostredí. Podľa autorov v [11] sú pocity pohltenia a zapojenia do virtuálneho prostredia dôležité, keďže oba tieto pocity sú potrebné pre navodenie prítomnosti vo virtuálnom prostredí. Ako hlavné vlastnosti zlepšujúce tieto faktory uviedli prítomnosť stimulov na rôzne aktivity používateľa z vnútra virtuálneho prostredia (napríklad vo forme reprezentácie pacientovej ruky alebo haptickej odozvy), pohodlie nosenia VR helmy, udržanie pozornosti používateľa, izolovanosť od vonkajšieho (nesimulovaného) prostredia.

Podľa výsledkov v [10], pacienti, ktorí boli zaradení do skupiny s motorickými zraneniami prejavovali najnižšiu úroveň uspokojenia s VR cvičením. Odchýlka ale nebola príliš signifikantná. Výsledky ukázali, že pohlcujúce VR prostredie môže mať priaznivé účinky na úroveň relaxácie a motivácie vykonávať dané cvičenia. Zároveň pacienti prejavili aj väčší záujem o samotnú VR rehabilitáciu, čo následne znova posilnilo aj ich pocit uspokojenia z rehabilitácie vo VR prostredí.

Zlepšenie týchto aspektov je možné dosiahnuť aj ďalšími spôsobmi, ako napríklad interakciou s terapeutom priamo vo virtuálnom prostredí alebo použitím rôznych náukových videí vytvorených za pomoci 3D technológií [10]. Ďalším spôsobom, ktorý môže zlepšiť spokojnosť a zvýšiť záujem pacienta o cvičenie je použitie herných scenárov [12]. Takéto scenáre môžu uľahčiť cvičenie, keďže pacient často nemusí mať pocit, že vykonáva nejakú nudnú opakujúcu sa aktivitu, namiesto toho vykonáva niečo zmysluplné, čo ho posúva ďalej v danej hre.

Ďalším faktorom je aj dostupnosť cvičenia, keďže množstvo rehabilitácií je vykonávaných v špecializovaných zariadeniach pod dozorom doktorov [12]. Rehabilitácia vykonávaná vo VR by mohla byť v budúcnosti ľahko dostupná aj na veľké vzdialenosti, čo by mohlo pre množstvo pacientov znamenať oveľa jednoduchší prístup k rehabilitácii, a to vo výsledku znamená zlepšenie ich života.

Pocit nevoľnosti zo simulácie

Významným problémom spájaným s virtuálnou realitou je pocit nevoľnosti pri používaní VR helmy. V spojení s VR sa hovorí o konkrétnom type kinetózy - nevoľnosť zo simulácie [13] (anglicky simulation sickness). Tento druh nevoľnosti nie je spôsobený reálnym pohybom, napríklad v porovnaní s nevoľnosťou pri cestovaní (anglicky motion sickness), ale vizuálnym vnemom zobrazujúcim pohyb, aj napriek tomu, že fyzicky žiaden pohyb vykonaný nebol. [14]. Tento pocit môže byť spôsobovaný viacerými faktormi. Spoločnosť Oculus vytvorila príručku [14], v ktorej opisuje rady, ako najlepšie vyvíjať aplikácie pre VR. Z dokumentu je možné vymenovať niekoľko hlavných faktorov, ktoré ovplyvňujú úroveň komfortu pri používaní helmy, a to:

- **zrýchlenie pri pohybe** - vnímanie zrýchlenia bez skutočného zrýchľovania pôsobí negatívne na človeka,
- **pohyb hlavy** - napríklad mierny pohyb hore a dole pri chôdzi je pri používaní VR helmy nežiadúci,
- **pohyb do strán** - v reálnom živote sa málo kedy ľudia pohybujú do strán, prípadne dozadu, tieto pohyby často spôsobujú nevoľnosť,
- **latencia obrazu** - oneskorenie obrazu voči pohybu, ktorý reálne vykonávame,
- **dĺžka používania VR helmy** - dlhodobé používanie VR helmy bez prestávky,
- **používateľské rozhranie** - ak je prehltené a zároveň obsahuje elementy, ktoré sa často menia,
- **avatar** - prítomnosť avatara zlepšuje pocity používateľa, taktiež je potrebné mapovať pohyb senzorov na pohyb končatín ako sú ruky a nohy,
- **nedobrovoľný pohyb** - napríklad posun používateľa po náraze môže spôsobovať pocit nevoľnosti.

Podľa výsledkov v [15] opakované používanie VR nespôsobuje väčšiu mieru nevoľnosti, a teda je možné ho používať aj opakovane. Používanie VR helmy v kratších intervaloch častejšie taktiež pomáha s takýmito pocitmi nevoľnosti a znižuje náchylnosť človeka na nevoľnosť spôsobenú simuláciou [14].

1.3 Existujúce projekty

Existuje množstvo VR systémov a projektov, ktoré boli v posledných rokoch vytvorené. V poslednej dobe sa v rámci VR objavujú aj pokročilejšie systémy zamerané na kolaboratívne virtuálne prostredia. Jednotlivé systémy sa zameriavajú na rôzne oblasti života. V zásade sa dajú rozlíšiť na:

- viacúčelové prostredia,
- prostredia so špecifickým účelom.

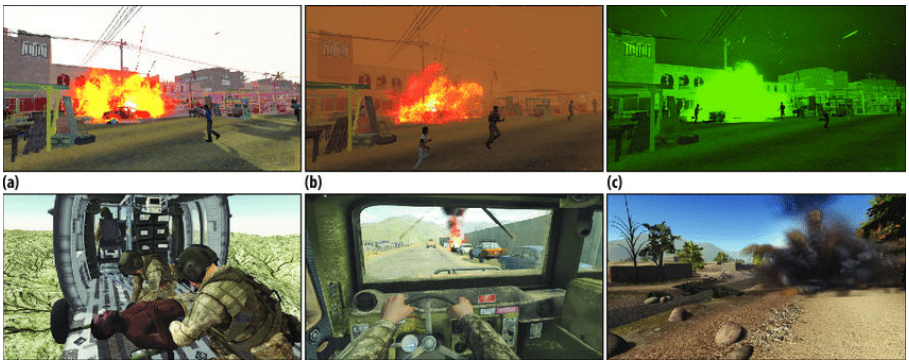
Ďalšou skupinou sú systémy, ktoré sa zameriavajú na terapiu a na liečenie určitých foriem zdravotných porúch. Takéto systémy sa primárne zameriavajú na tieto oblasti [16]:

- psychiatrické poruchy,
- liečba bolesti,
- neurorehabilitácia,
- liečba fóbií,
- liečba úzkostí,
- poškodenie mozgu.

Touto krátkou analýzou niektorých vybraných projektov sme sa snažili zamerať na to, aké projekty už existujú, aké majú vlastnosti a prípadné výhody a nevýhody.

Bravemind [17] je VR systém určený na liečenie post-traumatickej stresovej poruchy. Systém sa primárne zameriava na vojakov a liečenie porúch spôsobených nasadením vo vojnových oblastiach súvisiacich s bojom. Systém staval na už existujúcom systéme Virtual Iraq z roku 2007, ktorý sa ukázal ako úspešný. Aplikácia vyobrazená na obr. 1.1 ponúka viacero scenárov a zameriava sa aj na iné oblasti a traumy, ktoré môžu vojaci zažiť vo svojej práci. Systém je určený na

použitie súčasne jedným pacientom, teda neobsahuje funkcionalitu zdieľaného prostredia. Jedným z hlavných dôvodov, prečo takýto projekt vznikol, bolo rastúce množstvo mentálnych problémov u vojnových veteránov [18]. Ukážka záberov z rôznych scenárov je vyobrazená na obr. 1.1. Systém je vytvorený pomocou herného rámca Unity.



Obr. 1.1: Ukážka systému BRAVEMIND [17]

Hubs[19] je chatovacia miestnosť vytvorená vo VR organizáciou Mozilla Foundation. Aplikácia je založená na technológii WebVR. Podporuje veľké množstvo headsetov, aplikáciu je možné používať na stolnom počítači ale aj na mobile. Vďaka tomu, že využíva štandard WebVR, je možné aplikáciu používať v prehliadači bez nutnosti inštalácie aplikácie. Ukážka vzhľadu aplikácie vo webovom prehliadači je na obr. 1.2. Aplikácia je zameraná na tvorbu privátnych miestností, ktoré je možné následne používať na komunikáciu. Projekt je taktiež podporovaný komunitou, keďže sa jedná o open source riešenie[19]. Aplikácia ponúka aj prispôsobenie miestností, no okrem komunikácie neponúka širšiu funkcionalitu.

Obr. 1.2: Hubs - webové rozhranie

IPčko je internetová psychologická poradňa, ktorá ponúka pomoc najmä mladým ľuďom, ktorý majú nejaké problémy a nevedia nájsť pomoc. V roku 2021

oznámili svoj nový projekt - **psychologické laboratórium virtuálnej reality**[2]. Ich cieľom je využiť technológie virtuálnej reality na poskytovanie rôznych foriem pomoci a tak rozšíriť svoje pôsobisko.

Aplikácia **Coven**[3] predstavuje platformu na kolaboratívnu spoluprácu vo virtuálnom prostredí. Aplikácia je určená na viacero použití ako napríklad výučba, online prezentácie, ukážka praktických modelov. Systém je vytvorený v hernom rámci Unity. Celý projekt vznikol ako výsledok spolupráce medzi Moving Medical Media a FIIT STUBA.

LIRKIS G-CVE[20] predstavuje systém, ktorý ponúka VR prostredie zamerané na použitie viacerými používateľmi naraz. Systém je vytvorený využitím komponentovej štruktúry, čím ponúka možnosť rozšíriteľnosti a škálovateľnosti. Komunikácia je založená na klient-server architektúre. Systém je vytvorený využitím JavaScript knižnice A-Frame. Veľkou výhodou je to, že systém je založený na technológii WebVR, a teda je funguje v prehliadači bez nutnosti inštalácie softvéru. Vďaka tomu, že systém je modulárny je možné ho použiť na rôzne účely, ako napríklad: výučbu, virtuálne prehliadky, medicínske účely. Tento systém je použitý aj v implementácii riešenia určeného na rehabilitáciu pacientov po mozgovej príhode. Tento projekt je detailnejšie rozobraný v kapitole 2, keďže táto práca nadväzuje na už dosiahnuté výsledky v tomto projekte.

Preskúmané projekty majú svoje výhody aj nevýhody. Využívajú rôzne prístupy a taktiež aj rôzne technológie, v ktorých sú vytvorené. Pri návrhu nášho systému je užitočné využiť poznatky z už existujúcich systémov, aby sme prípadne predišli problémom, ktoré by sme inak nemuseli spozorovať. Zamerali sme sa na viacero projektov, z ktorých sú všetky v niečom odlišné a každý z nich má svoje špecifiká. Taktiež je potrebné sa riadiť podľa pokynov, ktoré boli spomenuté v kapitole 1.2, aby sme mohli používateľom našej aplikácie poskytnúť čo najlepší zážitok pri jej používaní.

[2]https://ipcko.sk/psychologicke-laboratorium-virtualnej-reality/

[3]https://www.stuba.sk/sk/diani-na-stu/slovensky-projekt-vytvoril-kolaborativne-virtualne-prostredia-nova-dimenzia-vzdelavania-podnikania-a-spoluprace-vo-virtualnej-realite-touchit.sk.html?page_id=13669

2 Aktuálny stav projektu

Táto diplomová práca je pokračovaním série diplomových prác, ktoré sa zameriavajú na vytvorenie kolaboratívneho virtuálneho systému, so zameraním na neurorehabilitáciu pacientov po mozgovej príhode. Práca pokračuje dopĺňaním systému, ktorý už bol pred tým vyvíjaný študentmi. Ako posledné pracovali na systéme Sára Javorková[21] a Jana Gvuščová[22].

2.1 Aktuálna implementácia

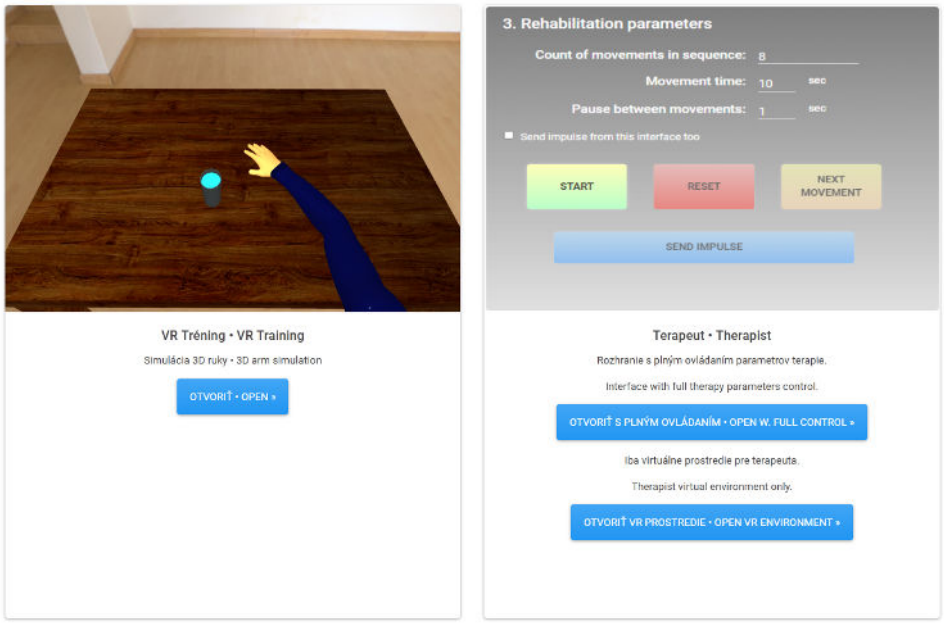
Systém je vybudovaný v jazyku JavaScript, pomocou frameworku A-Frame, ktorý umožňuje tvorbu 3D a VR aplikácií využitím technológie Web-VR [23]. Vďaka tomu, je jeho použitie podporované na širokom množstve VR headsetov, ako aj vo webovom prehliadači.

V systéme už sú implementované tieto časti:

- používateľské rozhranie pacienta,
- používateľské rozhranie terapeuta,
- komunikácia pomocou protokolu easyRTC umožňujúca prenos zvuku,
- animácie pohybu ruky
- objekty a ukážky pohybov, ktoré je potrebné vykonať.

2.1.1 Používateľské rozhrania

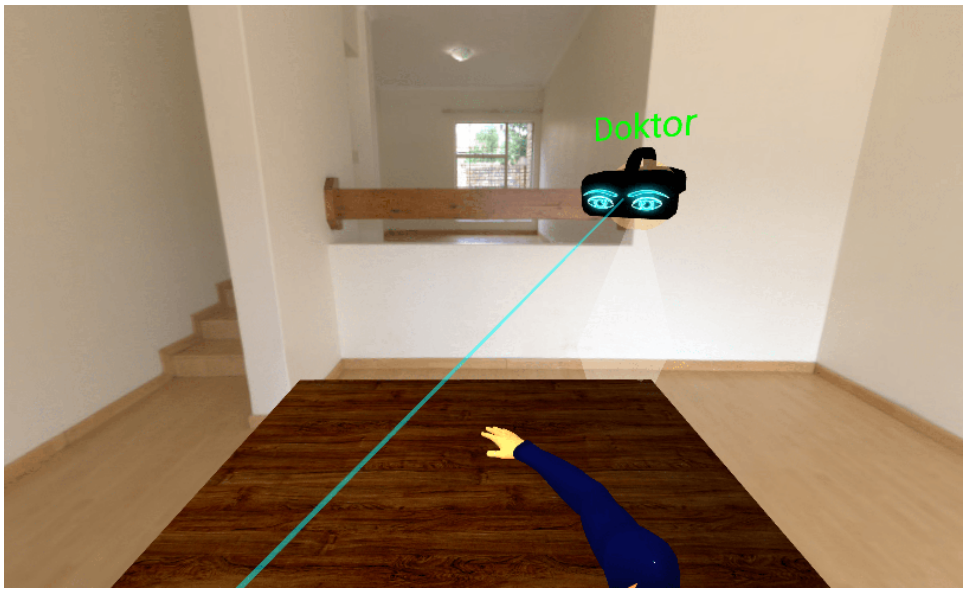
Ako už bolo spomenuté, používateľské rozhrania sú rozdielne z dôvodu, že je potrebné zabezpečenie rôznej funkcionality [21]. V skutočnosti existujú až 3 rôzne rozhrania, ako je možné vidieť na obr. 2.1.



Obr. 2.1: Možnosti zobrazenia rozhrania

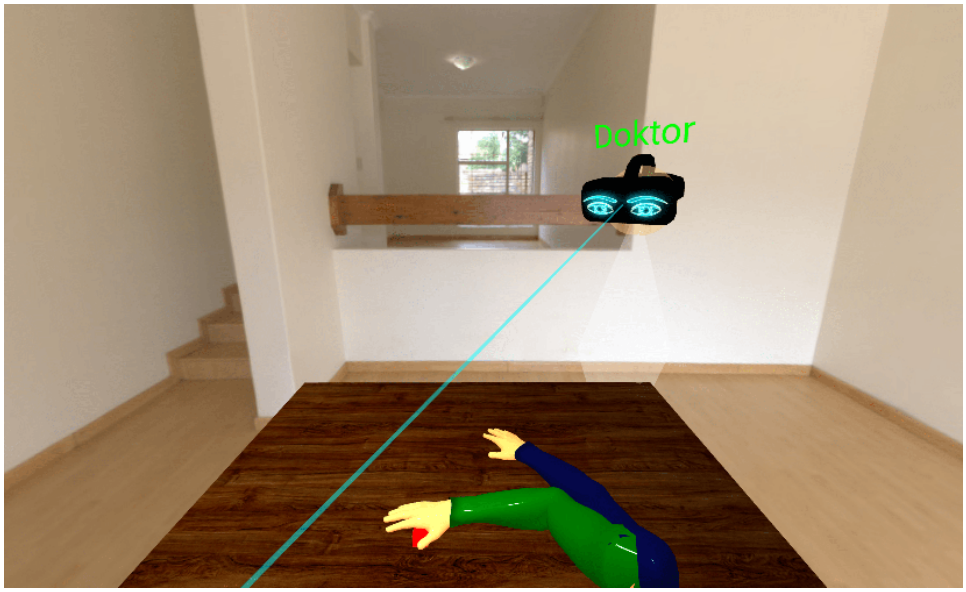
Rozhranie - VR tréning

Toto rozhranie je vytvorené pre pacienta. Pacient tu môže vidieť "svoju" virtuálnu ruku. Ak terapeut spustí ukážku pohybu, zobrazí sa mu animácia a bude mať určitý čas na jej vykonanie.



Obr. 2.2: Rozhranie pacienta

Ako môžeme vidieť na obr. 2.2, pacient sa nachádza v miestnosti, okrem svojej virtuálnej ruky nič iné nevidí. Čaká na terapeuta, pokiaľ nedá pokyn na vykonanie pohybu. Ukážku pohybu je možné vidieť na obr. 2.3. Ukážka je zobrazená



Obr. 2.3: Rozhranie pacienta - ukážka pohybu chytenia kocky

inou farbou. Po vykonaní ukážky animácie má pacient určitý časový interval, v ktorom sa má pokúsiť si mentálne predstaviť reálny pohyb. Tento pokus o predstavenie pohybu je následne zaznamenaný EEG zariadením a správa je poslaná na rozhranie servera. Ten pošle informáciu všetkým používateľským rozhraniam (pacient aj terapeut). Následne sa zobrazí aj animácia vo virtuálnom svete.

Rozhranie - Terapeut

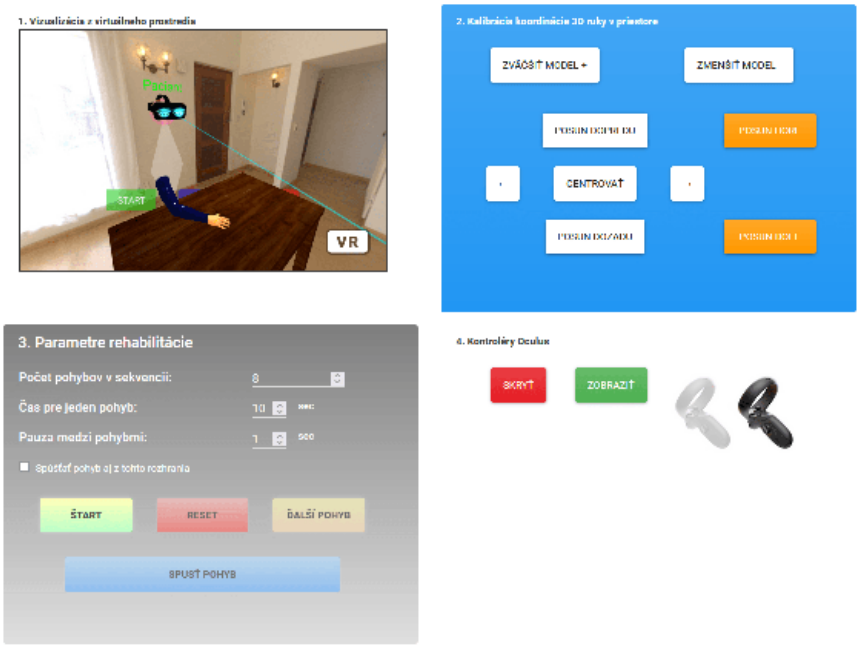
Terapeut je vo svojom rozhraní schopný spúšťať ukážky pohybov, zastaviť pohyb alebo spustiť ďalší pohyb. Na výber je z viacerých možných animácií:

- zdvihnutie a posunutie kocky,
- chytenie pohára,
- zdvihnutie kľúča a odomknutie zámky.

V rozhraní terapeuta s úplnou kontrolou môže terapeut meniť nastavenia pre animácie. Je možné nastaviť dĺžku čakania na pohyb pacienta, počet animácií, trvanie animácií, ovládať postupnosť animácií. Terapeut má ešte aj možnosť rozhrania bez týchto pokročilých nastavení. Toto rozhranie je prispôsobené na použitie v spojení s VR helmou(anglicky VR headset) (obr. 2.5) [21].

2.1.2 Komunikácia a zabezpečenie sieťového spojenia

Sieťovú komunikáciu zabezpečuje framework Networked-Aframe, ktorý je nadstavbou k A-Frame. Framework poskytuje niekoľko protokolov, pomocou ktorých



Obr. 2.4: Rozhranie terapeuta s ovládaním nastavení

môžeme poskytnúť sieťovú komunikáciu. Knižnica funguje na princípe klient-server. Server zabezpečuje spojenie medzi klientmi a poskytuje im "virtuálny priestor". V aplikácii sú použité adaptéry socketio na zabezpečenie websocket-ov [24] a easyrtc [1] zabezpečujúce prenos audia.

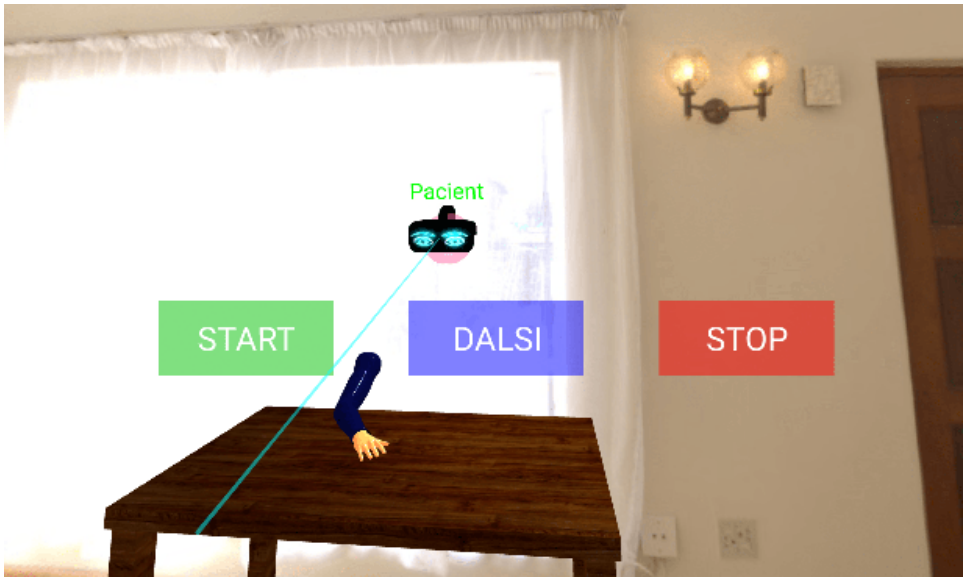
2.1.3 Animácie pohybu

Animácie pohybov sú implementované za pomoci algoritmu výpočtu inverznej kinematiky [21]. Výpočet spočíva v tom, že ruka je rozdelená do niekoľkých kostí. Každej kosti sa vypočíta pozícia v ktorej by sa mala nachádzať na konci animácie a vypočítajú sa aj uhly, ako sa majú jednotlivé kosti otáčať. Takto sa postupne vypočítajú pozície pre všetky kosti a pre rôzne postupné pozície. Na výpočet uhlov sa používa algoritmus na výpočet uhlov v trojuholníku[21]. Výpočet má implementovanú aj kontrolu, či je možné pohyb vykonať vzhľadom na vzdialenosť objektov.

2.2 Analýza implementácie

Vzhľadom na to, že riešenie už bolo niekoľko krát rozšírené o rôzne funkcie, sme vykonali analýzu kódu, aby sme si mohli utvoriť obraz toho, v akom stave sa riešenie nachádza. Zamerali sme sa hlavne na to, ako bolo riešenie navrhnuté a aké

[1] https://webrtc.org/



Obr. 2.5: Rozhranie terapeuta bez nastavení

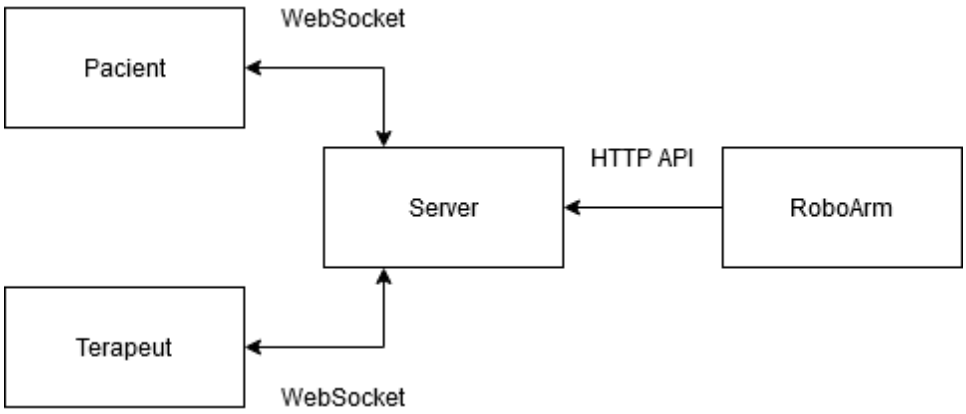
možnosti rozšírenia poskytuje. Tento krok považujeme za podstatný z hľadiska toho, že ak chceme systém ďalej rozširovať, nie je možné tak učiniť bez dostatočnej znalosti riešenia. Počas vývoja aplikácie taktiež nie je využívaný systém na správu verzií, čo znamená, že nazretie do starších verzií aplikácie je nemožné.

2.2.1 Stav kódu

Riešenie má z hľadiska kódu niekoľko problémov. Kód obsahuje súbory, ktoré už nie sú používané. Niektoré časti kódu sa spoliehajú na určitý podsystém, iné časti používajú ďalšiu technológiu, aj keď obidva by bolo možné upraviť, aby boli jednotné. Ako ďalší problém sme si všimli miešanie rôznych štýlov programovania. Vzhľadom na to, že na projekte pracovalo niekoľko ľudí, je logické, že sa tam budú miešať rôzne štýly programovania. V niektorých častiach kódu je použitý framework JQuery, inokedy zasa Javascript ES6. To spôsobuje, že kód je v niektorých častiach neprehľadný, rozdelenie funkcionality do súborov je často zmätočné, niekedy je JavaScript kód v samostatných súboroch, inokedy je množstvo funkcionality naraz v jednom súbore, niekedy je kód priamo v html súboroch. To spôsobuje ťažkú prácu s kódom a jeho neprehľadnosť. V niektorých častiach kód obsahuje pomerne málo komentárov. To výrazne sťažuje prácu s ním. V prípade, že by na projekte pracoval len jeden človek, nebol by to až taký problém, ale vzhľadom na to, že projekt je výsledok práce viacerých ľudí, je ťažké sa v ňom orientovať.

2.2.2 Komunikácia modulov

Ako už bolo spomenuté, systém je rozdelený do niekoľkých modulov, ktoré spoločne komunikujú. Na komunikáciu sa používa primárne websocket ale aj HTTP API. Na vytvorenie tejto funkcionality je použitý webový framework Express pre Node.js [21]. Štruktúra komunikácie je načrtnutá na obr. 2.6. V tomto prípade sa pod modulom Robo Arm myslí zariadenie komunikujúce s pacientom a sledujúce jeho aktivitu. Toto zariadenie informuje server o tom, či sa pacient pokúsil vykonať pohyb. Keďže je komunikácia riešená pomocou websocketov, je možné



Obr. 2.6: Ukážka komunikácie medzi modulmi

prakticky posielať čokoľvek a následne sa registrovať na rôzne akcie. Ako problém sme si všimli to, že niektoré správy sa posielajú a nijak sa na nich nereaguje, prípadne sú niektoré aktivity vykonané viackrát. Tento problém sme si všimli v prípade, že kód, ktorý spracúva reakciu na správu je rozdelený vo viacerých súboroch. Je možné, že tento kód bol pridaný neskôr.

2.2.3 Animácie pohybu

Vzhľadom na to, že animácie sú počítané pomocou inverznej kinematiky, ich výpočet je niekedy problematický. Aktuálne sú dostupné tri rôzne scenáre. Problém ale nastáva, ak by sme dané scenáre chceli meniť alebo rozširovať. Kód je napísaný na prvý pohľad tak, že je možné meniť pozície objektov, a tak vytvoriť rôzne variácie. V skutočnosti je to ale pomerne zložité a ťažkopádne. Množstvo animácií je vytvorených tak, aby fungovali na určité prípady, no je ťažké ich upravovať. Príkladom môže byť zdvihnutie a posunutie kocky, ktoré sa nám nepodarilo upraviť tak, aby kocka mala správne začiatočné a koncové pozície. Keďže daný kód obsahuje málo komentárov, je pomerne ťažké pochopiť, ako dané časti fungujú.

3 Prototyp riešenia

Za účelom využitia viacerých zdrojov a vytvorenia zaujímavejšieho virtuálneho prostredia, sme vytvorili prototyp za pomoci herného rámca Unity. Cieľom tohto prototypovania bolo použitie technológie, ktorá je už dlhodobo využívaná na tvorbu 3D prostredí, a ktorá by nám umožnila rozvíjať systém do budúcna. Pomocou prototypu dokážeme relatívne rýchlo zistiť, či nami zvolená technológia je schopná zabezpečiť všetku potrebnú funkcionalitu a či sa vôbec vyplatí v nej pracovať. Je dôležité si na záver zhrnúť všetky výhody aj nevýhody zvolenej platformy.

3.1 Unity XR

Unity[25] je platforma, primárne určená na vývoj hier. Podporuje vývoj 2D, 3D, VR aplikácií, ktoré je možné používať na rôznych platformách ako sú PC, mobily, konzoly, web. Medzi jej hlavné výhody patrí:

- veľmi kvalitná dokumentácia,
- veľká komunita,
- množstvo balíčkov, ktoré sú dostupné buď na zakúpenie alebo zadarmo,
- vysoká podpora rôznych štandardov,
- už pred-vytvorené prvky,
- editor.

Unity ponúka nástroje na prácu s VR, rôzne komponenty, jednoduché rozhrania na interakciu s objektmi a ich ovládanie. Problém by tu mohol tvoriť fakt, že by bolo potrebné vyvíjať aplikáciu na niekoľko rôznych zariadení ako napríklad Oculus Quest, HTC Vive, PlayStation VR. Tento problém môžeme vyriešiť pluginom OpenXR [26] vyvíjaným skupinou Khronos [1]. Tento štandard sa snaží

[1] https://www.khronos.org/openxr/

zjednotiť vývoj aplikácií pre rôzne zariadenia a aj platformy, použitím jednotného rozhrania vystupujúcim ako adaptér. To nám síce ponúka jednoduchšiu prácu s rozdielnymi platformami, zároveň to môže obmedziť možnosť rôznorodého použitie jednotlivých platforiem a využitie ich špecifických funkcií naplno. Vzhľadom na to, že my takúto funkciu nepožadujeme, OpenXR pre nás predstavuje ideálnu voľbu.

3.2 Požiadavky na minimálny prototyp

V stave prototypovania je potrebné určiť si ciele, ktoré musíme dosiahnuť. Vďaka tomu dokážeme zistiť, či sú dané technológie použiteľné na vývoj kompletnej aplikácie. Zároveň je už aj počas tejto fázy potrebné myslieť na možné budúce rozširovanie funkcionality. Za minimálne požiadavky, ktoré je potrebné splniť, sme si určili tieto ciele:

1. VR funkcionalita,
2. kolaboratívne prostredie,
3. komunikácia aplikácie s prostredím Robo Arm, prípadne OpenVibe,
4. podpora rozdielnej funkcionality podľa rolí - pacient, terapeut.

1. - funkcionalitu VR vieme zabezpečiť použitím balíčka XR Interaction toolkit[2]. Ten poskytuje množstvo už vytvoreným komponentov pre snímanie pohybov, interakciu s prvkami ako aj možnosti lokomócie.

2. - Unity nám implicitne neposkytuje žiadne zdieľané prostredie. Preto je potrebné použitie nejakej knižnice. Unity má svoj vlastný balík s názvom NetCode [3], ktorá je ešte veľmi mladá a má pomerne malú podporu zo strany komunity. Ďalšími riešeniami môžu byť balíky Mirror [4], Photon PUN2, DOTS. Samotné Unity vydalo článok na svojom blogu [5], kde je možné nájsť zhrnutie a aj odporúčanie podľa čoho si vybrať. Vzhľadom na pomerne veľkú komunitu a históriu sme si zvolili použitie knižnice Mirror. Pomocou nej môžeme vytvoriť sieťovú infraštruktúru a zabezpečiť sieťovú komunikáciu.

3. - vzhľadom na to, že infraštruktúra môže byť rozložená na niekoľkých zariadeniach, je potrebná komunikácia za účelom synchronizácie akcií a riadenia.

[2]https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.3/manual/
[3]https://docs-multiplayer.unity3d.com/
[4]https://mirror-networking.com/
[5]https://blog.unity.com/technology/choosing-the-right-netcode-for-your-game

4. - rozdelenie funkcionality je možné vykonať buď použitím rôznych objektov podľa zadanej funkcionality a takto odlíšiť pacienta od terapeuta, alebo použitím rolí a následným vypínaním a zapínaním určitých komponentov podľa potreby.

Podľa našich zistení je hlasová komunikácia nižšia priorita, keďže aktuálne je zamýšľané riešenie také, že aj pacient aj terapeut sa nachádzajú v jednej miestnosti, teda nepotrebujú spoločne komunikovať na diaľku. V prípade záujmu o implementáciu je možné zabezpečiť ju knižnicami ako je napríklad Dissonance alebo Photon Voice 2. Dissonance je možné spojiť priamo s knižnicou Mirror, Photon Voice 2 si vyžaduje samostatnú inštanciu a aplikáciu vytvorenú na oficiálnej stránke[6]. Prípadne je možné samostatné vytvorenie Photon servera a hosťovanie na vlastnom stroji.

3.2.1 Sieťová infraštruktúra

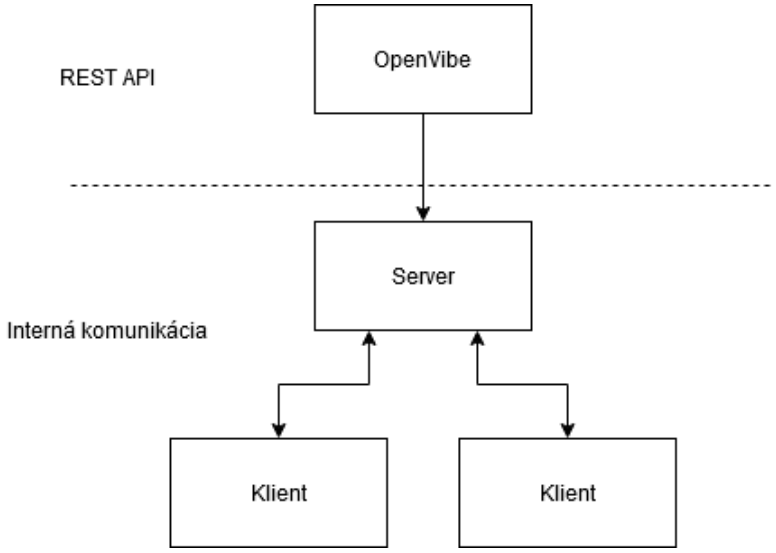
Mirror nám poskytuje prostriedky na vytvorenie sieťového modelu klient-server. Aktuálne riešenie v A-Frame svojím spôsobom takisto poskytuje určitú formu klient-server infraštruktúry na nadviazanie spojenia. V prípade balíčka Mirror musia klienti komunikovať so serverom, ten následne sprostredkuje informáciu všetkým klientom. Vďaka takejto infraštruktúre môžeme jednoducho ovládať, aké správy sú posielané, kedy a komu. Ak by sa napríklad klient#1 snažil vykonať akciu, ktorú môže vykonať len terapeut, no on má inú rolu, server by ju mohol jednoducho ignorovať. Vlastnosťou Mirror-u je aj to, že nie každý objekt musí byť zdieľaný objekt, takže niektoré objekty môžu byť špecifické len pre určitých používateľov, čo ďalej uľahčuje rozdelenie podľa rôznych funkcií.

3.2.2 Komunikácia s OpenVibe prostredím

Navrhovaný model komunikácie počíta s použitím REST API na zabezpečenie komunikácie, podobne ako tomu je aj v aktuálnom projekte vytvorenom pomocou A-Frameu. Problémom pri tomto type komunikácie je skutočnosť, že nie je možné komunikovať smerom k prostrediu OpenVibe, keďže na tejto strane systému nie je naprogramované REST API. Spôsob komunikácie je vyobrazený na obr. 3.1.

Do budúcnosti by bolo užitočné prerobiť komunikáciu, aby používala sockety. Tento typ komunikácie nám umožní komunikovať oboma smermi. Na to je ale potrebné vytvoriť aj aplikáciu zabezpečujúcu funkciu klienta, ktorá by bola kompatibilná s prostredím OpenVibe. Táto strana systému ale nie je v našej réžii, a

[6]https://www.photonengine.com/voice



Obr. 3.1: Komunikácia medzi našim systémom a prostredím OpenVibe

teda zatiaľ bude použitá komunikácia pomocou REST API.

3.2.3 Synchronizácia klientov

Knižnica Mirror stavia na opustenom sieťovom riešení priamo od Unity - UNet. Medzi funkcionalitu, ktorú knižnica ponúka patrí viacero rôznych transportných protokolov. Poskytuje API rozhranie na komunikáciu medzi klientom a serverom, možnosť pracovať so zdieľanými objektami a aj ďalšie nástroje potrebné na zabezpečenie sieťovej komunikácie.

Pri vytváraní zdieľaných objektov je každému objektu priradené unikátne ID, ktoré ho reprezentuje v scéne a server si drží záznamy o všetkých takýchto objektoch. Medzi takéto objekty patrí aj objekt používateľa. Podstatnou vlastnosťou zdieľaných objektov je aj to, ako sú rozdelené práva na riadenie objektov a kto je ich vlastníkom. Tento pojem sa označuje ako sieťová autorita (anglicky Netowrk Authority). Vo všeobecnosti sa v hrách odporúča použitie serverovej autority na objektoch z dôvodu, aby nemohli klienti ovládať ich parametre, a tak prípadne podvádzať. V našom prípade nejde o hru, a teda ani neočakávame, že by sa niekto snažil podvádzať, keďže nie je v čom. V niektorých prípadoch je pre nás lepšie a jednoduchšie zvoliť klientsku autoritu. Najlepšie to možno vysvetliť na príklade pohybu používateľa. V prípade serverovej autority by informácia o snahe pohnúť sa bola poslaná najprv na server, následne by server overil, či je pohyb validný, ak áno, server by upravil pozíciu používateľa u seba a táto informácia by sa následne synchronizovala na všetkých klientoch. V prípade klientskej autority sa pohyb vykoná priamo na klientovi, informácia sa synchronizuje na serveri a následne

na ostatných klientoch. V tomto prípade server nevykonáva žiadnu kontrolu ani overenie.

3.2.4 Viditeľnosť objektov z pohľadu jednotlivých používateľov

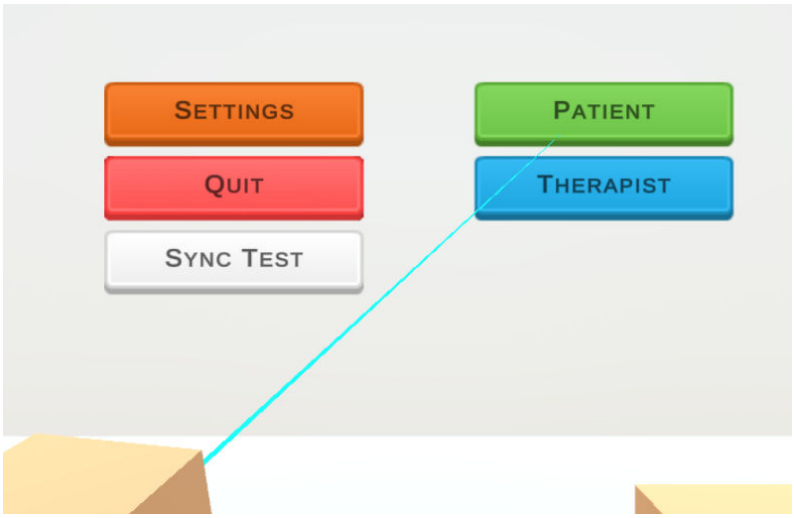
Naším cieľom je navrhnúť systém tak, aby sa pacient aj terapeut nachádzali v tom istom virtuálnom prostredí. Následne by mohli spolu interagovať, terapeut by mohol priamo viesť pacienta. Cieľom je, aby terapeut a pacient spolupracovali čo najprirodzenejšie, prostredníctvom avatarov. V prípade aktuálneho riešenia vytvoreného pomocou JavaScript-u je rozdielne rozhranie zabezpečené využitím dvoch rôznych skriptov pre terapeuta a pacienta.

V našom prípade nie je takéto riešenie žiadúce. Dôvodom je hlavne to, že obaja používatelia sa nachádzajú v spoločnej scéne, ktorá je rovnaká pre oboch. Riešenie viditeľnosti určitých objektov (ako napríklad menu pre terapeuta) je možné implementovať dvoma spôsobmi:

- vytváranie objektov lokálne po pripojení používateľa,
- skrývanie/odkrývanie objektov na základe rolí.

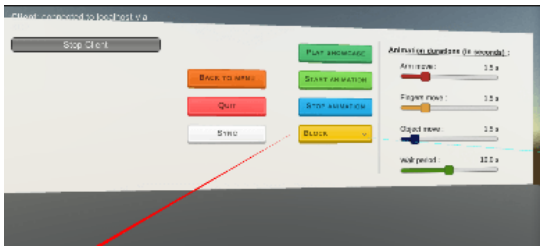
V našom prípade sme sa rozhodli ísť druhým spôsobom. Tu je možné použitie dvoch prístupov. Využitie vrstiev, ktoré ponúka Unity a následné nastavenie vrstiev, ktoré majú byť vynechané z vykresľovania alebo implementácia vlastnej funkcionality skrývania objektov. Princíp, ktorý sme implementovali, je založený na tom, že pri pripojovaní si používateľ zvolí svoju rolu (obr. 3.2). Následne sa v scéne nastaví komponent, ktorý bude obsahovať informáciu o roli používateľa. Takýto princíp je užitočný aj v iných prípadoch. Informáciu z komponentu môžeme ďalej využívať pri pripojovaní užívateľa, kde môžeme rozhodnúť, aký model avatara je potrebné zvoliť na základe roli používateľa. Samotný avatar môže teda obsahovať rôzne komponenty a objekty vzhľadom na to, či sa jedná o pacienta alebo terapeuta.

Ďalej sme už len implementovali komponent, ktorý si pri vytvorení objektu skontroluje, o akú rolu sa jedná a na základe toho objekt, ktorého je tento komponent súčasťou, skryje. Túto funkcionalitu sme implementovali tak, aby sa jednotlivé objekty iba skryli, no zostali aktívne. Dôvodom je to, že niektoré objekty môžu obsahovať funkcionalitu, od ktorej požadujeme, aby bola synchronizovaná na používateľoch s rôznymi rolami. Ukážka rozdielneho rozhrania je vyobrazená na obr. 3.3. V tomto prípade terapeut vidí tlačidlá na nastavovanie animácií, no



Obr. 3.2: Výber roli používateľa

pacient vidí len hlavné menu. Aj napriek tomu, že toto menu nevidí, zostáva aktívne a jeho hodnoty sú synchronizované. Vďaka tomu môžeme kódom nastavovať určité premenné. V našom prípade tomuto menu posielame referenciu na samotného pacienta, aby sme mohli spúšťať animácie.



(a) Pohľad terapeuta na UI

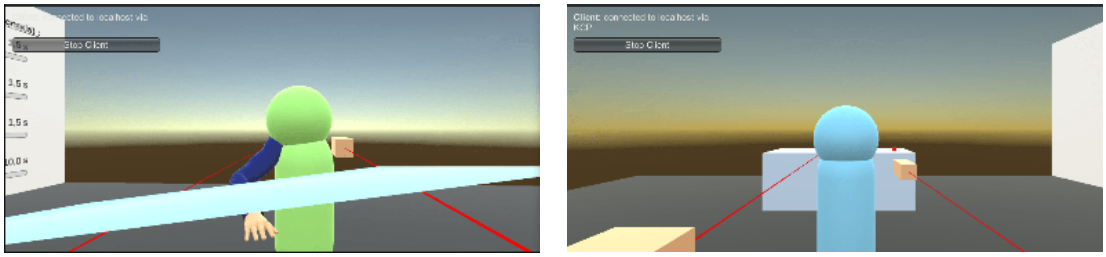
(b) Pohľad pacienta na UI

Obr. 3.3: Rozdielne UI

Na obr. 3.4 je možné vidieť využitie rolí na vytvorenie rozdielnych avatarov pre používateľov. V našom prípade sme okrem farby avatara zmenili aj to, že súčasťou avatara pacienta je zároveň aj ruka, ktorá bude animovaná pomocou inverznej kinematiky. Tieto odlišné roly sú potrebné aj preto, že z hľadiska servera sú klienti jednotní. Server nesleduje rozdiely v použití avatara alebo iného komponentu. My sami musíme túto funkcionalitu zabezpečiť.

3.3 Testovanie prototypu

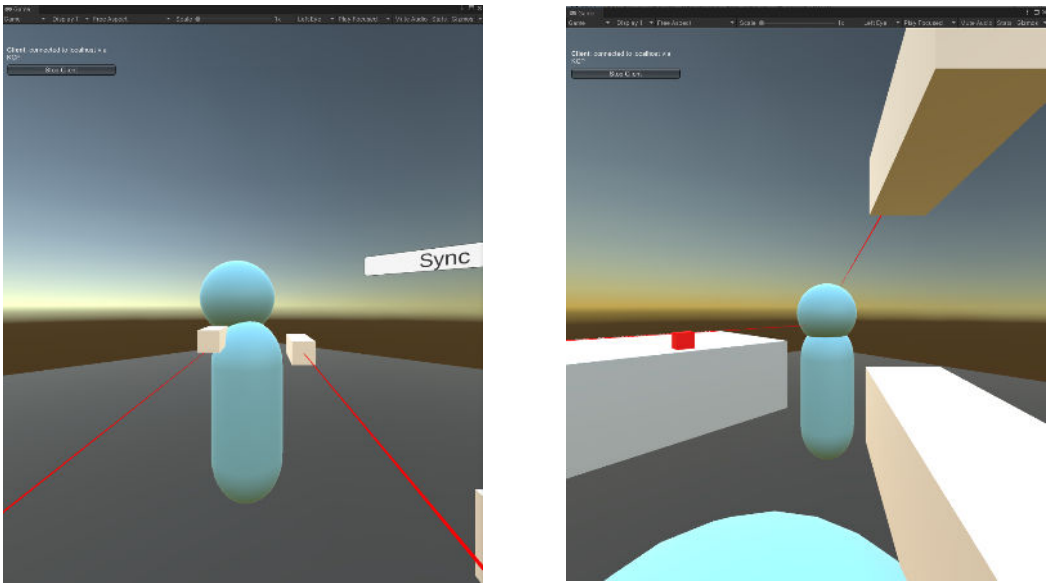
Ideálny scenár pre vytváranie aplikácií pre VR je ich okamžité testovanie pomocou VR helmy. Nie vždy to ale je možné. Unity XR modul poskytuje funkcionalitu použitia simulovanej VR helmy. Takýmto spôsobom vieme simulovať zariadenie



(a) Avatar pacienta (b) Avatar terapeuta

Obr. 3.4: Využitie rozdielnych avatarov

a ovládače. Unity naviac poskytuje simulátor, ktorý už má predvytvorené rozhranie na simulovanie tlačidiel a pohyb ovládačov pomocou klávesnice. Využitím balíka ParrelSync môžeme spustiť viacero inštancií a použitím simulátora ich môžeme otestovať aj na jednom počítači bez použitia VR helmy. Na obr. 3.5 je možné vidieť ukážku 2 inštancií, ktoré môžu vidieť jedno spoločné prostredie, spustené na jednom PC. V prostredí je synchronizovaný pohyb používateľov, pohyb hlavy, prostredie obsahuje aj kocku, s ktorou môžu používatelia hýbať a jej poloha je synchronizovaná.

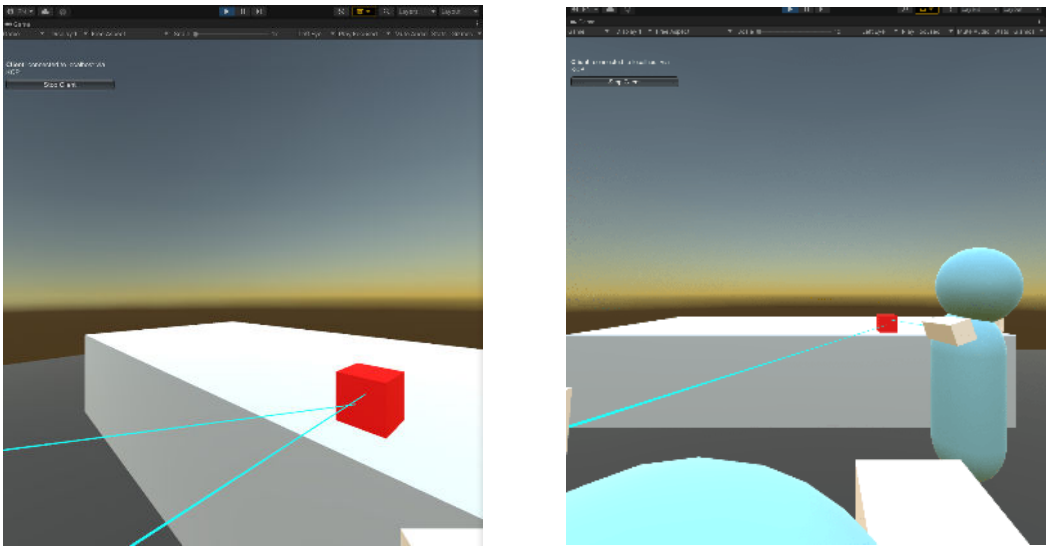


(a) Pohľad prvého používateľa (b) Pohľad druhého používateľa

Obr. 3.5: Pohľad 2 hráčov na seba

Problémom je to, že pohyb rúk nie je možné synchronizovať len tak jednoducho (čo je možné vidieť na obr. 3.5) . Na to, aby sa správne synchronizovala ich poloha a rotácia, je nutné vypnúť určité komponenty, prípadne vypnúť určitú funkcionalitu tých komponentov. Dôvodom je spôsob, akým si jednotlivé komponenty vytvárajú väzby na niektoré časti kódu, a tak sa jednotlivé referencie

nesprávne prepíšu na klientoch. Správne synchronizovaný pohyb rúk je možné vidieť na obr. 3.6.



(a) Pohľad prvého používateľa (b) Pohľad druhého používateľa

Obr. 3.6: Pohľad 2 hráčov na seba so synchronizovaným pohybom rúk

Dôvod prečo to spomíname je ten, že na synchronizáciu pohybu objektu XR rigu je možné len pridať komponent do objektu avatara používateľa, pri synchronizácii pohybu rúk a hlavy bolo potrebné vypnúť komponent pri vytvorení jeho inštancie za podmienky, že sa nejedná o lokálneho používateľa. Všetky objekty sú vytvárané a inštancované u každého klienta samostatne. Ak niektorý komponent obsahuje kód, ktorý by mohol zmeniť hodnoty potrebné pre správne fungovanie klienta, je potrebné si dať pozor, aby to nespôsobilo neočakávané správanie, ako v tomto prípade. Často pri spájaní niekoľkých balíkov a knižníc je ťažké povedať, ktorý konkrétny komponent je nutné vypnúť, aby bolo zabezpečené správne fungovanie. Tu je veľmi užitočné použitie inšpektora, ktorý sa nachádza v Unity a skúšať komponenty vypínať, dokiaľ to nebude fungovať. Aj napriek tomu, že to nie je práve najintuitívnejší spôsob, niekedy je to jediná cesta, ako sa dopracovať k žiadanému výsledku. Je dôležité dať si pozor, aby sa nestalo to, že sa zasekneme na určitom probléme niekoľko dní.

3.4 Porovnanie Unity a A-Frame

Vzhľadom na rozdielnosť týchto dvoch technológií je najlepšie si ich porovnať a vyhodnotiť, ktoré aspekty vývoja a práce s danou technológiou nám vyhovujú najviac. V tabuľke 3.1 sa nachádza porovnanie niektorých vlastností týchto dvoch technológií.

	Unity	*A-Frame*
Prototypovanie	pomalšie	rýchlejšie
Cieľové platformy	Windows, Linux, Android, Web	Web
HW nároky na vývoj	vyššie	nižšie (závisí od zložitosti VR prostredia)
Potrebné technické znalosti	vyššie	nižšie (spočiatku, neskôr rovnaké)
Škálovateľnosť	vysoká	obmedzenia platformy
Kompatibilita so zariadeniami	vyššia	nižšia
Vývojové prostredie	výkonný editor	inspector (obmedzená funkcionalita)
Podpora assetov	asset store, množstvo oficiálnych knižníc	menší počet knižníc
Podpora komunity	vysoká	nízka
Nasadenie aplikácie	zložitejšie	jednoduché nasadenie na web (napríklad Glitch alebo Heroku)

Tabuľka 3.1: Porovnanie Unity a A-Frame

Nástroje Unity

Unity samotné ponúka niektoré užitočné nástroje, ktoré vieme použiť na analýzu v prípade nejakého problému. Jedným z týchto nástrojov je **frame debugger**. Frame debugger nám ponúka možnosť analyzovať počet vykresľovaných objektov. Dosiahnutie menšieho počtu vykresľovaní nemusí nutne znamenať lepší výkon, ale toto číslo vieme využiť ako jeden z indikátorov. Jedným z najľahších spôsobov, ako zmenšiť počet volaní je označenie objektov, ktoré sa v scéne nemenia za statické. Unity tieto objekty spojí a vykresľuje ich spoločne ako jeden celok. To má za dôsledok zníženie počtu prechodov medzi jednotlivými volaniami vykresľovania. Jedným z najvýraznejších spôsobov ako znížiť počet volaní na vykresľovanie je zbavenie sa tieňov, prípadne ich obmedzenie.

Unity taktiež ponúka **nástroj na profilovanie** (anglicky Profiler). Profiler je možné použiť na určenie množstva času stráveného na jednotlivých častiach hard-

véru ako CPU, GPU, RAM. Taktiež zobrazuje aj množstvo výkonu, ktorý jednotlivé podsystémy spotrebúvajú. Ak by sa vyskytol problém a nebolo by jasné čo ho spôsobuje, tento nástroj nám ho môže pomôcť objaviť.

XR Interaction toolkit obsahuje **Input debugger**. Ten je možné použiť na analýzu vstupov, ktoré sú rozpoznané. Je možné ho použiť na zistenie, či systém správne rozpoznal jednotlivé akcie, prípadne či ich správne naviazal na klávesové skratky.

Počas spustenej aplikácie Unity taktiež ponúka pomerne detailné štatistiky, ako napríklad množstvo aktuálne vykresľovaných vrcholov a trojuholníkov, počet snímkov za sekundu a ďalšie parametre, ktoré sú užitočné pri analýze výkonu.

Zhodnotenie technológií

Pomocou tabuľky 3.1 je možné rozdeliť tieto dve technológie podľa toho, na čo ich chceme primárne použiť. V prípade, že chceme aplikáciu často a rýchlo meniť, rýchlejšie prototypovať a nasadiť na web, zvolíme A-Frame. Medzi hlavné nevýhody A-Frameu patria obmedzenia internetových prehliadačov (ako napríklad obmedzenia výkonu) a taktiež aj fakt, že dáta ako napríklad modely sú prenášané internetom, nie je možné ich stiahnuť len raz a lokálne načítať z disku.

V prípade, že plánujeme vytvárať robustnú aplikáciu, ktorá bude dlhodobo podporovaná so zložitejšou logikou, použitie Unity sa javí ako lepšia voľba. V prípade Unity ale môžu byť vyššie hardvérové nároky počas vývoja problémom. Vzhľadom na zámer gamifikovať systém a možnosti, ktoré nám Unity ponúka sme sa rozhodli pokračovať vo vývoji systému použitím herného rámca Unity.

4 Návrh a implementácia riešenia

Ešte pred tým, než je možné pracovať na samotnom riešení, je nutné si navrhnúť čo má samotná aplikácia obsahovať, aké technológie budú použité a aké ciele má dosiahnuť. Vzhľadom na to, že sme sa v tejto práci zamerali hlavne na prepísanie aplikácie do Unity, ako zoznam požadovaných funkcií sme mohli použiť už existujúce riešenie. Následne sa zameriame na implementáciu konkrétnych modulov aplikácie.

4.1 Návrh systému a procesov

Po spustení aplikácie sa používateľ nachádza v miestnosti, ktorá slúži na zmenu nastavení a pripojenie používateľa do zdieľaného prostredia. Túto scénu nazývame aj Offline scéna. V tradičných hrách je táto scéna riešená formou tradičného menu, keďže ale aplikácia musí podporovať VR, takéto riešenie nie je možné. Používateľ si môže zvoliť rolu a následne sa pripojí do prostredia, kde už prebieha proces rehabilitácie. Túto scénu nazývame Online scéna.

Vzhľadom na proces terapie v kolaboratívnom prostredí vieme rozdeliť roly, na ktoré sa používatelia budú deliť, na rolu pacienta a terapeuta. V prípade budúceho rozširovania by bolo možné pridať aj ďalšie roly ako napríklad pozorovateľ, to je ale v aktuálnom procese vývoja nepotrebné. Túto funkcionalitu by bolo možné nahradiť použitím roly terapeuta s jediným rozdielom a to tým, že pozorovateľ by nevchádzal do procesu rehabilitácie.

Používateľ s rolou **Pacient** by mal mať obmedzenú funkcionalitu, jeho úlohou je nasledovať pokyny terapeuta. **Terapeut** má za úlohu viesť terapiu, nastavovať parametre cvičení. Ďalej je schopný aj prispôsobiť pozíciu pacienta v prostredí. Táto funkcionalita je potrebná, keďže pacient má obmedzené možnosti používať kontroléry, a teda úplné ovládanie nie je možné ponechať len na pacienta.

4.1.1 Podporované platformy

Ďalej sme si určili, ktoré platformy máme za cieľ podporovať. Pri každej platforme uvedieme, aké role sú potrebné aby boli podporované. V našom prípade to sú nasledovné:

- VR helmy - oba roly pacient aj terapeut,
- PC platforma - len terapeut (pacient je podporovaný, ale nepočítame s jeho využitím).

VR helmy budú podporované vďaka použitiu štandardu OpenXR. Ten nám zabezpečí širokú kompatibilitu s veľkým množstvom headsetov [26]. OpenXR môžeme použiť pri samostatných VR helmách ako napríklad Oculus Quest 2, ako aj v spojení s helmami, ktoré fungujú pripojené k PC.

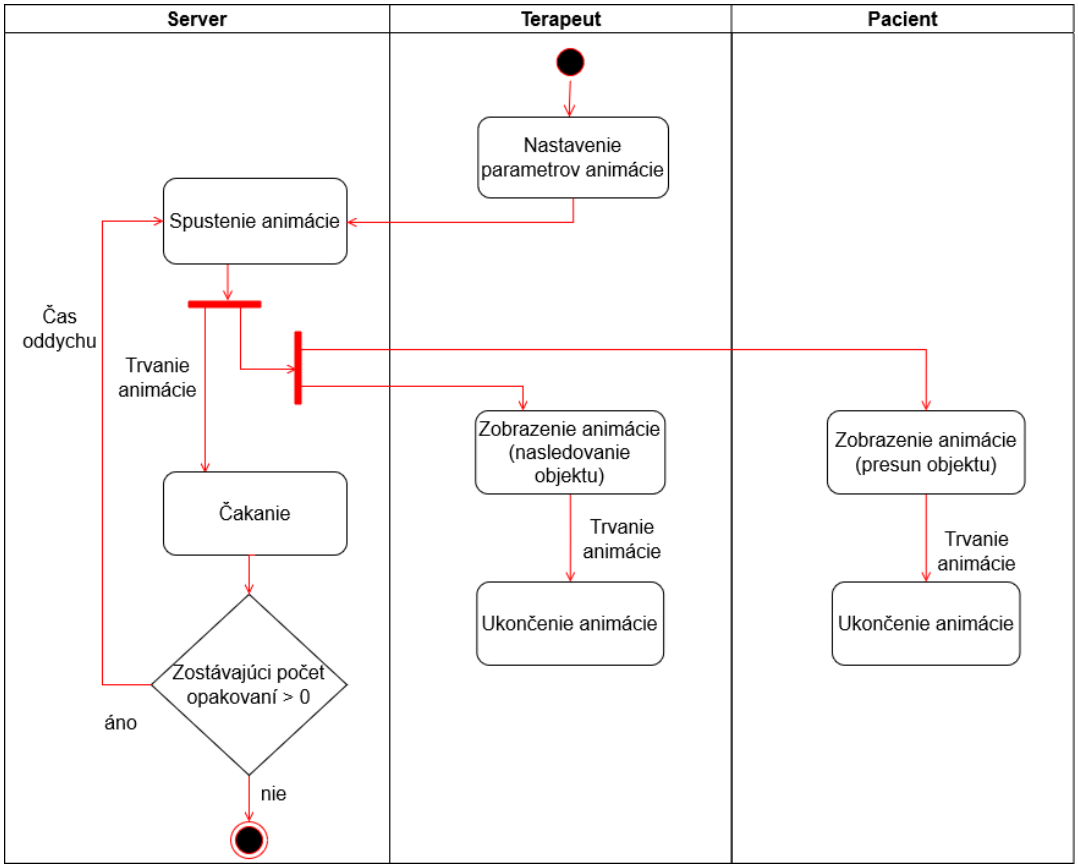
Podpora PC platformy bude zabezpečená formou desktopového klienta, ako aj možnosťou prepnutia do módu simulovanej VR helmy. Mód simulovanej VR helmy je potrebný z dôvodu testovania aplikácie. Desktopový mód je potrebné vytvoriť, keďže chceme poskytnúť funkcionalitu terapeuta aj v prípade, že nie je možné použitie VR helmy. V prípade simulácie VR helmy ale neplánujeme jeho využitie pri bežnom používaní aplikácie, keďže ovládanie je pomerne zložité. V tomto móde je potrebné samostatne ovládať avatara, pohyb a rotáciu hlavy, pohyb a rotáciu jednotlivých ovládačov.

4.1.2 Proces animovania počas rehabilitácie

Keďže sa nachádzame v kolaboratívnom prostredí, kde sa nachádza aj pacient aj terapeut, je potrebné si uvedomiť, že na to, aby sme boli schopní zobraziť všetky animácie, je potrebné meniť stavy týchto animácií a následne tento stav zdieľať. Takýto postup je potrebný, keďže nie je možné zdieľať úplne každý prvok avatarov. Takéto množstvo zdieľaných dát by zaťažilo sieť a zároveň aj samotný procesor zariadenia. Na diagrame 4.1 je vyobrazený proces spustenia a zobrazenia animácií. Na diagrame sme sa zamerali na ukázanie rozdielnych procesov vykonávaných na základe rolí používateľa. V rámci procesu animácie je tiež potrebné určiť jedného klienta, ktorý bude zároveň aj presúvať objekt (napríklad kocku), keďže tento je taktiež sieťovo zdieľaný objekt.

4.1.3 Proces rehabilitácie

Z pohľadu pacienta je proces rehabilitácie a cvičenia pomerne jednoduchý. Riadenie rehabilitácie má v réžii terapeut. Na diagrame 4.2 je vyobrazený postup



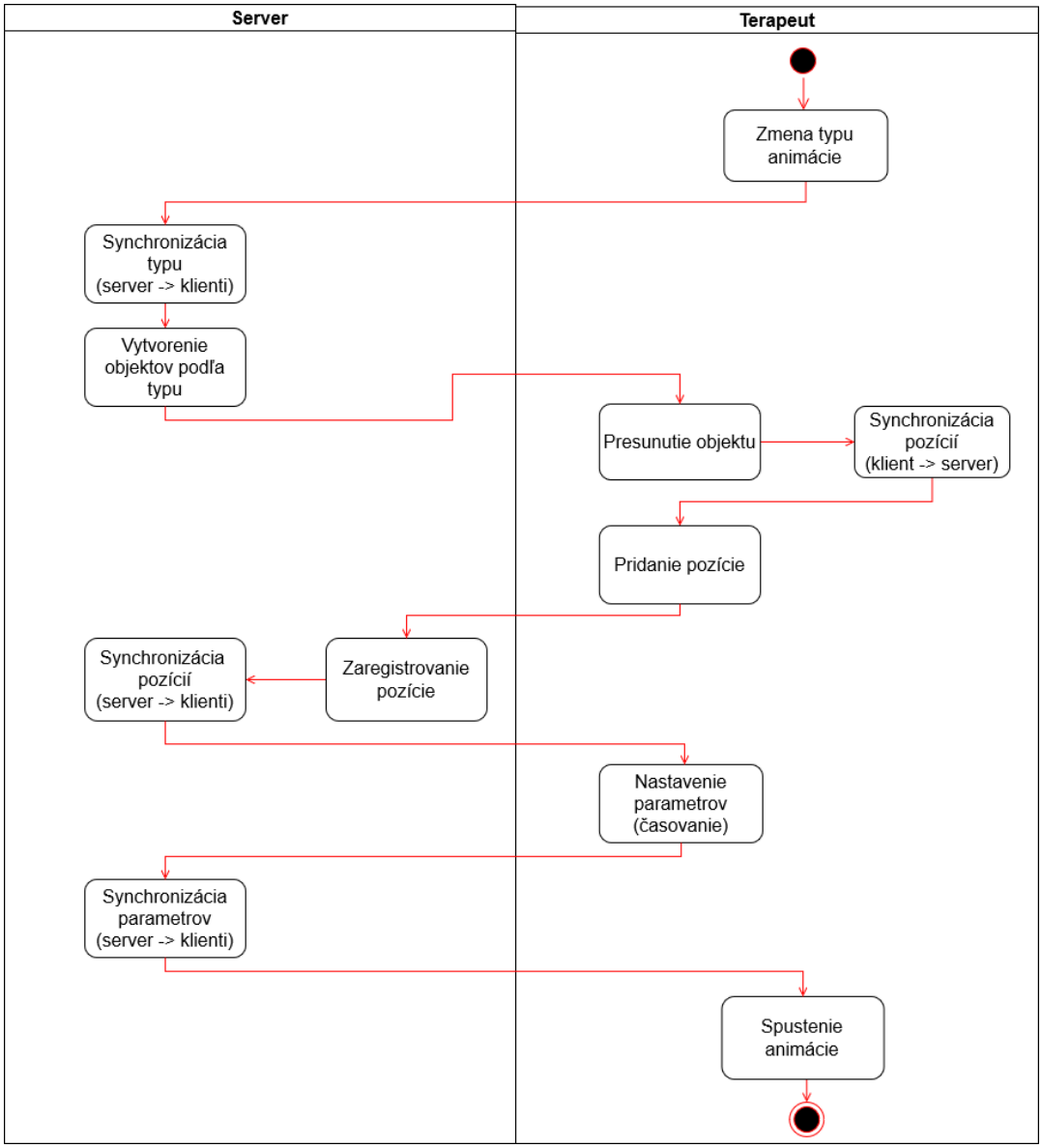
Obr. 4.1: Proces zobrazenia animácie v rámci rehabilitácie

komunikácie aj so smerom zdieľania informácií. Všetky parametre animácie ako typ animácie, počet pohybov a taktiež aj ich pozície a časovanie pohybov. Jedine pozícia objektu je zdieľaná zo smeru klienta. Všetky parametre sú zdieľané po každej ich zmene.

4.1.4 Zdieľané časti objektov používateľa

Nad touto funkcionalitou sme museli premýšľať už skôr pri práci na prototype (kapitola 3), no je užitočné ju spomenúť aj ako súčasť návrhu systému. V systémoch, ktoré poskytujú funkcionalitu zdieľaného online prostredia (ako napríklad hry pre viac hráčov) nie je možné zdieľať všetky informácie o pozíciách každého elementu avatara. Problémom je množstvo dát, ktoré by boli posielané po sieti. Všetky tieto dáta by musel spracovávať nielen server ale aj každý klient, čo by bolo z pohľadu sieťových nárokov a taktiež nárokov na hardvér náročné. Z tohto dôvodu je potrebné minimalizovať množstvo prvkov, ktorých pozície v priestore sú zdieľané. V našom prípade zdieľame teda len tieto objekty:

- samotný objekt používateľa,



Obr. 4.2: Proces nastavenia animácie z pohľadu terapeuta

- kamera,
- ovládače (len v prípade VR).

Všetky ostatné informácie ako napríklad animácie chôdze alebo otočenie ruky pri pohybe sú dopočítané samostatne na každej klientskej inštancii na základe synchronizovaných premenných alebo pozícií zdieľaných objektov.

4.1.5 Optimalizácia pri práci s VR

V spojení s VR je potrebné venovať značnú pozornosť optimalizácii aplikácie. Množstvo nastavení je možné zmeniť aj počas vývoja, no na niektoré je potrebné myslieť už v čase návrhu a implementovať ich od začiatku, aby neskôr nespôsobili

problémy, ktoré bude ťažké odstrániť. Je možné nájsť rôzne materiály v oficiálnych dokumentáciách, ktoré obsahujú rôzne odporúčania. Microsoft[1] vytvoril odporúčania pre prácu so zmiešanou realitou na zariadení HoloLens. Tieto odporúčania sú ale výrazne striktnejšie, než to čo musíme dodržiavať v našej implementácii. Spoločnosť Arm[2] taktiež vytvorila súhrn odporúčaní, ktoré môžu pomôcť pri vývoji. Tieto odporúčania nie sú záväzné, no je potrebné myslieť na nich už počas návrhu aplikácie a systémov. Dôležité je uvedomiť si obmedzenia, ktoré VR so sebou prináša.

4.2 Architektúra systému

Z funkčného hľadiska môžeme systém rozdeliť na niekoľko modulov, podľa funkcionality, ktorú jednotlivé komponenty a triedy zabezpečujú na:

1. **Animácie** - obsahuje komponenty používané pri animáciách ruky pacienta, pomocné triedy objektov používané na ukladanie informácií určených na animovanie rúk,

2. **Avatar** - obsahuje komponenty potrebné pre fungovanie avatarov, nastavenie parametrov avatarov a taktiež animovanie chôdze,

3. **Desktopový klient** - komponenty zabezpečujúce fungovanie kamery a pohybu desktopového klienta, interakcia s objektmi (ich presun v priestore) pomocou myši,

4. **Správa objektov** - tu sa nachádzajú komponenty použité na ovládanie objektov, komponenty riadenia nastavení aplikácie ako napríklad riadenie XR modulu, riadenie animácií, ďalej sa tu nachádzajú aj pod-moduly:

    - ovládanie objektu používateľa komponentom `CharacterManager.cs` (riadenie, vypínanie a zapínanie ostatných komponentov a objektov podľa rolí, nastavení a autority),
    - zoznam na správu objektov v scéne.

5. **Sieťová komunikácia** - ovládanie špecifických nastavení servera, mapovanie REST API funkcií,

[1] https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/performance-recommendations-for-unity?tabs=openxr

[2] https://developer.arm.com/documentation/102073/0100

6. **Používateľské rozhranie** - komponenty ovládajúce jednoduché elementy používateľského rozhrania, mapovanie udalostí menu na tlačidlá, špecifické skripty na riadenie menu objektov (napríklad menu na zápästí)

7. **Utility** - pomocné komponenty, ktoré používame napríklad na odlíšenie ovládačov,

8. **XR** - upravené implementácie niektorých XR komponentov.

XR modul neobsahuje všetku funkcionalitu potrebnú na riadenie XR komponentov, tá je zabezpečená balíkom XR Interaction Toolkit[3].

4.3 Podsystém používateľského rozhrania

Pre jednoduchšiu orientáciu a odstránenie potreby mať množstvo menu objektov v scéne na rôznych miestach, sme sa rozhodli implementovať systém menu pomocou kariet (anglicky tab menu). Implementácia sa nachádza v súbore `TabGroup.cs` a `TabGroupButton.cs`. Tento systém je možné vidieť na obr. 4.3 alebo aj na obr. 4.4. Je to systém, kde sa používateľ prepína cez jednotlivé karty a má tak prístup k rôznym funkciám a častiam nastavení. Hlavným cieľom je zlepšenie prehľadnosti a zoskupenie podobných funkcií na jednom mieste.

4.3.1 Úvodné menu

Po spustení aplikácie sa používateľ nachádza v scéne s menu, ako je vyobrazené na obr. 4.3. Toto menu je rozdelené na nasledovné karty:

- **Všeobecná časť**, kde sa môže používateľ pripojiť do online prostredia alebo vypnúť aplikáciu (komponent `GeneralMenuManager.cs`),

- **Ovládanie** - tu si môže používateľ pozrieť základné ovládanie, ako napríklad pohyb, interakcia s objektami, zobrazenie menu. Karta automaticky zmení obsah podľa toho, na akej platforme sa používateľ aktuálne nachádza, zobrazené ovládanie sa taktiež prispôsobí (napríklad pri používaní iného VR headsetu). Automatické vyplnenie obsahu je zabezpečené komponentom `InputBindingText.cs`,

- **Nastavenie avatara** - možnosť voľby vzhľadu avatara a taktiež resetovanie výšky používateľa (funkcie tlačidiel sú v súbore `AvatarMenuButton.cs`),

[3] https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.3/manual/

- **Nastavenia** - sa ďalej delia na všeobecné, audio a XR nastavenia (komponenty `SettingsMenuManager.cs` a `AudioMenuManager.cs`).

Obr. 4.3: Ukážka hlavného menu v offline scéne

4.3.2 Menu pre terapeuta

Menu pre terapeuta prešlo niekoľkými zmenami od prototypu. Na obr. 4.4 je možné vidieť, že okrem zmeny vzhľadu elementov menu k nim pribudli aj ďalšie možnosti na nastavovanie animácií. K tomuto menu sme pridali tiež dva ďalšie spôsoby, ako by s ním mohol terapeut interagovať, a to konkrétne:

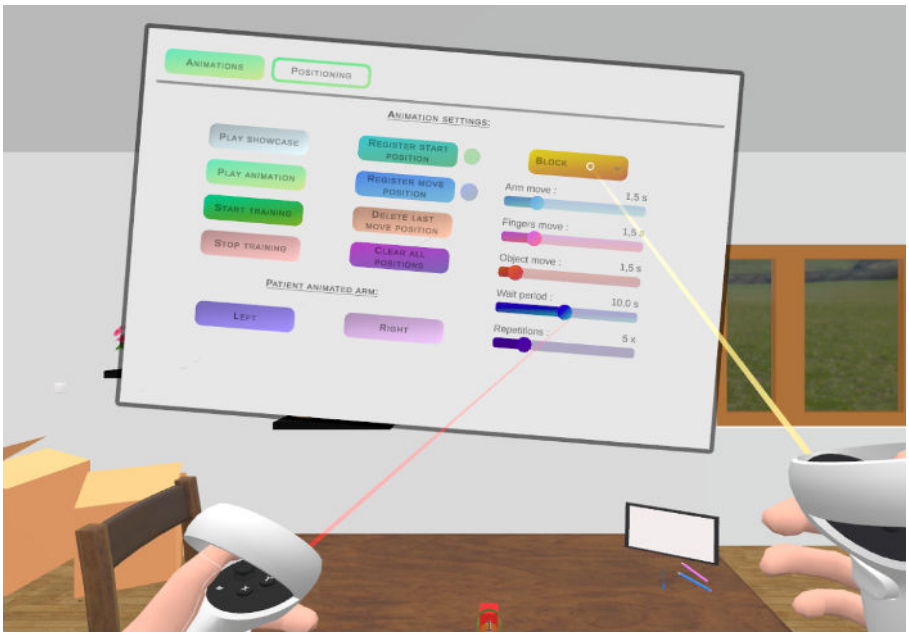
- menu nad zápästím (anglicky wrist menu),
- statické menu v 3D priestore.

Obidva tieto spôsoby interakcie sú dostupné ako v offline scéne tak aj v online scéne.

Wrist menu je možné vidieť na obr. 4.5. Táto funkcionalita je zabezpečená pomocou komponentu `MiniMenuManager.cs`, doplňujúca funkcionalita sa nachádza v súbore `MiniMenuVisibilityManager.cs`. Dôvodom prečo sme vytvárali túto funkcionalitu je to, že ak by potreboval terapeut nastavovať parametre počas práce s pacientom, bolo by preňho veľmi nepraktické, keby musel zakaždým odísť od pacienta a prísť k menu, ktoré je "nalepené" na stene. Keďže sme vo VR prostredí, nie je možné vykresliť menu elementy priamo na obrazovku ako pri bežných aplikáciách, ale musíme ich vykresľovať v 3D priestore. Takéto menu na zápästí nasleduje pohyb a rotáciu dlane, ako je to možné vidieť aj na obr. 4.5.

Statické menu je viditeľné na obr. 4.6. Toto menu si môže aktivovať podržaním tlačidla menu (v závislosti od platformy sa to môže líšiť, štandardne by to malo byť menu tlačidlo na ľavom ovládači). Menu sa zobrazí v smere kam sa používateľ práve pozerá a následne tam zostane "prilepené". Terapeut si teda môže celé menu zobraziť bližšie k sebe, no zároveň nie je spojené s jeho dlaňou a jej pohybom. Fungovanie menu využíva rovnaké komponenty ako v časti Wrist menu, je len doplnená komponentom `StaticMenuManager.cs`.

Obr. 4.4: Menu terapeuta určené na ovládanie animácií



Obr. 4.5: Ukážka menu na zápästí

Obr. 4.6: Ukážka statického menu v 3D prostredí

Nastavenie animácií

Funkcie používané ako obsluha udalostí tlačidiel sa nachádzajú v súboroch `TherapistMenuManager.cs` a `AnimationSettingsManager.cs`. V tejto karte (obr. 4.4) má terapeut možnosť meniť nastavenia animácií a ich spúšťanie. Medzi nastavenia, ktoré je možné meniť patrí:

- typ animácie (jednotlivé objekty majú vlastné menšie scenáre),
- pozície animácií
- trvanie animácie pohybu ruky,
- trvanie animácie pohybu prstov,
- trvanie animácie pohybu objektu (v prípade viacerých polôh, každý pohyb bude trvať daný čas),
- trvanie čakania na opakovanie pohybu,
- počet opakovaní.

Pozície pre animovanie je taktiež možné pridať pomocou tlačidiel v tejto karte. Nová pozícia je pridaná na koniec zoznamu už existujúcich pozícií podľa aktuálnej pozície cieľového objektu. Jediný rozdiel je v prípade animácie kľúča a zámku. V tomto prípade máme len dve pozície, štartovaciu a konečnú. Štartovaciu pozíciu je možné nastaviť pomocou presunutia kľúča ako v prípade ostatných typov

animácií. Konečná animácia sa nastavuje kliknutím na zámok. Následné spustenie animácií spôsobí presun objektu cez jednotlivé nastavené pozície.

Rovnako môžeme na tejto karte prepínať pacientovu ruku, ktorá je rehabilitovaná, prepínať ruku pacienta do polohy oddychu. Po prepnutí ruky do polohy oddychu sa ruka položí na stôl a taktiež sa vypne otáčanie avatara pacienta podľa rotácie hlavy. V tejto polohe akoby simulujeme pacienta sediaceho na stoličke. Rotáciu avatara vypíname preto, aby sa mohol pacient porozhliadať na svoje ruky počas ich animovania a taktiež aby sa pri otáčaní hlavy neotáčalo telo a ramená, čo by mohlo spôsobovať nepríjemný pocit a kazilo by to aj vzhľad animácií.

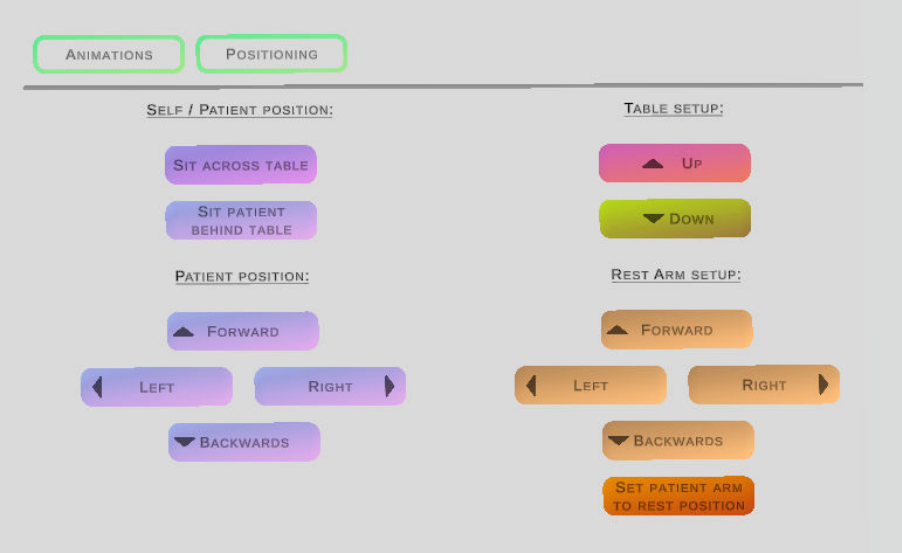
Menu obsahuje taktiež aj časovač v pravom dolnom rohu. Časovač sa spustí vždy pri čakaní na pohyb zo strany pacienta. Pod časovačom je aj ukazovateľ, ktorý postupne mizne podľa toho, koľko času ešte zostáva. Ukazovateľ zostávajúceho času je užitočný pre terapeuta, aby vedel či pacient potrebuje na pohyby viac, prípadne menej času. V ľavom hornom rohu časovača sa nachádza počet vykonaných pohybov. Pre pacienta sme vytvorili model malej obrazovky (pripomínajúci tablet), ktorý je položený na stole. Vďaka tomu aj pacient vidí koľko času mu ešte zostáva a koľko cvikov už vykonal. V ľavom dolnom rohu menu sa nachádza oblasť, kde sú zobrazované stavové správy. Správy používame napríklad pri komunikácii so serverom, pri ukončení cvičenia alebo ak nastane chyba s pohybom.

Prispôsobenie pozícií v menu

Pomerne dôležitá funkcionalita je aj nastavenie pozícií objektov. Funkcionalita tlačidiel tejto karty je v súbore `TherapistMenuManager.cs`. Medzi možné nastavenia patrí zmena výšky stola a teda zároveň aj ruky pacienta, doladenie pozície pacienta, zmena pozície pacienta a terapeuta za stôl. Na obr. 4.7 je možné vidieť toto menu aj s konkrétnymi tlačidlami. Počas diskusie pacientom sa zistilo, že sú náchylní na pozíciu ruky a zároveň teda aj výšku stola. Terapeut teda môže upravovať výšku stola hore alebo dole. Zároveň so zmenou výšky stola sa posunie aj ruka v polohe oddychu vyššie alebo nižšie. Tento posun je potrebné implementovať samostatne v kóde, keďže ruka nie je súčasťou stola, ale súčasťou avatara. Ďalej je možné aj prispôsobiť oddychovú polohu ruky na stole do strán.

4.4 Funkcie avatarov

Už skôr v kapitole 1.1 sme spomínali reprezentáciu používateľa v kolaboratívnom virtuálnom prostredí. Preto sme aj tomuto museli venovať dostatočne veľa



Obr. 4.7: Menu na prispôsobovanie pozícií

času a pozornosti. Hlavné ciele, ktoré sme sa snažili splniť pri implementácii tejto funkcionality sú:

- animácie avatara musia byť integrované s pohybom senzorov (hlava a ruky),
- avatar nesmie byť hardvérovo náročný na vykresľovanie,
- vzhľad avatara by mal byť príjemný.

Po zvážení všetkých týchto požiadaviek a taktiež vzhľadom na obtiažnosť tvorby avatarov, sme sa rozhodli pre použitie avatarov platformy z readyplayer.me[4]. Táto platforma ponúka možnosť vytvorenia si vlastných avatarov, buď výberom rôznych vzhľadov jednotlivých častí tela, alebo nahratím fotky. Pre lepšiu integráciu sme vytvorili celkovo 6 avatarov, z ktorých umožňujeme výber používateľom. Na obr. 4.8 je možné vidieť ukážku vygenerovaného avatara. Platforma ponúka taktiež aj SDK a integráciu s Unity, čo znamená, že je možné implementovať aj napríklad vlastný editor avatarov. Túto funkcionalitu v našej aplikácii ale nepotrebujeme, taktiež by to ani nebolo úplne možné, kvôli úpravám, ktoré potrebujeme na avataroch vykonať. S konkrétnym SDK v Unity bol len jeden problém, po importovaní avatarov najnovšou verziou SDK chýbali a nefungovali niektoré časti avatarov. Tento problém sme vyriešili prejdením na staršiu verziu 1.5.1. Možné je aj úplné odstránenie tohto SDK a stiahnutie avatarov ručne z webovej stránky platformy, keďže v aplikácii nevyužívame žiadnu funkcionalitu SDK. Aktuálne systém obsahuje troch mužských a troch ženských avatarov.

[4]https://readyplayer.me

Obr. 4.8: Ukážka avatara

Úprava modelu avatara

Jediný problém, ktorý sme si všimli so stiahnutými avatarmi je to, že rozdelenie jednotlivých častí avatarov bolo nevhodné pre naše použitie. Pôvodne má model avatara rozdelené časti ako napríklad "outfit_top" alebo "body". Nemal rozdelenie pre jednotlivé ruky, čo je ale pre nás dôležité, keďže chceme pracovať s jednotlivými rukami v prípade pacienta a jeho animáciami rúk. Z tohto dôvodu sme si modely avatarov upravili v aplikácii Blender[5]. Na obr. 4.9 je vyobrazený rozdiel štruktúr avatarov. Na obr. 4.9a) je štruktúra modelu pred úpravou, na obr. 4.9b) po úprave. Taktiež sme rovno aj premenovali niektoré iné časti, ktorých názvy boli automaticky vygenerované.

[5]https://www.blender.org/

(a) Časti modelu pred úpravou

(b) Časti modelu po úprave

Obr. 4.9: Štruktúra modelov avatarov

4.4.1 Animácie avatarov

Dôležitou časťou každého avatara sú tiež aj animácie. Medzi animácie avatarov patria hlavne:

- chôdza vpred, vzad,
- chôdza do strán.
- animácia nečinnosti,
- animácie rúk,
- animácia pohybu hlavy a otáčanie avatara.

Na animovanie týchto činností sme nepoužívali na všetko dynamické animácie. Hlavnými dôvodmi je fakt, že vytvorenie animácií chôdze pomocou dynamických animácií je pomerne zložité a taktiež príliš časté používanie dynamických animácií môže byť hardvérovo náročné.

Animácie chôdze

Ovládanie animácií avatara pri chôdzi a prepínanie animácií zabezpečuje komponent `AvatarWalkingController.cs`. Na animovanie nôh počas chôdze a na animáciu nečinnosti sme použili animácie z platformy mixamo[6]. Mixamo ponúka ob-

[6]https://www.mixamo.com/

rovské množstvo animácií, animácie je možné priamo ešte pred stiahnutím upraviť. Aj napriek tomu, že mixamo patrí do rodiny produktov Adobe je možné voľne a zadarmo používať ich animácie. Veľkou výhodou je možnosť nahratia avatara, keďže jednotlivé animácie sa týkajú rôznych častí kostry. Každý avatar môže mať nastavené kosti iným spôsobom, prípadne mať aj úplne iné počty kostí. Keďže sme použili avatarov z jednotného zdroja, postačí nám stiahnuť si animáciu len pre jedného z nich a následne ju môžeme použiť aj na iných avatarov. Dôvodom je to, že jediný rozdiel medzi nimi je vzhľad textúr, zatiaľčo kostry avatarov sa zhodujú. Existuje tu ale jedna výnimka a to v prípade či sa jedná o mužského alebo ženského avatara. Na prvý pohľad sú kostry týchto avatarov rovnaké, no v skutočnosti majú rôzne dĺžky a uhly. Z tohto dôvodu je potrebné stiahnuť si animácie zvlášť pre každú skupinu avatarov.

Animácie pre chôdzu do strán sme použili odlišné, zatiaľ čo animáciu pre chôdzu vpred a vzad používame len jednu. Chôdzu vpred a vzad odlišujeme tak, že spustíme animáciu so zápornou rýchlosťou, čo spôsobí, že sa animácia prehráva odzadu. To môžeme urobiť vďaka tomu, že tieto animácie z platformy mixamo je možné spúšťať v cykle. Animácie sú spúšťané na základe udalostí vyvolanými v Unity systéme. Vďaka tomu dokážeme jednotne vyvolať udalosti z rôznych miest v kóde, ak by to bolo potrebné. Pri zavolaní udalostí na spustenie animácií bolo potrebné ešte implementovať aj funkcionalitu synchronizácie informácií so serverom, ktorý následne informuje klientov. Takýmto spôsobom sa dozvedia, či majú danú animáciu spustiť v prípade konkrétneho avatara.

Okrem jednoduchých animácií chôdze sme implementovali aj schopnosť avatara prikrčiť sa do podrepu (súbor `AvatarLowerBodyAnimationController.cs`), aby bol pohyb prirodzenejší a aby nohy neprechádzali cez podlahu v našom virtuálnom prostredí. Súčasťou algoritmu ale je aj určitá voľnosť, kedy umožníme aby nohy prechádzali aj pod podlahu a to z dôvodu, že ak by sa pri používaní VR headsetu, chcel používateľ pozrieť na svojho avatara, pri sklonení hlavy by okamžite prešiel avatar do podrepu, čo ale nezodpovedá reálnemu pohybu.

Ako bolo v predošlej časti 4.4 spomenuté, museli sme avatarov upravovať v Blendri, čo ale spôsobilo, že sme už viac nemohli používať vzorový model pre animácie, ktorý je dodaný v rámci readyplayerme SDK. To je spôsobené rozdielnym spôsobom exportovania v prípade použitia softvéru Blender. To ale nie je až taký problém, keďže ako vzor je možné použiť ktorýkoľvek model z danej skupiny (jeden mužský a jeden ženský). Mixamo požaduje na použitie vzoru súbor s príponou FBX. Vzhľadom na rozdielne spôsoby exportu medzi typmi GLB a FBX sme museli používať aj v Unity naďalej už len typ súborov FBX.

```
if (isAnimatingLegs) {
    return;
}
Vector3 headPosition = headMove.action.ReadValue<Vector3>();
Vector3 positionDiff = headPosition - lastHeadPosition;

if (Mathf.Abs(positionDiff.x) < headMoveTreshold && Mathf.Abs(
     positionDiff.z) < headMoveTreshold) {
    return;
}
lastHeadPosition = headPosition;
lastHeadMovementTime = Time.time;
isAnimatingHead = true;
handleMovement(new Vector2(positionDiff.x, positionDiff.z));
```

Zdrojový kód 4.1: Ukážka animovania chôdze v prípade pohybu hlavy

Animácie chôdze pri pohybe hlavy

Virtuálna realita je špecifická tým, že pohyb je oproti bežným hrám možné vykonať nielen stlačením tlačidla (v prípade VR joystickom na ovládači), ale aj pohybom hlavy v reálnom svete. Unity XR interaction toolkit nám poskytuje rozhranie, ktoré môžeme využiť na snímanie pozície hlavy, vzhľadom k počiatku (jednoducho povedané lokálnu polohu hlavy). Zároveň je možné využiť akciu, ktorá sa vykoná pri zmene polohy hlavy na zaregistrovanie udalosti. Animácie pri pohybe hlavy sú riadené komponentom `AvatarWalkingController.cs`. V ukážke 4.1 je vyobrazený kód, ktorý používame na spustenie animácií pri pohybe hlavy. Oproti pohybu po stlačení tlačidla, je tu potrebné vykonať niekoľko obmedzení. V prípade, ak už je spustená animácia spôsobená tlačidlom, tá má prednosť pred pohybom hlavy. Následne sa skontroluje, či vykonaný pohyb presiahol určitú prahovú hodnotu. V našom prípade kontrolujeme súradnicu X aj Z samostatne. Až potom sa vypočíta, aká animácia sa má spustiť. Prahová hodnota pre animáciu je zavedená z toho dôvodu, že pri používaní VR helmy človek hýbe hlavou takmer neustále. Ak by sme zakaždým spúšťali animácie, bolo by to neprirodzené a zbytočné.

Nakoniec je ešte potrebné vyriešiť ukončenie animácie. Keďže udalosť zmeny pozície hlavy nám nedokáže povedať, kedy sa už pozícia hlavy meniť nebude. Spôsob ukončenia animácie je potrebné vykonať iným spôsobom, než pri ani-

máciách v časti Animácie chôdze. Pri týchto animáciách vieme použiť udalosť, kedy používateľ pustí joystick, prípadne kláves. Pri pohybe hlavy ukončujeme animáciu tak, že ju po určitom čase vypneme. V metóde Update kontrolujeme, či je spustená animácia hlavy a zároveň, či uplynul určený čas od spustenia tejto animácie. Ak áno, animáciu jednoducho vypneme. Trvanie animácie je potrebné nastaviť na hodnotu, ktorá nebude príliš vysoká, aby sa pri menších pohyboch neprehrávala animácia chôdze aj keď sa používateľ nehýbe. Zároveň ale nesmie byť animácia príliš krátka, pretože máme definovanú hranicu, kedy sa pohyb spúšťa. Ak by sa používateľ pohyboval pomaly, animácia sa by mohla javiť trhavo, kedy by sme ju neustále zapínali a vypínali.

Animácie pohybu v online prostredí

Ďalšou podstatnou časťou, na ktorú je potrebné myslieť, je to, že animácie spúšťame pri vstupe z ovládača, prípadne klávesnice alebo pri pohybe hlavy. Tieto akcie sú ale vyvolávané len lokálne, čo znamená, že animácie by v online priestore neboli prehrávané. Postavy by sa len kĺzali v priestore. Je nutné teda implementovať systém, vďaka ktorému informujeme ostatné inštancie, kedy je potrebné prehrávať konkrétnu animáciu. Zabezpečenie tejto funkcionality sa nachádza v súbore `NetworkAvatarWalkingController.cs`. Využívame pri tom synchronizované premenné, ktoré ponúka knižnica Mirror. K akciám, ktoré používame na animovanie zo sekcií Animácie chôdze a Animácie chôdze pri pohybe hlavy pridáme ešte naviac akciu, ktorá nastaví premenné a synchronizuje ich na serveri. Ten následne informuje ostatných klientov. Synchronizované premenné môžu mať aj definované funkcie, ktoré sa po zmene hodnoty vyvolajú. V tomto prípade tieto trigger funkcie využívame na spustenie a ukončenie animácie avatarov.

Knižnica Mirror síce ponúka vlastné riešenie, ktoré je určené na zdieľanie stavov pri animáciách, no tento komponent nebol s našim riešením kompatibilný. V prípade použitia ženského avatara kód vyhadzoval chybu, ktorú sa nám nepodarilo opraviť. Z toho dôvodu sme sa rozhodli implementovať vlastné riešenie. Zároveň sme tým získali väčšiu voľnosť pri rozširovaní systému do budúcnosti.

Animácie pohybu kontrolérov a otáčanie hlavy

Animácie pohybu rúk a otáčania hlavy avatara sme implementovali za pomoci dynamických animácii. Riadenie animácií sa nachádza v súbore `AvatarController.cs`. Pri rukách sme využili inverznú kinematiku. Princíp je založený na tom, že bod, vzhľadom ku ktorému je inverzná kinematika počítaná je neustále zarovnaný na miesto, kde sa práve nachádza konkrétny ovládač. Otáčanie hlavy

nasleduje otočenie kamery, zároveň trocha rotujeme aj chrbticu avatara, tá je ale otáčaná menej. Spoločne to tvorí efekt prirodzenejšieho otáčania. Okrem hlavy sa otáča aj celé telo avatara, no pomalšie. To je implementované takýmto spôsobom z dôvodu, že v prípade VR sa pri otočení používateľa (konkrétne headsetu) otočí aj celý avatar v našom prostredí. Túto funkcionalitu je možné samozrejme kedykoľvek vypnúť, napríklad v prípade pacienta, aby sa mohol porozhliadať na svoje ruky a aby sa mu pri tom neotáčal celý avatar.

Za účelom vytvorenia prirodzenejších animácií sme taktiež pridali komponent, ktorý zabezpečí otáčanie kostí v oblasti ramena a kľúčnej kosti. Táto kosť sa bude natáčať dopredu podľa toho, ako ďaleko pred ňou sa nachádza dlaň avatara. Otáčanie tejto kosti je teda obmedzené len na osy Y. Takéto natáčanie tvorí prirodzenejší dojem pri pohybe rúk dopredu a dozadu.

Na animovanie týchto pohybov sme využili balíček animation rigging [27]. Okrem komponentov na inverznú kinematiku obsahuje aj rôzne komponenty určené na otáčanie objektov, nasledovanie smeru objektu alebo tvorbu zložitejších animácií. Zároveň je možné animácie aj spolu "miešať", teda mať viac obmedzení, ktoré zároveň ovplyvňujú daný objekt. Komponenty obsahujú rôzne parametre ako mieru aplikovania animácie, obmedzenia na jednotlivé osy. Tieto parametre následne využívame na ovládanie, prípadne postupné animovanie pohybov.

4.4.2 Zmena avatara

Na zabezpečenie funkcionality zmeny avatara sme museli vytvoriť špecifickú implementáciu, keďže nie je len také ľahké meniť avatara za behu aplikácie. Implementácia sa nachádza v súboroch `AvatarModelManager.cs` a `AvatarSetup.cs`. Dôvodom je to, že avatar je súčasť objektu obsahujúceho XR komponenty a aj avatar potrebuje určité komponenty sám pre svoje fungovanie. Ak by sme nahradili celý objekt iným, nový objekt by taktiež potreboval všetky komponenty aj so správnymi hodnotami premenných. Takéto riešenie by si vyžadovalo duplikovanie všetkých komponentov do všetkých objektov avatarov, čo je ale výrazne neefektívne, pretože pri akejkoľvek zmene by bolo potrebné všetky zmeny vykonať ručne na všetkých objektoch (prípadne prefabrikátoch). Okrem nepraktického upravovania je taktiež aj pomerne náročné na výkon vytvárať inštancie celých objektov, obzvlášť tých, ktoré sú pomerne zložité a obsahujú množstvo komponentov. Z tohto dôvodu princíp, ktorý sme implementovali spočíva len v zmene meshov daných avatarov na iné. Zároveň pri tejto zmene je potrebné prejsť aj kostru daného meshu, pretože sú vždy naviazané aj na konkrétne kosti pomocou váh. Vďaka tomu, že všetky modely majú rovnaké kostry (v rámci kategórie - mužský

/ ženský model) je možné len podľa názvov nájsť kosti a tie nahradiť tými, ktoré už sú inicializované a nachádzajú sa v našom avatarovi. Čo sa týka materiálov, tak tie môžeme jednoducho skopírovať z prefabrikátu nového avatara a nahradiť aktuálne používané.

Tento princíp nám zabezpečí to, že sa nezmení celý objekt avatara, iba jeho vzhľad. Úplne ale problém s redundantnou prácou neobídeme, keďže je potrebné mať vytvorené a pripravené dva prefabrikáty - jeden pre mužských avatarov a jeden pre ženských.

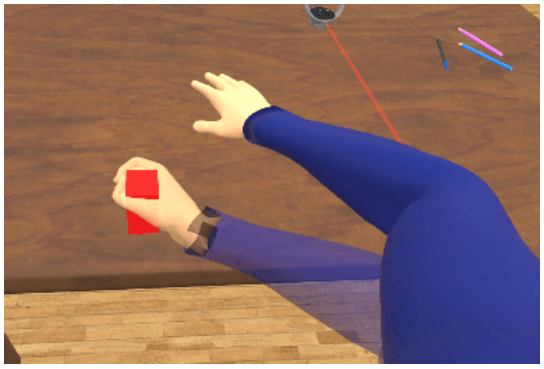
4.5 Dynamické animácie rúk pacienta

V aktuálnom systéme vytvorenom pomocouu A-Frameu je implementovaný vlastný podsystém, ktorý zabezpečuje dynamické animácie pomocou inverznej kinematiky. Problémom je pomerne vysoká zložitosť algoritmu.

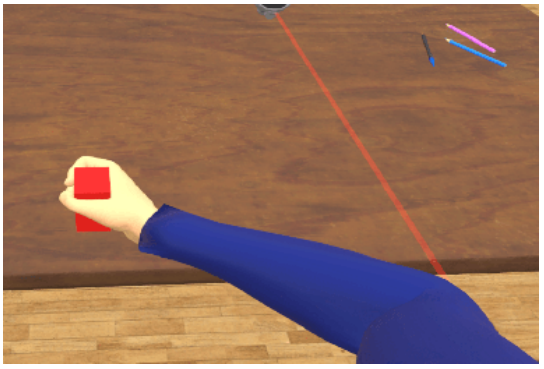
4.5.1 Počítanie inverznej kinematiky

Unity ponúka balíček animation rigging [27], ktorý obsahuje funkcionalitu rôznych animácií. Pre nás sú zaujímavé v tomto prípade inverzná kinematika dvoch kostí (anglicky Two Bone IK Constraint) a reťazová inverzná kinematika (anglicky Chain IK Constraint). Tie nám poskytujú možnosť použitia funkcionality inverznej kinematiky s tým, že tieto algoritmy sú dostatočne odladené a máme istotu, že budú fungovať správne. Princíp fungovania je založený na tom, že okrem častí kostry, ktoré budú súčasťou výpočtu sa používa aj ďalší objekt, tzv. cieľ (anglicky target). Tento objekt bude predstavovať cieľ kam sa konkrétny reťazec kostí bude snažiť dostať. Dalo by sa povedať, že to predstavuje objekt ktorého sa snažíme dotknúť. Inverzná kinematika pre dve kosti taktiež umožňuje nastaviť pomocný objekt (anglicky hint), ktorý slúži na určenie bodu, smerom ku ktorému sa budú kosti ohýbať. Týmto vieme obmedziť niektoré pohyby natoľko, aby vyzerali podobne ako je tomu v reálnom živote.

Ovládanie spúšťania animácií a komponentov inverznej kinematiky používaných pri animovaní je riadené komponentom `ArmAnimationController.cs`. Súčasťou implementácie sú aj animácie pre ľavú ruku pacienta. Pozície, ktoré používame pri animovaní ľavej ruky sú odzrkadlené pozície pravej ruky. Takýmto spôsobom nie je potrebné mapovanie pozícií nanovo.

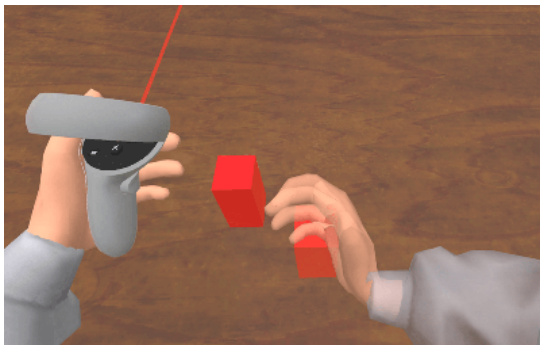


(a) Ukážková animácia

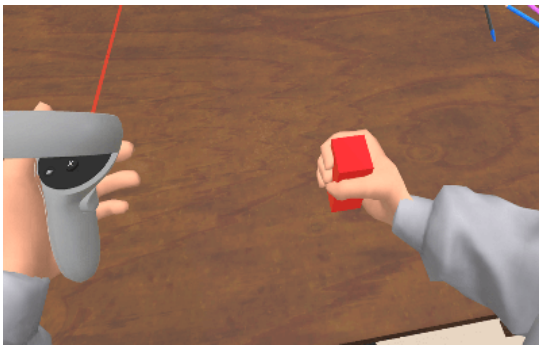


(b) "Pravá" animácia

Obr. 4.10: Ukážka použitia starého modelu ruky



(a) Ukážková animácia



(b) "Pravá" animácia

Obr. 4.11: Ukážka použitia modelu ruky, ktorý je súčasťou avatara

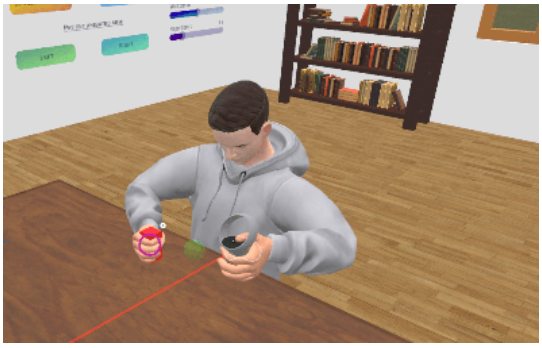
4.5.2 Ukážková a reálna animácia

Na obrázkoch 4.10 a 4.11 je možné vidieť rozdiel medzi ukážkovou a reálnou animáciou počas rehabilitácie (tréningu). Prvotne sme používali samostatný model ruky, ktorý je používaný aj v A-Frame verzii aplikácie (obr. 4.10). V aktuálnej implementácii, keďže používame celo-telového avatara, môžeme priamo animovať časť avatara, v našom prípade jeho ruky. Na obr. 4.11a je vyobrazená ukážková animácia z pohľadu pacienta, ukážka normálnej animácie použitím ruky avatara je na obr. 4.11b.

Animácie z pohľadu terapeuta je možné vidieť na obrázkoch 4.12. Ako už bolo spomínané v sekcii 4.1.2, terapeut len zobrazuje animáciu, jeho klientská inštancia objekt nepresúva. Pozíciu predmetu môže meniť len jedna klientska inštancia naraz, v našom prípade to je pacient. Na klientskej inštancii terapeuta, objekt pacienta len nasleduje pozíciu predmetu.

Na obr. 4.11a je animácia, ktorá slúži len na ukážku pohybu pre pacienta, vpravo je už reálna animácia, ktorá sa spustí po prijatí signálu z prostredia OpenVibe. Okrem toho, že sú animácie vizuálne odlišné, odlišné je aj ich spracovanie.

(a) Ukážková animácia



(b) "Pravá" animácia

Obr. 4.12: Ukážka animácií z pohľadu terapeuta

V prípade ukážkovej animácie, vytvoríme objekt, ktorý je lokálny pre každého používateľa, nie je zdieľaný. Ten potom následne každý klient vykresľuje a vyrátava sám, jeho animáciu aj pozíciu. Dôvodom prečo sme implementovali dva odlišné spôsoby práce s animáciami je zníženie posielaných dát medzi klientskymi inštanciami a serverom.

V prípade už posunu normálneho objektu, chceme aby bola jeho pozícia zdieľaná medzi všetkými klientmi, teda terapeutom a pacientom (prípadne ak by tam boli aj ďalší účastníci, tak aj medzi nimi). Postup je v takomto prípade nasledovný:

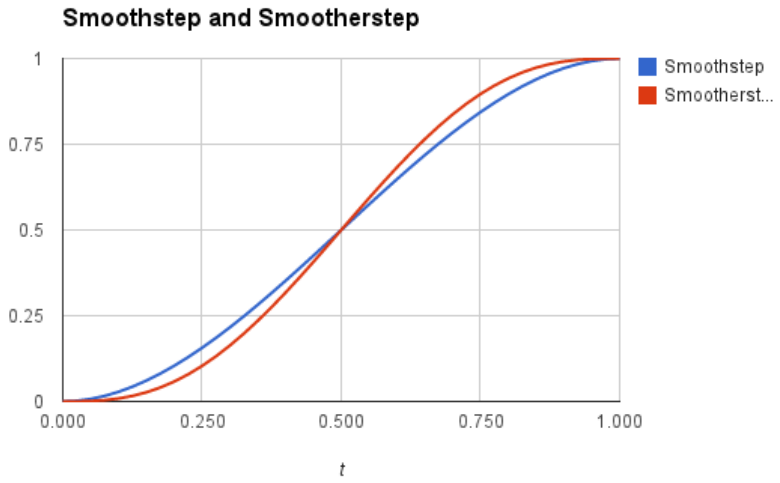
1. posunutie ruky a uchopenie na všetkých klientoch do pozície k objektu,
2. vyhodnotenie, či sa jedná o klienta pacient alebo nie
   - ak nie, spustí sa len rutina nasledovania pozície objektu,
   - ak áno, pacient najprv od servera získa vlastníctvo objektu a následne ho začne posúvať, zároveň aj nasleduje objekt.
3. spustí sa rutina ukončenia animácie.

Časť kódu používaného pri zmene pozície objektu a zároveň nasledovania tohto objektu je možné vidieť v ukážke 4.2. Najprv sa vypočíta veľkosť kroku, o koľko sa pohneme v rámci posunu. Aj napriek tomu, že používame algoritmus lineárnej interpolácie, je možné dosiahnuť zmenu hodnoty, ktorá sa bude meniť akoby po krivke [28]. Takéto hodnoty dokážeme dosiahnuť napríklad použitím výpočtu:

$$t = t * t * t * (t * (6f * t - 15f) + 10f)$$

Obrázok 4.13 ukazuje tvar tejto krivky červenou farbou. Na začiatku je animácia pomalšia, následne zrýchlime a na konci animácie (priblíženie sa k cieľovej hodnote) znova spomalíme. Použitím takéhoto výpočtu dosiahneme animácie, ktoré

vyzerajú viac prirodzene. Ak by sme použili len obyčajnú postupnú zmenu hodnoty, dostali by sme animácie, ktoré majú počas celého svojho trvania konštantnú rýchlosť. Tieto animácie by vyzerali neprirodzene roboticky.



Obr. 4.13: Priebeh zmeny hodnoty pomocou krivky [28]

Na vytváranie animácií a pohybov nepoužívame kód vo funkcii update, ale namiesto toho spúšťame takzvané korutiny (anglicky Coroutine). Korutiny sú metódy, ktoré je možné spustiť a bežia postupne na každom frame. Nie sú to vlákna a teda nevykonávajú funkcionalitu paralelne s hlavným vláknom. Korutiny je možné taktiež aj spomaliť, ak by sme napríklad volali množstvo korutín naraz na rôznych objektoch, mohlo by to byť náročné na výkon. V našom prípade to ale nie je potrebné, keďže máme zvyčajne spustenú len jednu korutiny naraz a taktiež plynulosť animácií je nutnosťou. Hlavným rozdielom oproti bežným funkciám je ten, že tie sa spustia a vykonajú takmer okamžite, v prípade korutín tomu ale tak nie je. Môžeme teda jednoducho oddeliť funkcionalitu a zároveň zrýchliť, zjednodušiť a sprehľadniť celý kód.

4.5.3 Pozície pre pohyby animácie

Terapeut môže taktiež zmeniť pozíciu presunutím objektu a stlačením príslušného tlačidla v menu, prípadne pridať ďalšiu pozíciu. Jediná výnimka je v prípade animácie kľúča, kde je vždy len jedna štartovacia a jedna cieľová pozícia. Cieľová pozícia sa vyberá kliknutím na príslušný zámok. Pridávanie je synchronizované na serveri, ktorý následne zdieľa informácie s klientskymi inštanciami. Riadenie pozícií je riadené komponentom `AnimationSettingsManager.cs`.

Keďže niektoré animácie podporujú viacero pozícií pre pohyb, bolo potrebné

```
float time = 0;
while (time < duration) {
    float t = time / duration;
    t = t * t * t * (t * (6f* t - 15f) + 10f);
    startTarget.transform.position = Vector3.Lerp(startMapping.position,
        endMapping.position, t);
    startTarget.transform.rotation = Quaternion.Lerp(Quaternion.Euler(
        startMapping.rotation), Quaternion.Euler(endMapping.rotation),t);
    if (alignTransforms) {
       targetsHelperObject.alignTargetTransforms();
    }
    time += Time.deltaTime;
    yield return null;
}
startTarget.transform.position = endMapping.position;
startTarget.transform.rotation = Quaternion.Euler(endMapping.rotation);
if (alignTransforms) {
    targetsHelperObject.alignTargetTransforms();
}
```

Zdrojový kód 4.2: Ukážka posunu objektu

Obr. 4.14: Ukazovatele pozícií

ich nejakým spôsobom aj vizualizovať kde presne sa nachádzajú. Na obr. 4.14 je možné vidieť ukazovatele vo podobe malých gulí. Prvá štartovacia pozícia je vždy zelená, ostatné pozície sú modré. Ukazovatele taktiež obsahujú komponent, ktorý zobrazí ich obrys v prípade, že sú zakryté iným objektom a používateľ by ich inak nevidel.

Zároveň sme pridali algoritmus, ktorý zistí, či sa aktuálne objekt nachádza nad stolom a zároveň, či je v dosahu ruky pacienta. Ak sú splnené oba podmienky, je zaregistrovaná nová pozícia pre presun objektu. Algoritmus sa nachádza v súbore `AnimationSettingsManager.cs`. Aby bolo jednoduchšie pre terapeuta určiť aký má každá ruka dosah, pridali sme ukazovateľ dosahu ruky. Na obr. 4.15 je vyobrazený ako takýto ukazovateľ vyzerá. Tento ukazovateľ sa zobrazuje len v prípade, že terapeut aktuálne presúva nejaký predmet. Zároveň je ukazovateľ viditeľný len v konkrétnej klientskej inštancii, teda ak by sa napríklad v scéne nachádzali viacerí terapeuti, len ten, ktorý presúva predmet by tento ukazovateľ videl. Pre zlepšenie viditeľnosti dosahu sme vytvorili špeciálny materiál pomocou shaderov v Unity, ktorý ukazuje svoju textúru len okolo prienikov s ostatnými objektami, ako napríklad stolom. V 3D priestore by len obyčajný materiál, ktorý by bol čiastočne priesvitný nedával dostatočný prehľad, kde presne na stole končí dosah ruky a kam ešte pacient dosiahne. Kód 4.3 obsahuje algoritmus použitý na zistenie, či je objekt v dosahu. Na zistenie, či sa objekt nachádza nad stolom použijeme raycast, ktorým skontrolujeme, ktoré objekty sa nachádzajú pod pozíciou predmetu, ak tam je aj stôl, skontrolujeme či je v dosahu ruky. Dosah ruky je počítaný z objektu ukazovateľa aktuálne animovanej ruky. Alternatívne by bolo

možné použiť dĺžky kostí na výpočet dosahu ruky. Pri tomto výpočte nesmieme zabudnúť prirátať k dĺžke ešte vzdialenosť približne do stredu dlane, keďže to je miesto, kde za bežných okolností človek chytá predmety do dlane.

Obr. 4.15: Ukazovateľ dosahu ruky

4.5.4 Rozdielne typy animácií

Pri vytváraní animácií sme si zobrali ako základ animácie z už existujúceho riešenia. Animácie sme ale trocha upravili a taktiež sme pridali aj ďalší typ animácie. Animácie sú rozdelené podľa objektu, s ktorým sa počas animácie pracuje na:

- **kváder** - animácia s kvádrom spočíva v presúvaní kvádra po určených pozíciách. Na obr. 4.16a je možné vidieť ako to vyzerá,
- **kocka** - táto animácia je podobná ako v prípade kvádra, kedy pacient presúva kocku do stanovených pozícií postupne. Jediným rozdielom je spôsob uchopenia objektu, čo je možné vidieť na obr. 4.16b,
- **pohár** - v tejto animácií okrem presúvania pohára po určených pozíciách pacient na každej pozícii pohár aj zodvihne do vzduchu (obr. 4.16c),
- **kľúč a zámok** - v tejto animácií sú pred pacientom položené tri zámky, terapeut môže zámky presúvať a kliknutím na jeden zo zámkov sa určí cieľová poloha. V tejto animácii je možné definovať len začiatočnú a konečnú pozíciu. Po spustení animácie pacient uchopí kľúč, presunie ho do zvolenej

```
GameObject tableObject = ObjectManager.Instance.getFirstObjectByName("
    Table");

if (tableObject == null) {
    return false;
}

RaycastHit[] hits = Physics.RaycastAll(targetPosRotMapping.position,
    Vector3.down, targetPosRotMapping.position.y);
bool isAboveTable = false;
foreach (RaycastHit hit in hits) {
    if (hit.collider.gameObject.Equals(tableObject)) {
        isAboveTable = true;
        break;
    }
}
if (!isAboveTable) {
    Debug.LogWarning("Target Object not above Table");
    return false;
}

if (CharacterManager.activePatientInstance != null) {
    if (!CharacterManager.activePatientInstance.
         activeArmAnimationController.isTargetInRange(targetPosRotMapping.
         position)) {
        Debug.LogWarning("Arm cannot reach object, too far away");
        return false;
    }
}
return true;
```

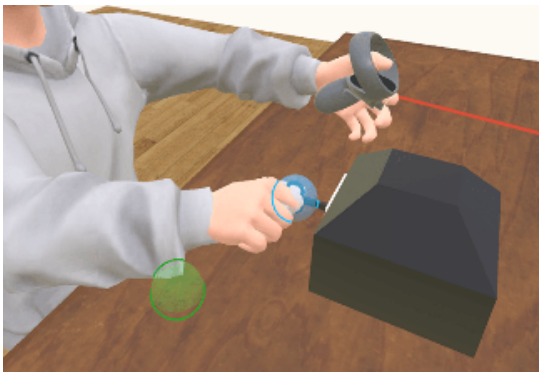
Zdrojový kód 4.3: Kontrola pozície objektu, či je v dosahu ruky a zároveň nad stolom

zámky a pretočí ním, následne kľúč znova vráti na stôl. Na obr. 4.16d je vyobrazená ukážka tejto animácie.

(a) Uchopenie a presun kvádra

(b) Uchopenie kocky

(c) Zdvihnutie pohára



(d) Presunutie kľúča do zámky

Obr. 4.16: Ukážka rôznych typov animácií

4.6 Prispôsobenie nastavení spojených s VR

Keďže je systém tvorený primárne pre VR, nebolo možné vyriešiť všetky problémy len simuláciou na PC. Testovanie prebiehalo kombinovane na headsete Oculus Quest 2 a zároveň simuláciou na počítači. Medzi hlavné problémy, ktoré vznikli počas testovanie na VR headsete patrili:

- nefunkčné trackovanie pozície a rotácie headsetu,
- nesprávne zdieľanie pozície postavy / kontrolérov,
- nízka kvalita obrazu.

Vo všeobecnosti sa pri práci s VR používa pojem XR rig pre avatara používateľa. XR rig zabezpečuje funkcionalitu ako trackovanie pozície a rotácie headsetu,

kontrolérov v prostredí, lokomóciu, mapovanie vstupov z kontrolérov, interakciu s UI, pozíciu hlavy avatara. Najčastejšie sa v dnešnej dobe používa trackovanie v prostredí oproti úrovni podlahy. Ak by sme ale použili také nastavenie pri simulovaní na PC, náš klient by mal hlavu priamo na zemi. Teda pri testovaní sme používali konfiguráciu parametrov, ktorá nám "natvrdo" nastavila pozíciu hlavy. Po spustení aplikácie na VR headsete to ale spôsobilo, že sme boli až príliš vysoko, keďže headset trackoval pozíciu hlavy a zároveň sme mali nastavený offset. Bolo teda potrebné algoritmom upraviť veľkosť offsetu v prípade použitia VR helmy. Aplikácia taktiež poskytuje možnosť prispôsobenia výšky avatara, zmenu avatara alebo aj zmenu zobrazeného ovládača. Ovládače taktiež obsahujú komponent, ktorý zobrazí ich obrys, ak sú zakryté. To je užitočné v prípade, ak by používateľ pohol rukami za nejaký objekt.

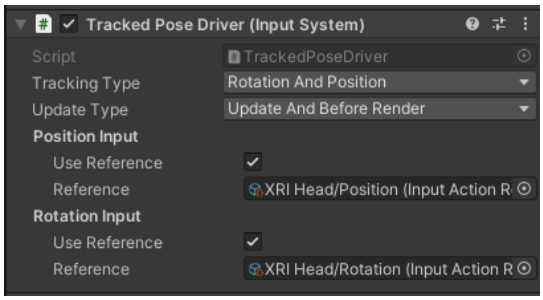
Snímanie polohy hlavy

Vďaka systému vstupu, ktorý Unity ponúka (nazývaný aj New Input System[7]) je možné použiť referencie (obr. 4.17a) na polohu a rotáciu hlavy na trackovanie VR headsetu. Iný spôsob predstavuje použitie akcií priamo, ktoré obsahujú tieto hodnoty. Na obr. 4.17 sú vyobrazené oba tieto spôsoby. Problém s použitím akcií priamo vzniká vtedy, keď chceme použiť viac než jeden zdroj týchto informácií. Príkladom tohto je použitie simulovaného VR headsetu. V takomto prípade by sme museli prepínať medzi týmito dvoma spôsobmi pomocou algoritmu podľa toho, či používame reálne VR zariadenie alebo len simulované. Ukážka použitia akcií priamo je na obr. 4.17b. Je potrebné si dať pozor, ktorý spôsob používame a prípade problému skontrolovať, či je všetko nastavené správne.

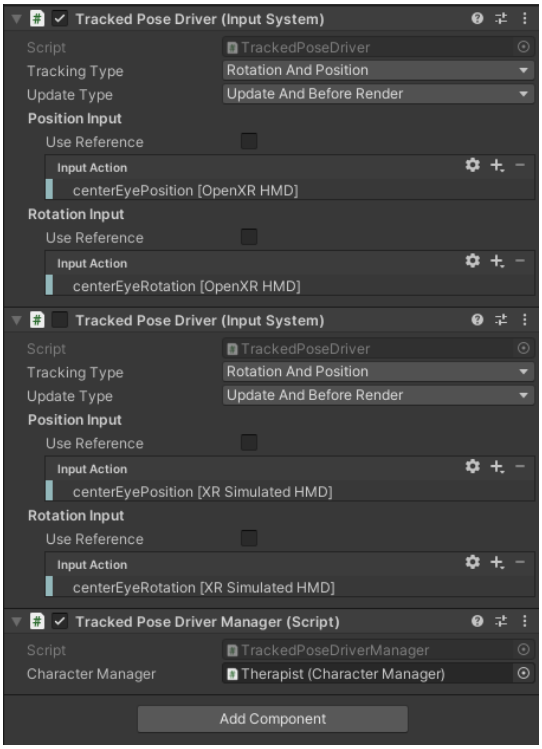
Optimalizácia grafických nastavení

Jedným z dôležitých nastavení je aj cieľová obnovovacia frekvencia. V prípade VR headsetov nenastavujeme cieľový počet snímkov. Počet snímkov je obmedzený obnovovacou frekvenciou daného VR headsetu[8]. Unity samotné teda obmedzuje obnovovaciu frekvenciu, akoby bola povolená technológia vertikálnej synchronizácie (VSYNC). Výnimkou je simulovanie VR headsetu. V tomto prípade sa obnovovacia frekvencia riadi nastavením akoby sme používali desktopového klienta, v našom prípade je počet snímkov za sekundu obmedzený obnovovacou frekvenciou monitora, na ktorom je aplikácia spustená.

[7] https://docs.unity3d.com/Packages/com.unity.inputsystem@1.5/manual/index.html
[8] https://docs.unity3d.com/Manual/VRFrameTiming.html



(a) Trackovanie použitím referencií



(b) Trackovanie použitím akcií priamo

Obr. 4.17: Komponenty trackovania headsetu

V rámci riešenia sme taktiež prešli na Universal render pipeline. Dôvodom boli hlavne možnosti, ktoré URP ponúka pre prácu so shadermi a možné prispôsobenie čo sa týka výkonu. URP má taktiež lepšie vyhliadky do budúcnosti čo sa týka dlhodobej podpory oproti vstavanému systému na vykresľovanie.

Dôležitá je aj miera toho, ako rušivé je používanie VR headsetu. Otestovanie tohto aspektu je ale zložité, keďže je výrazne subjektívny. Za účelom zlepšenia pocitu počas používania sme použili nasvietenie scény pomocou ambientného svetla, ktoré nie je príliš výrazné. Použitím odtieňov šedej sme znížili vysoký jas, ktorý môže pri používaní VR headsetov pôsobiť nepríjemne.

Nastavenie kvality vykresľovaných textúr je možné zmeniť parametrom **miera škálovania textúr**. Ideálna hodnota sa pohybuje medzi 1,0 až 1,6. Nesmieme zabudnúť dať si pozor na to, že príliš vysoké nastavenia grafickej kvality môžu ovplyvniť celkový výkon. Zmenu kvality textúr riadime podľa platformy, na ktorej sa nachádzame. Ak vykresľujeme na reálny VR headset, používame nižšiu mieru škálovania textúr. V prípade, že sa nachádzame na desktope a VR len simulujeme, môžeme si dovoliť zvýšiť škálovanie za účelom krajšieho obrazu.

Aliasing je jav, ktorému je potrebné taktiež venovať výraznú mieru pozornosti. Vzniká v prípade, keď nie je možné dostatočne vierohodne vytvoriť objekt. Spôsobuje tzv. "schodíkový efekt". Rozlišujeme aliasing geometrie a specu-

lar aliasing. Aliasing geometrie vzniká najčastejšie pri prudkom prechode medzi dvoma farbami môžeme čiastočne odstrániť použitím techniky Multisample anti-aliasing. Pre VR je hodnota MSAA 4x ideálna. Vyššia hodnota by mohla výrazne ovplyvňovať výkon. Specular aliasing je ťažšie odstrániť. Používanie hladších, oblých objektov a hrán, používanie matných materiálov môže výrazne pomôcť tento jav odstrániť.

Kompresiu textúr je potrebné vykonávať, keďže textúry s príliš veľkým rozlíšením by boli príliš náročné pri vykresľovaní vo VR. V Unity je možné nastaviť kompresiu globálne pre jednotlivé platformy, alebo ju meniť pri každej textúre samostatne. **Filtrovanie textúr** pomáha pri odstránení "štvorčekovaného efektu" pri aplikovaní textúr na väčšie objekty. Použitie bilineárneho, prípadne tri-lineárneho filtra je ideálne pre VR. Je možné ešte použiť aj anizotropný filter, ten môže ale výrazne zvyšovať náročnosť na výkon. Z tohto dôvodu je lepšie použiť nižšiu hodnotu úrovne tohto filtra.

Za účelom dosiahnutia rovnováhy medzi úrovňou detailov a potrebným výkonom je výhodné použitie **úrovní detailov modelu** (anglicky Level of detail, LOD). Vytvorením viacerých modelov rovnakého objektu a následným ovládaním zobrazovania úrovne detailov podľa vzdialenosti od kamery, dokážeme dosiahnuť optimalizáciu pri vykresľovaní veľkého množstva objektov. Niektoré objekty tieto úrovne detailov už majú vytvorené.

Užitočným nastavením je taktiež aj **farebný priestor**. Unity ponúka na výber z dvoch možností, a to gama a lineárny. Lineárny farebný priestor pomáha so znížením výskytu specular aliasing. Lineárny farebný priestor taktiež funguje lepšie pri použití URP, keďže sa používa fyzikálne založené vykresľovanie.

Pri práci s VR je tiež potrebné obmedziť množstvo **tieňov**, ktoré sú používané v scéne. Preto je najlepšie vybrať si z dvoch možností:

- nepoužívať tiene vôbec,
- použitie tzv. baked svetiel, ktoré sú prepočítané počas vývoja, teda neovplyvňujú výrazne výkon.

Problémom pri použití baked svetiel je ten, že je možné ich použiť len na statické objekty, ktoré sa nebudú v scéne hýbať. Vzhľadom na to, že v našom systéme je hlavný dôraz kladený práve na objekty, ktoré sú neustále v pohybe (objekty počas animovania, avatar pri pohybe) môže takéto riešenie pôsobiť rušivo. Existuje ešte jedna alternatíva, a to tzv. blob shadow. Tieto tiene predstavujú pomerne nenáročný spôsob ako vytvoriť jednoduché tiene objektov. V našom prípade nepoužívame tiene vôbec.

Všeobecná optimalizácia

Okrem konkrétnych nastavení hlavnú rolu pri optimalizácii zohráva efektivita kódu. Medzi všeobecné princípy, ktoré je užitočné dodržiavať patrí napríklad udržiavanie nízkej cyklomatickej zložitosti funkcií, nízkej miery vnorenia, či minimalizácia používania rekurzie. Z hľadiska čitateľnosti a udržateľnosti kódu je taktiež výhodné vyhýbať sa jednoriadkovým programom (anglicky one-liner code). Častým problémom pri nich býva nízka miera čitateľnosti kódu a často aj nízka efektivita z hľadiska výkonu.

Unity ponúka na nájdenie objektu v scéne a získanie jeho referencie funkciu `GameObject.Find()`. Problémom tejto funkcie je ale jej nízka efektivita. Táto metóda prejde všetkými objektmi, ktoré sa nachádzajú v aktuálnej scéne a porovnáva ich mená, či sa zhodujú so zadaným reťazcom. Je ale potrebné si uvedomiť, že v scéne sa často nachádzajú aj objekty, s ktorými nikdy nebudeme pracovať (objekty tvoriace prostredie). Za účelom zefektívnenia práce s objektmi sme implementovali komponent `ObjectManager.cs`, ktorý obsahuje zoznam zaregistrovaných objektov. Na uschovanie objektov využívame slovník (anglicky Dictionary), ktorý má časovú zložitosť prístupu k hodnotám podľa kľúča $O(1)$. V prípade, že by sme chceli pod jedným kľúčom uložiť viacero objektov, ukladáme ich do zoznamu. Prístup k týmto objektom v zozname je aj napriek tomu rýchlejší než použitie funkcie, ktorú ponúka Unity. Objekty sa komponentom `ObjectListMember.cs` do tohto zoznamu môžu samé zaregistrovať a zároveň sa pri zničení samé aj odstránia a teda zoznam obsahuje vždy len aktuálne existujúce objekty.

V niektorých prípadoch taktiež používame návrhový vzor singleton. Používame ho hlavne pri komponentoch, ktoré obsahujú všeobecné nastavenia, ako napríklad nastavenia prostredia, nastavenia XR, informácie o používateľovi ako napríklad jeho výška, jeho zvolená rola, zvolený vzhľad avatara. Tento návrhový vzor taktiež používame na uchovanie referencie na lokálnu inštanciu používateľa, cez ktorú môžeme následne komunikovať so serverom.

4.7 3D prostredie

Vzhľadom na to, že sa jedná o rehabilitáciu pacientov po traumatickom zážitku, je potrebné dbať aj na pocit z prostredia, kde sa nachádzajú. Cieľom je vytvorenie prostredia kde sa budú pacienti cítiť príjemne. Na obrázkoch 4.18 a 4.19 je možné vidieť niektoré časti nábytku a iných predmetov, ktoré boli pridané do prostredia za účelom zlepšenie pocitov pacienta počas rehabilitácie. Z predošlých testov ešte v systéme bežiacom na technológii A-Frame bolo zistené, že niektorí pacienti na

to reagujú veľmi citlivo. Naším cieľom je vytvorenie útulnej a príjemnej atmosféry, kde by sa mohol pacient jednoduchšie sústrediť. Existujúci systém využíva skybox pre vytvorenie interiéru prostredia. V našej implementácii používame taktiež skybox, ale rozdielom je to, že je použitý na vytvorenie okolia mimo miestnosti. Veľkým problémom pri použití skyboxu v spojení s VR je skutočnosť, že pri pohybe je skybox statický, teda človek nemá reálny pocit z toho, že by sa v priestore hýbal. Aj z tohto dôvodu sme skybox použili len na doplnenie prostredia.

Obr. 4.18: Pohľad na stôl

Obr. 4.19: Pohľad na miestnosť

Zároveň je potrebné myslieť aj na obmedzenia, ktoré s VR prichádzajú. Pra-

vidlá, podľa ktorých sme sa riadili sú nasledovné:

- používať objekty s rovnakým štýlom (ak používame objekty s menšou kvalitou, nemiešať ich s objektami s vysokým rozlíšením),
- nepoužívať veľa tieňov (alebo ich nepoužívať vôbec),
- nepridávať príliš veľa objektov, ktoré nemajú s prostredím nič spoločné.

4.8 Desktopový klient

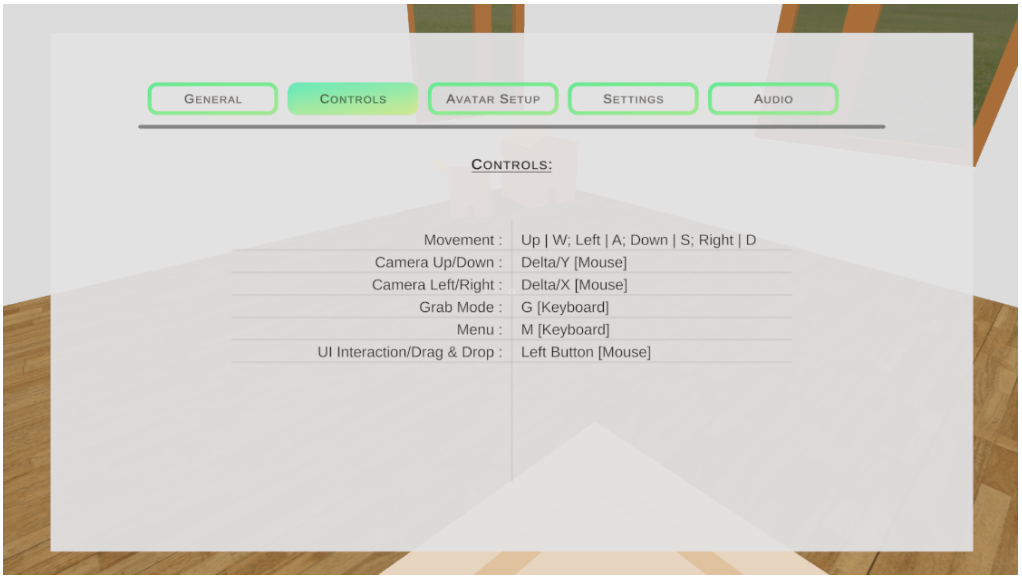
Ako bolo spomenuté v časti 4.1.1, jednou z požiadaviek je aj implementácia desktopového klienta. Táto funkcionalita je zabezpečená modulom "Desktopový klient". V tomto prípade nemáme na mysli simulovanie XR helmy, ale vytvorenie plnohodnotného klienta prispôsobeného na prácu na počítači. Na obmedzenie frekvencie vykresľovania používame vertikálnu synchronizáciu (skrátene VSYNC). Pre použitie tejto technológie sme sa rozhodli z dôvodu, že sa nejedná o hru a teda nie je potrebné dosahovať vysoké počty snímkov za sekundu. Taktiež toto nastavenie šetrí grafickú kartu počítača. Medzi hlavnú funkcionalitu, ktorú bolo potrebné implementovať patrí:

- prezencia v priestore rovnako ako pri používaní VR helmy,
- plnohodnotné ovládanie menu,
- možnosť presúvať objekty v priestore rovnako ako pri použití VR ovládačov.

Za účelom naplnenia všetkých bodov sme vytvorili ovládanie avatara z pohľadu prvej osoby (komponent `DesktopMovement.cs`). Ovládať pohyb je možné pomocou klávesov "WASD" alebo kurzorových šipiek, otáčanie funguje pomocou myši, interakcia s objektmi (UI aj 3D objekty) pomocou ľavého tlačidla myši. Otáčanie kamery funguje pohybom myši (`CameraMovement.cs`).

Menu je možné ovládať priamo stlačením tlačidiel v 3D priestore, alebo je možné zobraziť 2D menu na vrchu obrazu. Po stlačení skratky na zobrazenie menu (klávesa "M" alebo "Escape") sa odomkne myš a ovládanie menu je možné štandardným spôsobom. Na obr. 4.20 je možné vidieť vzhľad menu v tomto móde. Táto funkcionalita je implementovaná komponentmi `MouseClick.cs` a `MouseManager.cs`.

Presúvanie objektu je implementované jeho presúvaním myšou (komponent `DragAndDrop.cs`). Presúvať objekt je možné v bežnom móde pohybu, stlačením klávesy "G" je možné uzamknúť rotáciu kamery. Zároveň sa odomkne kurzor myši.



Obr. 4.20: Menu na desktopovom klientovi

4.9 Špecifické nastavenia servera

V našom projekte nie je server oddelený od klientskych inštancií. Spúšťanie servera zabezpečujeme prekladom, ktorý pri spúšťaní zabezpečí automatické pridanie prepínačov. Následne podľa týchto prepínačov vieme, či sa jedná o normálne klientske spustenie alebo nie. Jediný rozdiel počas behu serveru oproti klientom je ten, že má určité funkcie vypnuté. Medzi ne patrí zobrazenie obrazu a vypnutie modulu pre XR. Dôvodom je šetrenie prostriedkov servera a zrýchlenie jeho fungovania. Nastavenia pri preklade projektu je možné meniť priamo v Unity, v záložke Edit a následne Project Settings. V jednotlivých záložkách je možné meniť aj nastavenia pre konkrétne balíčky. Pre nás je zaujímavá hlavne záložka XRPlug-in management, kde je možné vypnúť XR modul priamo pre server build.

Server obsahuje aj vlastné implementácie niektorých algoritmov. Riadenie servera sa nachádza v komponente `CustomNetworkManager.cs`. Upravili sme spôsob pripájania používateľa do online prostredia. Pred pripojením používateľa server upraví určité hodnoty, ktoré sú potrebné pri nastavovaní objektov a avatarov používateľa.

4.9.1 Vytváranie objektov na serveri

Ako bolo spomenuté v časti 4.5.2, v prípade pohybu reálneho objektu používame zdieľaný objekt medzi všetkými klientmi. Funkcie používané pri vytváraní sa nachádzajú v komponente `AnimationSettingsManager.cs`. Vytváranie takýchto objektov nie je možné vykonať na klientovi. Ak by sme objekty vytvárali na klient-

```
NetworkIdentity[] ownedObjects = new NetworkIdentity[conn.owned.Count];
conn.owned.CopyTo(ownedObjects);
int avatarLayer = LayerMask.NameToLayer("Avatar");
foreach (NetworkIdentity networkIdentity in ownedObjects) {
    if (networkIdentity.gameObject.layer == avatarLayer) {
        continue;
    }
    networkIdentity.gameObject.GetComponent<NetworkTransform>().
         syncDirection = SyncDirection.ServerToClient;
    networkIdentity.RemoveClientAuthority();
    Debug.Log("Object with netID '" + networkIdentity.netId + "' released
          authority.");
}
base.OnServerDisconnect(conn);
```

Zdrojový kód 4.4: Odobratie autority nad objektami pri odpojení klienta

skych inštanciách, objekty by neboli zdieľané. Mirror poskytuje funkcionalitu na vytváranie týchto objektov správnym spôsobom, teda tak, že budú vytvorené na serveri a automaticky aj na klientoch so všetkými potrebnými nastaveniami. Pri prepínaní objektov je taktiež potrebné zničiť tie staré. Tu by sa mohlo zdať, že jednoduchšie by bolo nastaviť tieto objekty na neaktívne, čo je menej náročné na výkon. Problém tu ale vznikol v prípade, že by sa pripojil používateľ, v čase keď už nejaké takéto objekty existujú. V prípade neaktívnych objektov nedochádza k ich zdieľaniu a teda by ani neboli vytvorené na klientovi. To by mohlo spôsobiť nesprávnu synchronizáciu a teda celkovo by to zvýšilo zložitosť tejto časti algoritmu.

S vytváraním a predávaním vlastníctva je taktiež spojené aj odstránenie vlastníctva (autority). Predvolené nastavenia v knižnici Mirror spôsobia zničenie všetkých objektov, ktoré patria klientovi, ak sa klient odpojí. Z toho dôvodu sme museli túto funkcionalitu prerobiť tak, aby sme pri odpojení klienta skontrolovali všetky objekty, ktoré mu patria (má nad nimi autoritu) a odoberie sa klientovi vlastníctvo objektu. Jedinú výnimku predstavuje objekt avatara, keďže tento chceme aby bol s odpojením klienta odstránený. Ukážka 4.4 obsahuje kód metódy, ktorá sa zavolá tesne pred odpojením klienta.

4.9.2 Komunikácia so serverom

Komunikáciu so serverom rozdeľujeme na internú a externú. V kapitole 3 sme ukázali spôsob komunikácie použitím REST API (obr. 3.1).

Interná komunikácia

Pod internou komunikáciou rozumieme komunikáciu medzi klientmi a serverom. Komunikácia prebieha cez metódy, ktoré sú označené ako [*Command*], prípadne cez synchronizované premenné. Hlavný komponent, ktorý používame na komunikáciu so serverom je `NetworkCharacterManager.cs`. Komponent obsahuje statický odkaz na lokálnu inštanciu používateľa, všetky objekty k nemu môžu pristupovať a volať metódy cez referenciu. Metódy označené ako [*Command*] sa spúšťajú len na serveri. Tieto metódy môžu byť za bežných podmienok zavolané len vtedy, ak ich zavolá objekt s autoritou lokálneho klienta. To sa zvyčajne rieši tak, že lokálny klient si po pripojení nastaví hodnotu statickej inštancie na svoj objekt. Následne sa objekty odkazujú na túto inštanciu a cez ňu volajú metódy na serveri. V niektorých prípadoch je ale výhodnejšie umožniť zavolanie týchto metód ak objekt nie je objekt lokálneho klienta. Príkladom sú komponenty, ktoré obsahujú zdieľané premenné. Na to aby sa aktualizovala hodnota na všetkých klientoch je potrebné najprv aktualizovať hodnotu na serveri, ten následne aktualizuje hodnoty na všetkých klientoch.

Komunikácia zvonku

Komunikácia je zabezpečená použitím HTTP REST API. Mapovacie funkcie sa nachádzajú v súbore `RestRequestHandler.cs`.Na vytvorenie REST API používame plugin Simple HTTP and REST Server[9], ktorý je voľne dostupný. API slúži na spustenie animácií. Na diagrame 4.1 je vyznačená akcia "Spustenie animácie" na serveri. Táto akcia sa ale spustí iba ako reakcia na prijatie požiadavky na REST API. Rovnako to funguje aj v existujúcom riešení. Okrem pohybov počas rehabilitácie je možné spustiť aj samostatnú animáciu, prípadne zmeniť typ aktuálnej animácie. Spôsob, akým je spúšťanie animácií implementované sa neviaže na konkrétny protokol. To znamená, že je možné jednoducho doplniť iný spôsob komunikácie, napríklad pomocou socketov.

[9]https://assetstore.unity.com/packages/tools/utilities/simple-http-and-rest-server-244127

5 Vyhodnotenie implementácie

Počas vývoja sme komunikovali a spolupracovali s tímom Ing. Mgr. Romana Rosipala, DrSc. z Ústavu merania SAV. S ich pomocou sme identifikovali potrebnú funkcionalitu, ktorá je prioritná a zároveň novú funkcionalitu, ktorú je potrebné vytvoriť. Okrem postupného testovania funkcionality je potrebné aj neustále komplexné testovanie systému. Vzhľadom na to, že táto práca je pokračovaním existujúceho projektu a jej hlavným cieľom bolo prepracovanie aplikácie na novej platforme, neboli potrebné neustále experimenty za pomoci pacientov. Vďaka tomu sme sa mohli opierať už o existujúce poznatky.

Fungovanie systému bolo potrebné pravidelne testovať aj použitím VR headsetu. Na testovanie bol použitý VR headset Oculus Quest 2. Headset sme počas testovania používali dvomi spôsobmi. Pripojený k počítaču a samostatne fungujúci. Takýmto spôsobom sme dokázali otestovať aj funkcionalitu pre headsety, ktoré potrebujú byť pripojené k počítaču počas svojho fungovania. Počas testovania sme taktiež využívali aj počítač, kde sme mali spustené viaceré inštancie. Takýmto spôsobom sme boli schopní otestovať funkcionalitu pre rôzne roly, ako aj inštanciu servera.

V rámci projektu sme sa zúčastnili aj konferencie 2022 IEEE 16th International Scientific Conference on Informatics[1], ktorá sa uskutočnila v novembri 2022. Na konferencii boli prezentované témy, ktoré sa zaoberajú problematikou interakcie počítača a ľudského mozgu. Konferencie sa zúčastnili aj pracovníci z tímu Ing. Mgr. Romana Rosipala, DrSc., s ktorými spolupracujeme na projekte. Počas konferencie mali možnosť otestovať aplikáciu. S ich pomocou a skúsenosťami sme boli schopní analyzovať nedostatky a taktiež určiť ďalšie smerovanie aplikácie. Medzi hlavné aspekty, na ktoré sme sa zamerali počas tohto testovania bolo použitie prostredia z pohľadu terapeuta, funkcionalita kolaborácie, funkčnosť zdieľaného virtuálneho prostredia.

Jedným z testov, na ktoré sme sa zamerali je aj schopnosť plynulosť aplikácie počas jej používania. Počas testovania sme mali vždy spustené tri inštancie -

[1] https://informatics.kpi.fei.tuke.sk/

server, dva klientske inštancie. Testovanie sme vykonali rôznom hardvéri. Hardvérové špecifikácie zariadení je možné vidieť v tabuľke 5.1. Počas testovania sme

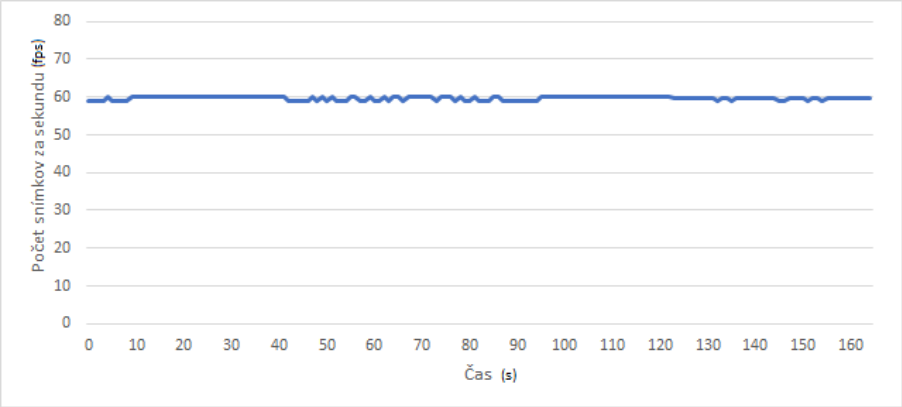
Platforma	*CPU*	*GPU*	*RAM*	*Obnovovacia frekvencie displeja*
Oculus Quest 2	Qualcomm Snapdragon XR2	Qualcomm Snapdragon XR2	6GB	90Hz (72Hz)
Stolový počítač	Ryzen 5 3600	Nvidia GTX1660 (6GB VRAM)	16GB	59.9Hz
Laptop	Intel Core i5-8250	GTX 1050 (4GB VRAM)	8GB	60Hz
Laptop (eko režim)	Intel Core i5-8250	Intel UHD 620 (128MB VRAM)	8GB	60Hz

Tabuľka 5.1: Hardvérové špecifikácie zariadení použitých pri testovaní

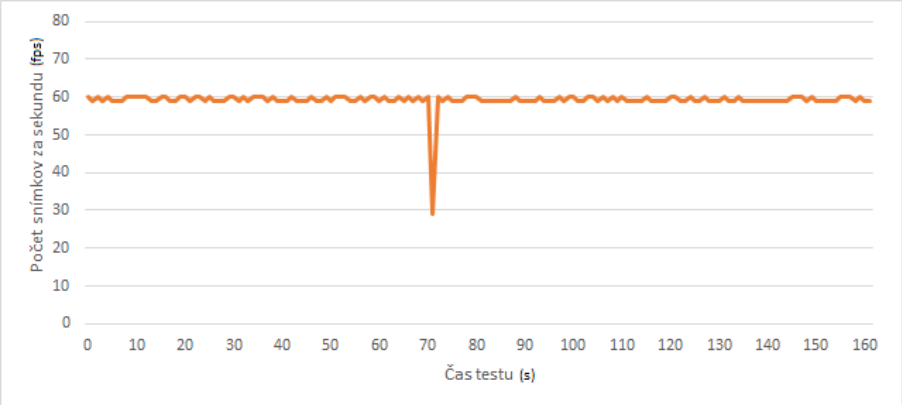
vykonali rôzne nastavenia, pohyb po scéne, presun objektov, pridávanie pozícií a animovanie rúk pacienta (proces rehabilitácie). Na vyhodnotenie výsledkov testov sme použili hodnotu počtu snímkov za sekundu. Testovanie na stolovom počítači a na laptope sme vykonali v dvoch režimoch: desktopový klient, simulovaný VR headset.

Na obrázkoch 5.1 a 5.2 je možné vidieť vizualizáciu priebehu testu na stolovom počítači. Test prebiehal podľa očakávaní bez problémov, keďže hardvér stolového počítača je pomerne výkonný.

Testovanie na laptope bolo ešte rozdelené do dvoch režimov. Režim šetrenia energie sme použili z dôvodu, že v tomto režime laptop nepoužíva dedikovanú grafickú kartu, ale integrovanú. Na obrázkoch 5.3 a 5.4 je možné vidieť priebehy počtu snímkov za sekundu počas testov s dedikovanou grafickou kartou. Z obr. 5.4 je možné vidieť, že narozdiel od testu na stolovom počítači (obr.5.2), bol počet snímkov za sekundu pri použití simulovaného VR headsetu prakticky neobmedzený, aj napriek tomu, že podľa nastavení aplikácie by mal byť obmedzený obnovovacou frekvenciou monitora. Počas testovania aplikácia nehlásila žiadnu



Obr. 5.1: Vizualizácia počtu snímkov - desktop

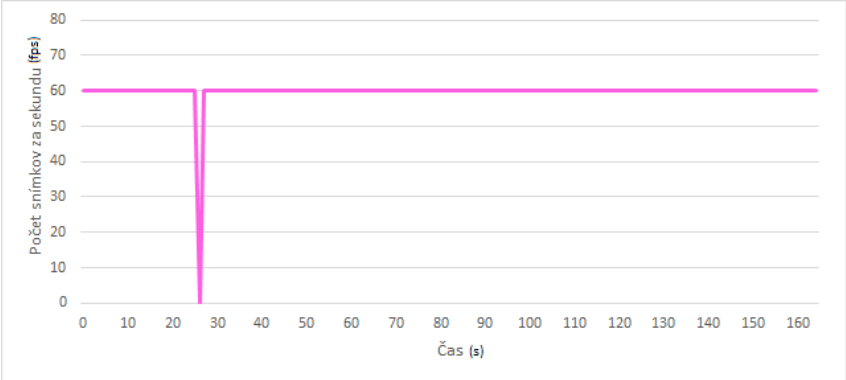


Obr. 5.2: Vizualizácia počtu snímkov - desktop + simulovaný VR headset

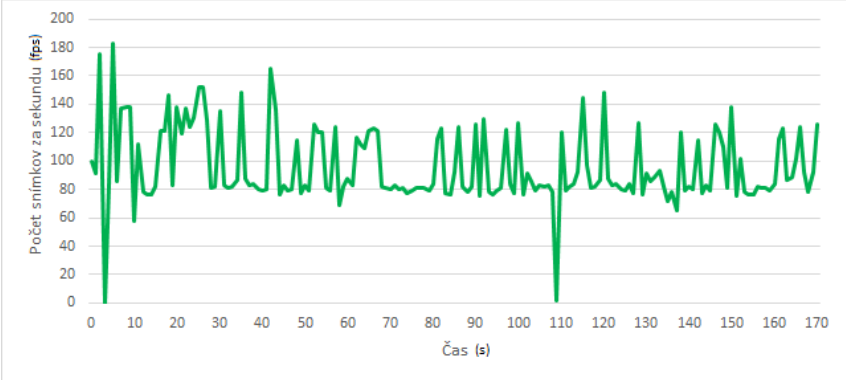
chybu. Domnievame sa, že tento výsledok je následkom rozdielnych nastavení zariadení.

Testovaním aplikácie v eko režime sme testovali výkon aplikácie na slabšom hardvéri. Na obr. 5.5 je priebeh počas testovania aplikácie v desktopovom režime. Počet snímkov za sekundu pomerne výrazne preskakuje medzi dvoma hodnotami - 60 a 30. Dôvodom týchto zmien môže byť aj skutočnosť, že daný laptop je schopný meniť obnovovaciu frekvenciu monitora pri väčšej záťaži v eko režime. Aplikácia bola aj napriek tomu plne použiteľná. Na obr. 5.6 je priebeh testu na laptope so zapnutým eko režimom so simulovaným VR headsetom. Počet snímkov za sekundu bol výrazne nízky. Je potrebné ale podotknúť, že v tomto režime grafická karta vykresľuje obraz dva-krát, pre každé oko zvlášť. Vzhľadom na nízku veľkosť VRAM integrovanej grafickej karty a jej nízky výkon boli hodnoty očakávané.

Aj keď má zariadenie Oculus Quest 2 obnovovaciu frekvenciu 90Hz na každý displej, plugin OpenXR je schopný využiť len 72Hz mód. Toto obmedzenie vychádza z obmedzenia pluginu, ktorý nemá implementovanú funkcionalitu použitia



Obr. 5.3: Vizualizácia počtu snímkov - laptop



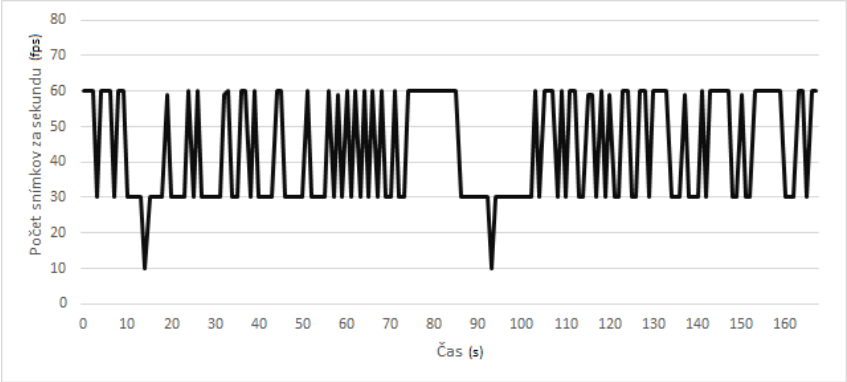
Obr. 5.4: Vizualizácia počtu snímkov - laptop + simulovaný VR headset

inej obnovovacej frekvencie[2]. V budúcnosti by táto funkcionalita mala byť doplnená. Výsledky testovania na VR headsete Oculus Quest 2 je možné vidieť na obr.5.7. Počas testovania sme si všimli mierne poklesy počtu snímkov. Po zmene niektorých materiálov, meshov a nastavení pre preklad sme test zopakovali. Z priebehu na obr.5.8 je možné vidieť, že stabilita sa výrazne zlepšila.

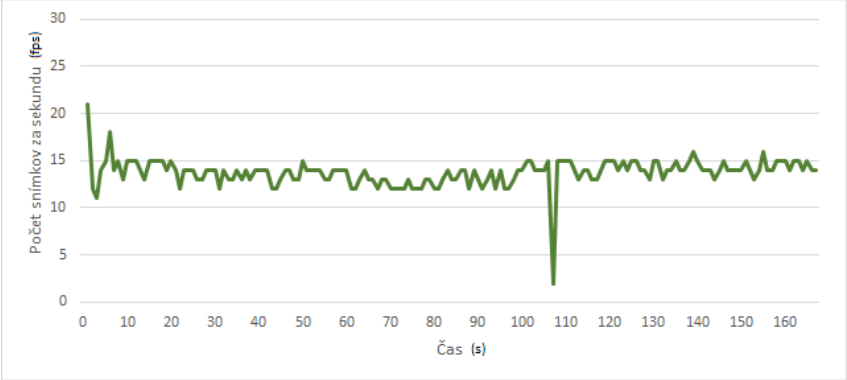
Testom sme zároveň potvrdili schopnosť samostatného VR headsetu zvládať prepočítavanie inverznej kinematiky aj v prípade, že sa v scéne nachádza viacero používateľov naraz. Testovanie plynulosti aplikácie je obzvlášť dôležité, keďže pri VR je to jeden z faktorov, na ktorý používatelia výrazne reagujú. Hodnoty minimálneho, maximálneho a priemerného počtu snímkov za sekundu pre jednotlivé platformy sú vyobrazené v tabuľke 5.2.

Dôležitým bolo otestovanie aj celkovej infraštruktúry v prípade, že je server nasadený v reálnej prevádzke. Na nasadenie servera sme použili cloudovú službu AWS EC2. Testom sme zisťovali plynulosť aplikácie v prípade, keď je potrebné komunikovať so serverom na veľké vzdialenosti. Keďže sa nachádzame vo VR prostredí, ktoré je obzvlášť náchylné na rýchlosť odozvy a komunikáciu v reálnom

[2]https://forum.unity.com/threads/how-to-set-refresh-rate-with-openxr.1159352/

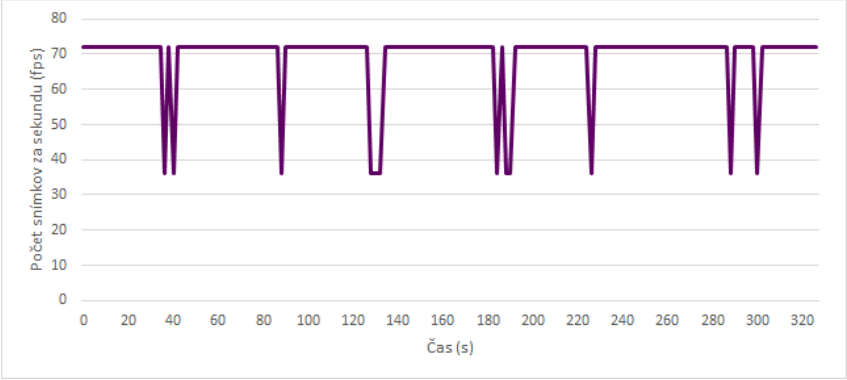


Obr. 5.5: Vizualizácia počtu snímkov - laptop (eko režim)

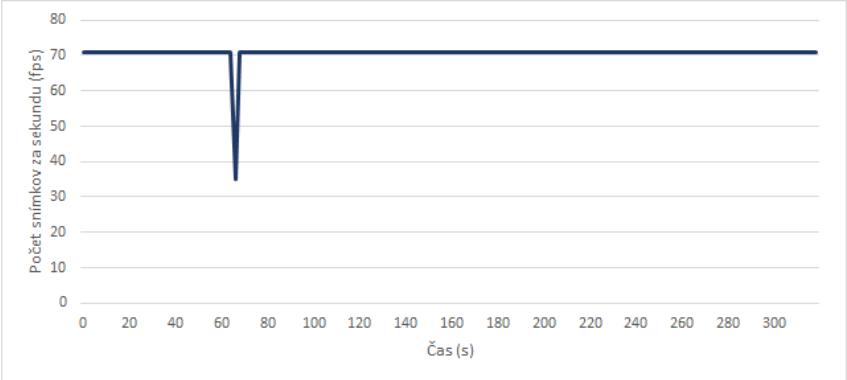


Obr. 5.6: Vizualizácia počtu snímkov - laptop (eko režim) + simulovaný VR headset

čase, bolo potrebné otestovať či množstvo posielaných dát nie je príliš vysoké. Počas testovania sme neprekročili hodnotu približne 10kB/s pre odosielané dáta na klientovi. Posielané dáta tvoria primárne informácie o zdieľanej polohe avatara používateľa. Čo sa týka prijímaných dát, tie sú rovné množstvu odosielaných dát na druhej strane (teda na ostatných klientoch). Pri použití desktopového klienta množstvo odosielaných dát výrazne klesá, keďže nie sú zdieľané dáta ako napríklad pozície ovládačov. Množstvo posielaných dát je ale väčšinou nižšie, než namerané maximum, keďže parametre objektov sú zdieľané len po zmene.



Obr. 5.7: Vizualizácia počtu snímkov - Oculus Quest 2



Obr. 5.8: Vizualizácia počtu snímkov - Oculus Quest 2 opakovaný test

Platforma	*Minimálny počet snímkov*	*Maximálny počet snímkov*	*Priemerný počet snímkov*
Oculus Quest 2 - po optimalizácii	35	71	70,7764
Stolový počítač	59	60	59,7202
Stolový počítač - simulovaná VR	29	60	59,2317
Laptop	0	60	59,6428
Laptop - simulovaná VR	0	183	95,7076
Laptop (eko)	10	60	43,5789
Laptop (eko) - simulovaná VR	2	21	13,8652

Tabuľka 5.2: Namerané hodnoty počtu snímkov za sekundu

6 Záver

Hlavné ciele práce boli návrh a implementácia systému vytvoreného použitím herného rámca Unity, určeného na neurorehabilitáciu horných končatín pacientov vo virtuálnej realite. Takýto systém môže pomôcť pacientom počas rehabilitácie, keďže sa ukázalo, že jedným z faktorov ovplyvňujúcich výsledky rehabilitácie je aj motivácia pacienta vykonávať cvičenie. Práve systém využívajúci virtuálnu realitu tomu môže pomôcť. Za účelom splnenia týchto cieľov sme v úvode analyzovali možnosti ako aj obmedzenia, ktoré so sebou virtuálna realita prináša. Naším ďalším krokom bola analýza použitia technológií virtuálnej reality na rehabilitáciu a terapiu pacientov. Súčasťou analýzy bol taktiež aj pohľad na vybrané projekty využívajúce virtuálnu realitu. Našou snahou bolo analyzovanie vlastností a technológií využívaných v týchto projektoch. Keďže táto práca nadväzuje na už predošlú sériu prác, bolo potrebné analyzovať aj už existujúci systém. Dôležitým krokom bolo určenie funkcií, ktoré bude potrebné implementovať do nášho systému, ako aj prípadné nedostatky, ktoré je potrebné vyriešiť.

Ďalším krokom bolo vytvorenie prototypu v hernom rámci Unity. Prototyp musel spĺňať niekoľko podmienok, a to: zabezpečenie zdieľaného prostredia v reálnom čase, možnosť použitia helmy na virtuálnu realitu, možnosť komunikácie zvonku, možnosť rozširovať systém na základe rozdielnych rolí. Prototyp bol následne prezentovaný pracovníkom zo Slovenskej akadémie vied. Počas konzultácií s nimi sme sa dohodli na ďalšom smerovaní systému. Vďaka prototypu sme boli schopní hlbšie analyzovať možnosti platformy Unity, ako aj jej obmedzenia, bez nutnosti vytvárania komplexného systému.

Na základe vytvoreného prototypu už existujúcej implementácie v jazyku JavaScript a taktiež aj iných už existujúcich projektov sme následne navrhli a implementovali systém. Základ aplikácie tvorí zdieľané virtuálne prostredie. Používatelia sa v systéme delia na dve skupiny: terapeut a pacient. Úlohou terapeuta je vedenie procesu rehabilitácie a spolupráca s pacientom v reálnom čase. Existujúce riešenie vytvorené v JavaScripte obsahuje jednoduchých avatarov. V našej implementácii sme použili celotelových avatarov, ktorí majú zároveň aj vytvorené

animácie. Oproti existujúcemu riešeniu sme taktiež rozšírili animácie určené na rehabilitáciu pridaním ďalších pohybov. Celkovo systém umožňuje štyri rôzne spôsoby cvičenia, z ktorých každé sa odlišuje vykonávanými pohybmi. Animácie vytvárame dynamicky použitím inverznej kinematiky. Terapeut má taktiež aj možnosť rôzne meniť parametre animácií, ako dĺžka jednotlivých častí animácie alebo pozície pre jednotlivé pohyby. Pri aplikáciách využívajúcich virtuálnu realitu je taktiež dôležitá práca s používateľským rozhraním. V aplikácii sme implementovali niekoľko spôsobov, akými s ním môže používateľ pracovať. Aplikácia okrem podpory heliem na virtuálnu realitu poskytuje možnosť jej použitia terapeutom na klasických stolových počítačoch. Dôležité bolo plnohodnotné poskytnutie rovnakej funkcionality na oboch týchto platformách. Dôležitým aspektom systému je aj prostredie, v ktorom by mala rehabilitácia prebiehať. Našou snahou bolo vytvorenie prostredia, kde by sa pacienti cítili príjemne. V rámci tejto práce sme sa taktiež zúčastnili konferencie 2022 IEEE 16th International Scientific Conference on Informatics, na ktorej bol čiastočný systém testovaný. Na základe testovania sme ďalej systém rozširovali a upravovali.

Neoddeliteľnou časťou tvorby komplexného systému je aj jeho optimalizácia. Vzhľadom na využitie virtuálnej reality, je potrebné venovať optimalizácii obzvlášť výraznú pozornosť. Plynulosť systému a jednoduchosť jeho použitia je veľmi dôležitá pri rehabilitácii pacientov, keďže tieto faktory môžu taktiež vplývať na jej výsledky. Podľa dosiahnutých meraní počas testov, systém funguje plynulo na viacerých platformách. Jediný problém s plynulosťou aplikácie nastal pri simulovaní helmy na virtuálnu realitu na laptope, pri použití integrovanej grafickej karty. Tento mód aplikácie bol ale primárne vytvorený na testovanie aplikácie. Jeho široké využitie nie je plánované, aj z dôvodu jeho zložitejšieho ovládania. Primárnym cieľom je ale zabezpečenie jeho plynulosti počas používania helmy na virtuálnu realitu. Počas testovania sme taktiež server nasadili na cloudovú platformu Amazon Web Services. Následne sme testovali odozvu systému a množstvo dát, ktoré boli pri používaní systému odosielané. Výsledky testov ukázali, že množstvo odosielaných dát nepresiahol hranicu 10KB/s. Počas bežného použitia systému je ale táto hodnota výrazne nižšia, keďže počas testov sme systém zaťažili maximálne. Testovaním sme potvrdili to, že použitie systému je plynulé aj na systémoch so slabším hardvérom.

Jednou z tém, ktorými sa bude do budúcnosti potrebné zaoberať je aj nasadenia a distribúcia aplikácie. Existujú dve alternatívy, distribúcia vlastnou cestou (napríklad github) alebo využitie platformy na distribúciu softvéru (napríklad App Lab, Sidequest). Obe možnosti majú svoje výhody aj nevýhody, preto je po-

trebné zvážiť, ktorý spôsob je v danom prípade najvýhodnejší. Systém umožňuje rozšírenia ako napríklad pridanie ďalších cvičení a vytvorenie komplexných scenárov, ktoré by bolo možné detailnejšie prispôsobiť podľa potrieb pacientov. Ďalšími možnými rozšíreniami sú pridanie hlasovej komunikácie a podpora ďalších jazykov, keďže momentálne je podporovaný iba anglický jazyk. Aktuálne systém ponúka na komunikáciu zvonku REST API. Tento prístup má jedno veľké obmedzenie v podobe smeru, ktorým je možné komunikovať. Server nie je schopný informovať systém OpenVibe o udalostiach, ako napríklad začiatok cvičenia. V budúcnosti je preto potrebné rozšírenie systému pridaním komunikácie pomocou WebSocketov, ktoré umožňujú obojsmernú komunikáciu. Pri rozširovaní systému je nutné myslieť aj na optimalizáciu celého systému. Keďže podpora virtuálnej reality je neustále vo vývoji zo strany Unity, je dôležité aktualizovať knižnice, v ktorých je systém tvorený na nové verzie. Vďaka tomu bude možné využívať nové možnosti počas vývoja ako aj dosiahnuť lepšiu optimalizáciu systémov.

Literatúra

1. HUDÁK, Marián; SOBOTA, Branislav. *Kolaboratívna virtuálna realita a rozhrania systémov*. 2021. ISBN ISBN 978-80-553-3974-0.
2. SOBOTA, Branislav; KOREČKO, Štefan; HUDÁK, Marián; SIVÝ, Martin. Mixed Reality: A Known Unknown. In: SOBOTA, Branislav; CVETKOVIĆ, Dragan (ed.). *Mixed Reality and Three-Dimensional Computer Graphics*. Rijeka: IntechOpen, 2020, kap. 10. Dostupné z DOI: 10.5772/intechopen.92827.
3. STEUER, Jonathan. Defining Virtual Reality: Dimensions Determining Telepresence. *Journal of Communication*. 1992, roč. 42, č. 4, s. 73–93. Dostupné z DOI: 10.1111/j.1460-2466.1992.tb00812.x.
4. FAISAL, Aldo. Computer science: Visionary of virtual reality. 2017, s. 298–299. Dostupné z DOI: 10.1038/551298a.
5. JACK, D.; BOIAN, R.; MERIANS, A.S.; TREMAINE, M.; BURDEA, G.C.; ADAMOVICH, S.V.; RECCE, M.; POIZNER, H. Virtual reality-enhanced stroke rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. 2001, roč. 9, č. 3, s. 308–318. Dostupné z DOI: 10.1109/7333.948460.
6. UEKI, Satoshi; KAWASAKI, Haruhisa; ITO, Satoshi; NISHIMOTO, Yutaka; ABE, Motoyuki; AOKI, Takaaki; ISHIGURE, Yasuhiko; OJIKA, Takeo; MOURI, Tetsuya. Development of a Hand-Assist Robot With Multi Degrees of Freedom for Rehabilitation Therapy. *IEEE/ASME Transactions on Mechatronics*. 2012, roč. 17, č. 1, s. 136–146. Dostupné z DOI: 10.1109/TMECH.2010.2090353.
7. CONNELLY, Lauri; JIA, Yicheng; TORO, Maria L.; STOYKOV, Mary Ellen; KENYON, Robert V.; KAMPER, Derek G. A Pneumatic Glove and Immersive Virtual Reality Environment for Hand Rehabilitative Training After Stroke. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. 2010, roč. 18, č. 5, s. 551–559. Dostupné z DOI: 10.1109/TNSRE.2010.2047588.

8. CHURCHILL, Elizabeth F; SNOWDON, David N; MUNRO, Alan J. *Collaborative virtual environments: digital places and spaces for interaction*. Springer Science & Business Media, 2012.

9. BENFORD, Steve; BOWERS, John; FAHLÉN, Lennart E.; GREENHALGH, Chris; SNOWDON, Dave. User Embodiment in Collaborative Virtual Environments. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995, s. 242–249. CHI '95. ISBN 0201847051. Dostupné z DOI: 10.1145/223904.223935.

10. BIALKOVA, Svetlana; DICKHOFF, Bob. Encouraging Rehabilitation Trials: The Potential of 360° Immersive Instruction Videos. In: *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2019, s. 1443–1447. Dostupné z DOI: 10.1109/VR.2019.8797805.

11. WITMER, Bob G.; SINGER, Michael J. Measuring Presence in Virtual Environments: A Presence Questionnaire. *Presence: Teleoperators and Virtual Environments*. 1998, roč. 7, č. 3, s. 225–240. Dostupné z DOI: 10.1162/105474698565686.

12. LANGE, Belinda; FLYNN, Sheryl; RIZZO, Albert. Game-based telerehabilitation. *European Journal of Physical and Rehabilitation Medicine*. 2009, roč. 45, č. 1, s. 143–151. ISSN 1973-9087.

13. KENNEDY, Robert S.; LANE, Norman E.; BERBAUM, Kevin S.; LILIENTHAL, Michael G. Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *The International Journal of Aviation Psychology*. 1993, roč. 3, č. 3, s. 203–220. Dostupné z DOI: 10.1207/s15327108ijap0303_3.

14. OCULUS VR, Inc. *Oculus VR Best Practices Guide* [online]. 2014 [cit. 2023-03-09]. Dostupné z : https://s3.amazonaws.com/arena-attachments/238441/2330603062c2e502c5c2ca40443c2fa4.pdf.

15. VOURVOPOULOS, Athanasios; PARDO, Octavio Marin; LEFEBVRE, Stephanie; NEUREITHER, Meghan; SALDANA, David; JAHNG, Esther; LIEW, Sook-Lei. Effects of a Brain-Computer Interface With Virtual Reality (VR) Neurofeedback: A Pilot Study in Chronic Stroke Patients. *Frontiers in Human Neuroscience*. 2019, roč. 13. Dostupné z DOI: 10.3389/fnhum.2019.00210.

16. BOHIL, Corey J.; ALICEA, Bradly; BIOCCA, Frank A. Virtual reality in neuroscience research and therapy. *Nature Reviews Neuroscience*. 2011, roč. 12, č. 12, s. 752–762. ISSN 1471-0048. Dostupné z DOI: 10.1038/nrn3122.

17. RIZZO, Albert; HARTHOLT, Arno; GRIMANI, Mario; LEEDS, Andrew; LIEWER, Matt. Virtual Reality Exposure Therapy for Combat-Related Posttraumatic Stress Disorder. *Computer*. 2014, roč. 47, č. 7, s. 31–37. Dostupné z DOI: 10.1109/MC.2014.199.

18. KOK, Brian C.; HERRELL, Richard K.; THOMAS, Jeffrey L.; HOGE, Charles W. Posttraumatic Stress Disorder Associated With Combat Service in Iraq or Afghanistan. *Journal of Nervous and Mental Disease*. 2012, roč. 200, č. 5, s. 444–450. Dostupné z DOI: 10.1097/nmd.0b013e3182532312.

19. FOUNDATION, Mozilla [online]. [B.r.] [cit. 2023-03-24]. Dostupné z : https://labs.mozilla.org/projects/hubs/.

20. HUDÁK, Marián; KOREČKO, Štefan; SOBOTA, Branislav. Enhancing Team Interaction and Cross-platform Access in Web-based Collaborative Virtual Environments. In: *2019 IEEE 15th International Scientific Conference on Informatics*. 2019, s. 000171–000176. Dostupné z DOI: 10.1109/Informatics47936.2019.9119312.

21. JAVORKOVÁ, Sára. *Terapia pomocou technológií virtuálnej reality*. 2021. Dostupné tiež z: https://opac.crzp.sk/?fn=detailBiblioForm&sid=14E8631BC0ED910BF6F4A633AA8B.

22. GVUŠČOVÁ, Jana. *Terapeutický systém na báze technológií virtuálnej reality*. 2022. Dostupné tiež z: https://opac.crzp.sk/?fn=detailBiblioForm&sid=83298E4B591633891884D9B92829.

23. FIREFOX. *A-frame* [online]. [B.r.] [cit. 2023-03-24]. Dostupné z : https://aframe.io/.

24. NETWORKED-AFRAME. *Networked-Aframe* [online]. [B.r.] [cit. 2023-03-24]. Dostupné z : https://github.com/networked-aframe/networked-aframe.

25. UNITY. *Unity documentation* [online]. [B.r.] [cit. 2023-03-09]. Dostupné z : https://docs.unity3d.com/Manual/index.html.

26. THE KHRONOS GROUP INC., Unity. *OpenXR Plugin for Unity* [online]. [B.r.] [cit. 2023-03-12]. Dostupné z : https://docs.unity3d.com/Packages/com.unity.xr.openxr@1.6/manual/index.html.

27. UNITY. *Unity documentation - Animation Rigging* [online]. [B.r.] [cit. 2023-03-09]. Dostupné z : `https://docs.unity3d.com/Packages/com.unity.animation.rigging@1.2/manual/index.html`.

28. BUTTER, Raw. *How to Lerp like a pro* [online]. [B.r.] [cit. 2023-03-24]. Dostupné z : `https://chicounity3d.wordpress.com/2014/05/23/how-to-lerp-like-a-pro/`.

Zoznam skratiek

AR Augmentovaná realita

CVE Kolaboratívne virtuálne prostredie

FBX Filmbox

LOD Level of detail

MR Zmiešaná realita

MSAA Multisample anti-aliasing

URP Universal render pipeline

VR Virtuálna realita

VSYNC Vertical synchronization

XR Rozšírená realita

Slovník

Aliasing je nepresné vykresľovanie grafiky (väčšinou pri nízkom rozlíšení), kedy nie je možné presne zobraziť obraz

Offline scéna je scéna, ktorá slúži na zmenu nastavení a pripojenie používateľa do zdieľaného online prostredia. V tejto scéne používateľ nie je pripojený na server, scéna beží len lokálne.

Online scéna je scéna, v ktorej sú používatelia spoločne v zdieľanom online prostredí. Môžu spolupracovať, navzájom sa vidia. V hrách to predstavuje napríklad online zápas, kde proti sebe súperia hráči.

Zoznam príloh

Príloha A Systémová príručka

Príloha B Používateľská príručka

Príloha C CD médium – záverečná práca v elektronickej podobe,

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Neurorehabilitácia v zdieľanej virtuálnej realite
Systémová príručka

Príloha A

Košice 2023 **Bc. Peter Nehila**

Importovanie projektu

Projekt je potrebné možné stiahnuť a importovať do Unity, napríklad použitím aplikácie Unity Hub. Aplikácie aktuálne používa verziu Unity editora 2021.3.4f1[1], ktorú je možné stiahnuť z oficiálnych stránok Unity. Moduly pre platformy, ktoré je potrebné stiahnuť pri inštalácií sú :

- Android Build Support,
- Windows Build Support,
- Linux Dedicated Server,
- Windows Dedicated Server.

Nie je nutné stiahnuť všetky moduly naraz, keďže je možné ich stiahnuť aj neskôr. Pri prvom otvorení projektu sa automaticky stiahnu všetky potrebné balíčky, ktoré nie sú súčasťou projektu. Po spustení editora je ešte potrebné počkať, kým sa importujú všetky potrebné knižnice a nastavia sa potrebné parametre prostredia. Pre spustenie aplikácie a otestovanie plnej funkcionality je potrebné použiť viacero inštancií (aspoň jeden server a jeden klient). Odporúčame použitie rozšírenia ParrelSync do Unity. Pomocou ParrelSyncu je možné vytvoriť si klony a tie používať ako ďalšie inštancie aj na rovnakom počítači. Veľkou výhodou ParrelSyncu je to, že zmeny vykonané v hlavnej inštancii sa automaticky aktualizujú aj v klonoch. Pri spúštaní aplikácie v editore je potrebné dať si pozor na to, akú scénu máme aktuálne otvorenú. Je potrebné spúšťať aplikáciu so scénou "MenuScene" otvorenou. Server spustíte stlačením ServerOnly vľavo hore (v normálnom builde sa server spustí sám, ale v Editore tomu tak nie je). Klientov pripojiť k serveru je možné priamo použitím menu v scéne. Je možné použitie aj HUD vľavo hore, toto menu je ale len zamýšľané na použitie počas vývoja, nie počas prevádzky. Aplikácie funguje len na Windowse alebo Androide (Oculus). OpenXR nepodporuje DX11 na Linuxe z nejakého dôvodu. Pokusy o spustenie na Linuxe

[1] https://unity.com/releases/editor/qa/lts-releases?version=2021.3&major_version=2021&minor_version=3&page=1

pomocou Protonu alebo Wine-u neboli úspešné. Jediný spôsob, ako poskytnúť funkcionalitu na Linuxe je zmena nastavení (nižšie opísané v sekcii 6).

Ak by bol nejaký problém, napr. nefungujúce pripojenie klientov na server, je potrebné skontrolovať niekoľko parametrov, ktoré môžu pomôcť opraviť tento problém:

- správne IP adresy (je možné použiť localhost) - server aj klienti musia byť nastavený na rovnakú IP adresu,
- antivírusový softvér - niekedy antivírusový softvér blokuje porty (aktuálne server používa port 7777),

Preklad aplikácie

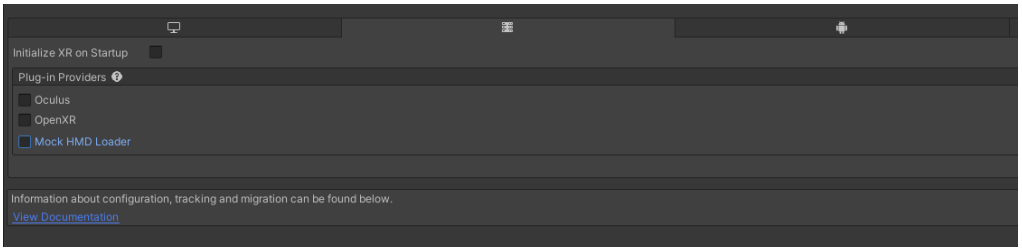
Preklad v Unity sa vykonáva cez menu “File” > “Build Settings”. Otvorí sa okno, kde je možné meniť nastavenia pre preklad. Je dôležité dať si pozor na to, aby boli zvolené všetky scény, ktoré sú v aplikácii používané. Pri preklade aplikácie je potrebné nastaviť XR Plugin, ktorý poskytuje funkcionalitu XR. V aplikácii používame dva pluginy: OpenXR a Mock HMD. OpenXR zabezpečuje fungovanie VR headsetov, Mock HMD Loader používame pri simulovaní XR na desktope.

Preklad pre server

Z ľavej časti je potrebné vybrať možnosť Dedicated Server. Následne je potrebné vybrať Target platform. Je možné použiť aj Linux aj Windows, aplikácia je otestovaná na Windowse primárne, Linux je otestovaný čiastočne. Pred tým, než môžeme vygenerovať potrebné súbory je potrebné zmeniť nastavenia projektu. Do nastavení môžeme prejsť dvoma spôsobmi, v okne určenom na build stlačením tlačidla “Player settings”, alebo cez menu “Edit” > “Project Settings”. Nastavenia kvality a iných špecifikácií už sú vytvorené. Najdôležitejšie nastavenie je ale v karte “XR Plug-in management”. V otvorenom menu je potrebné vybrať podkartu určenú pre dedikované servery (obr. 1). v prípade prekladu pre servery je potrebné vypnúť loader-y. Ak ostanú loader-y povolené predlžuje to čas buildovania a aj veľkosť vygenerovaných súborov. OpenXR loader taktiež nie je kompatibilný s operačným systémom Linux. Tieto loadre nie sú vôbec potrebné pre fungovanie servera, keďže XR modul nikdy nezapíname. Následne je možné zmeniť platformu a spustiť generovanie buildu.

Preklad pre stolový počítač

V okne “Build Settings” zvolíme možnosť pre “Windows, Mac, Linux”. Target platform zvolíme Windows. Ak sme pred tým menili nastavenia pre dedikovaný server je potrebné prejsť do nastavení projektu, koknkrétne “XR Plug-in manage-



Obr. 1: Nastavenia XR Plug-in management pre server

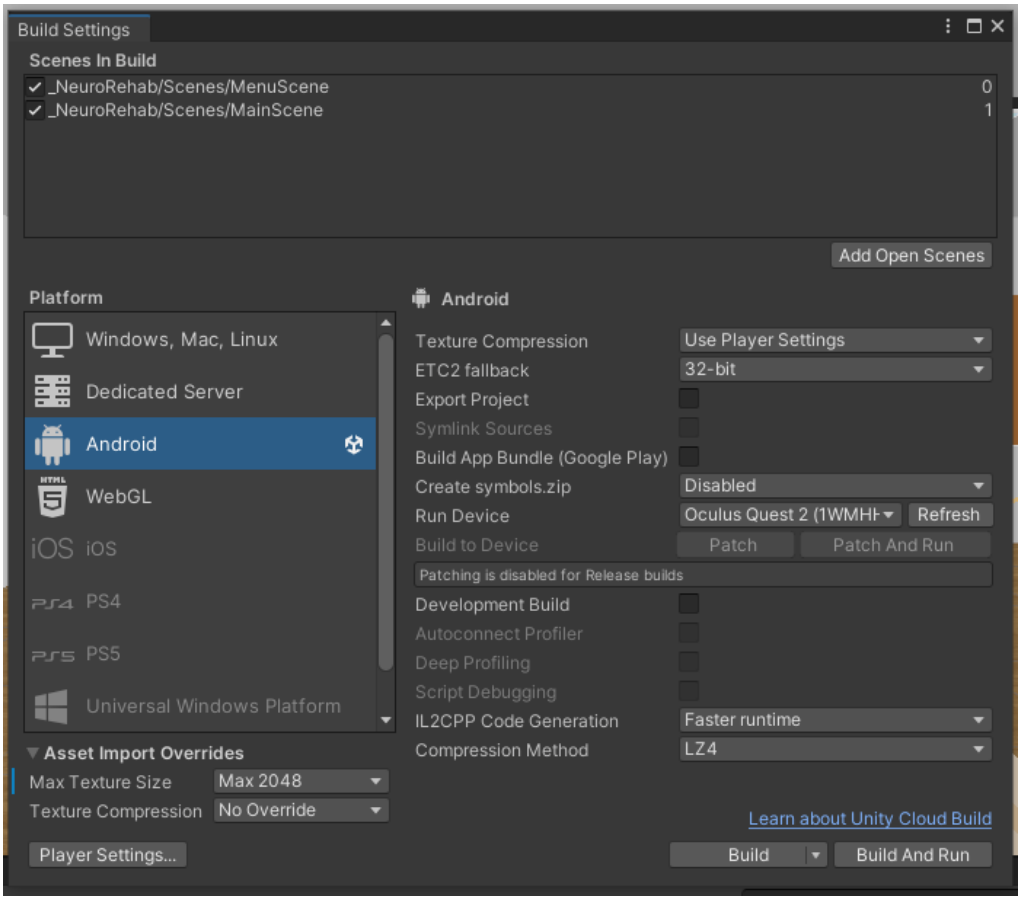
ment" a zapnúť loadre nanovo. Dôvodom je, že pri zmene nastavení "XR Plug-in management" pre server sa zmenia aj pre desktop. Podľa informácií, ktoré sme našli na fóre je to takto "by design". Možno sa to v budúcnosti zmení. OpenXR plugin sa použije v prípade, že používame VR headset pripojený k stolovému počítaču.

Linux

Aj napriek tomu, že OpenXR nefunguje na Linuxe, je možné preložiť aplikáciu bez tohto pluginu. V nastaveniach pre "XR Plug-in management" je potrebné vypnúť loader OpenXR. Problémom je potreba čiastočne prispôsobiť kód. Bez pluginu OpenXR bude možné aplikáciu používať v desktop móde, prípadne použitie simulovaného VR (keďže táto funkcionalita je zabezpečená pluginom Mock HMD). Aplikácie ale nie je otestovaná na Linuxe, takže je potrebné tomu venovať pozornosť v prípade, že chceme ponúkať podporu pre Linux.

Preklad pre Oculus Quest 2

Vytvorenie apk súboru pre Oculus Quest 2 je možné aj bez pripojeného zariadenia. Ak je ale zariadenie aj pripojené je možné hneď nainštalovať aplikáciu priamo na Oculus. Ohľadom loadrov, ktoré sme museli meniť v prípade prekladu na stolový počítač alebo server, to na android nie je potrebné. Nesmie ale byť povolený Mock HMD Loader, keďže jeho spustenie by znefunkčnilo OpenXR plugin. Nastavenia prekladu je možné do určitej miery upravovať. Jedným z nich je Run Device. Ak je Oculus Quest 2 pripojený k počítaču, je možné prepnúť na tento VR headset. Následne Stlačením tlačidla "Build nad Run" sa spustí preklad aplikácie. Nová verzia aplikácie sa zároveň aj nainštaluje na zariadenie. Ďalším nastavením je "IL2CPP Code Generation" . Je možné zvoliť z dvoch možností, možnosť pre rýchlejší beh aplikácie a možnosť pre rýchlejší preklad. Počas vývoja je možné používať druhú možnosť, keďže generuje menšie inštalačné súbory a taktiež aj



Obr. 2: Nastavenia na preklad pre Android

preklad trvá kratšie. Ak chceme inštalovať priamo na Oculus je potrebné mať povolený developer mode. Rozdiel v trvaní prekladu ale nie je príliš signifikantný. Na obr. 2 je možné vidieť nastavenia, ktoré sme používali počas vývoja.

Nasadenie servera

Nasadenie servera je jednoduché, jediné čo je potrebné spraviť je spustiť správny build. Server umožňuje spustenie na konkrétnej IP adrese, takže nie je potrebné vytvárať nový build len kvôli zmene IP adresy (klientske inštancie je potrebné vygenerovať nanovo, aktuálne neponúkajú možnosť konfigurácie). Spustenie servera so špecifickou IP adresou je možné prepínačom `serverip`. Celý príkaz by mohol vyzerať nasledovne:

```
.\NeuroRehabVR.exe -serverip 192.168.1.210
```

Namiesto IP adresy je možné taktiež použiť kľúčové slovo "localhost".

Dokumentácia aplikácie

Kapitola je rozdelená na dve časti, dokumentácia API a dokumentácia zdrojového kódu.

API dokumentácia

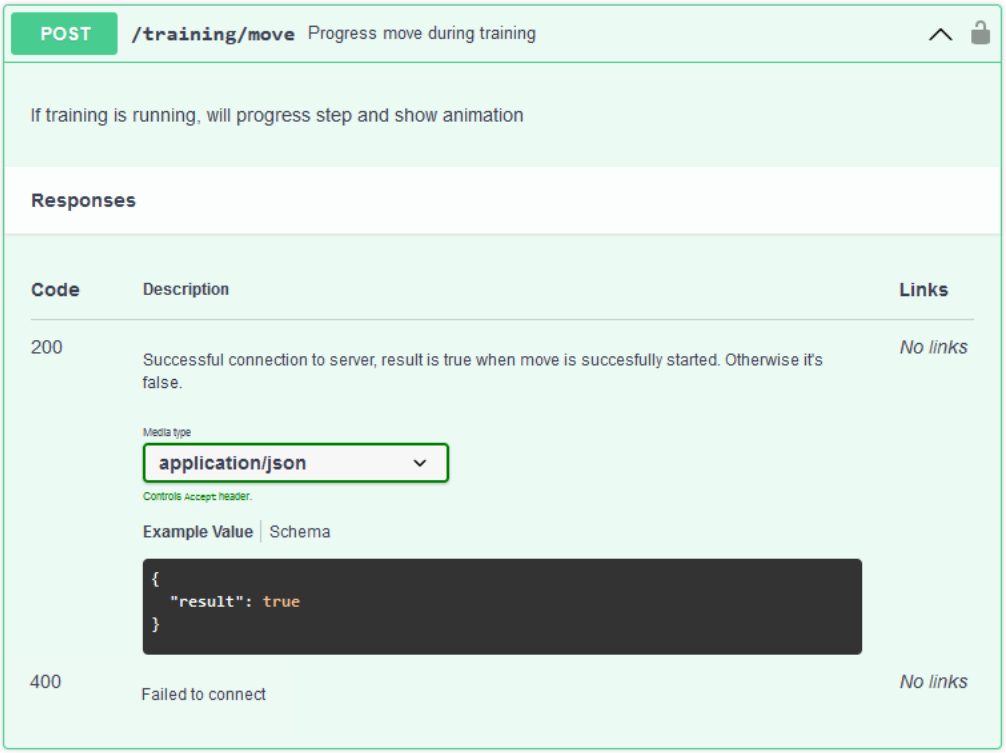
Systém obsahuje prístup pomocou REST API k serveru zvonku. API obsahuje niekoľko referencií. POST Request bude vždy úspešný, ak sa podarí kontaktovať server. Mení sa response body. Hodnota "result" je rovná "true" len v prípade, že sa naozaj podarí vykonať funkciu spustenú POST requestom. Na obr. 3 je vyobrazená referencia na spustenie animácie počas tréningu. Na obr. 4 je request pre obyčajný pohyb, tento je možné vykonať pokiaľ nebeží tréning. Request na obr. 5 zmení aktuálny typ animácie na "Off" a tým skryje objekty určené na animácie. Request `/spawn` slúži na nastavenie typu animácie, typ animácie sa prepne do posledného typu animácie. (obr. 6).

Dokumentácia zdrojového kódu

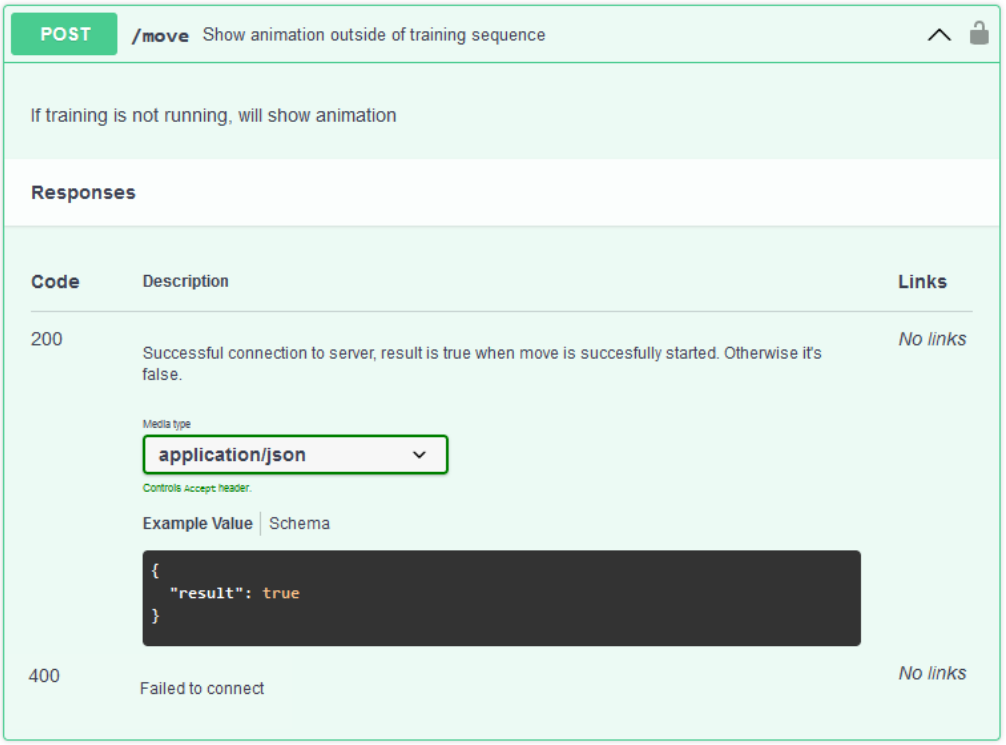
Detailná dokumentácia zdrojového kódu sa nachádza v Prílohe C, pod priečinkom s názvom "Docs". Dokumentácia je dostupná vo viacerých formátoch : HTML, Latex, RTF, XML. Dokumentácia je dostupná aj online, na odkaze `https://nehilap.github.io/NeuroRehabVR/docs/html/index.html`.

Dokumentácia je generovaná pomocou nástroja doxygen[2]. Komentáre sú vytvárané v štýle "XML documentation comments". Tento štýl umožňuje použitie nástrojov na generovanie dokumentácie. Odporúčame dodržiavať jednotný štýl komentovania. Zároveň ale neodporúčame komentovanie každej funkcie a každého riadku kódu, keďže to znižuje prehľadnosť.

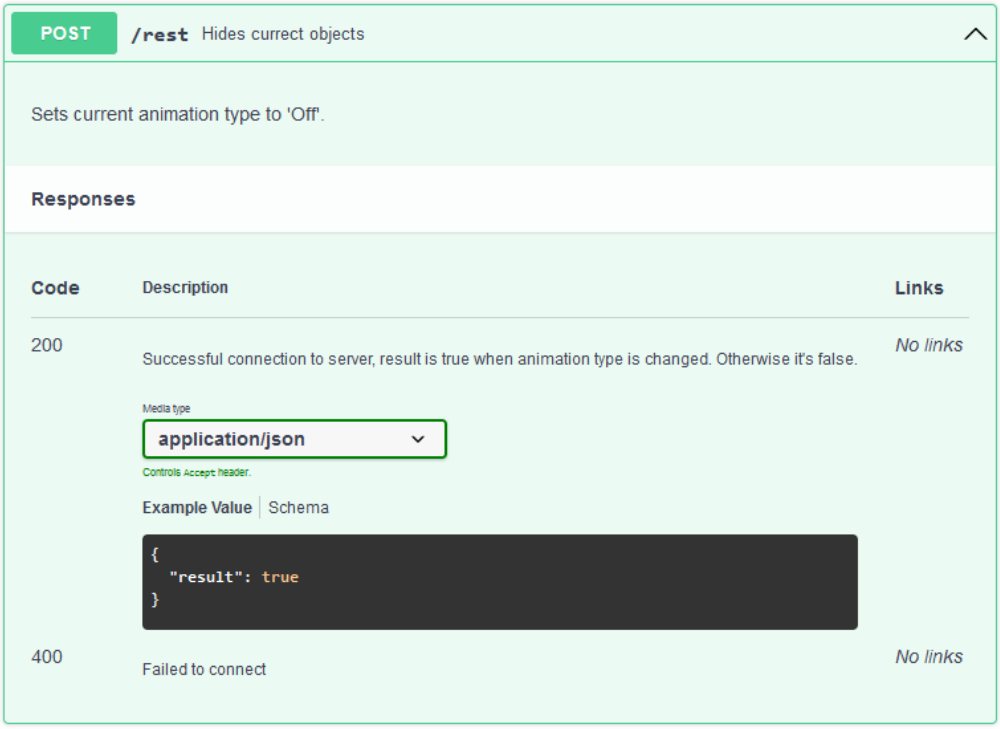
[2] `https://www.doxygen.nl/`



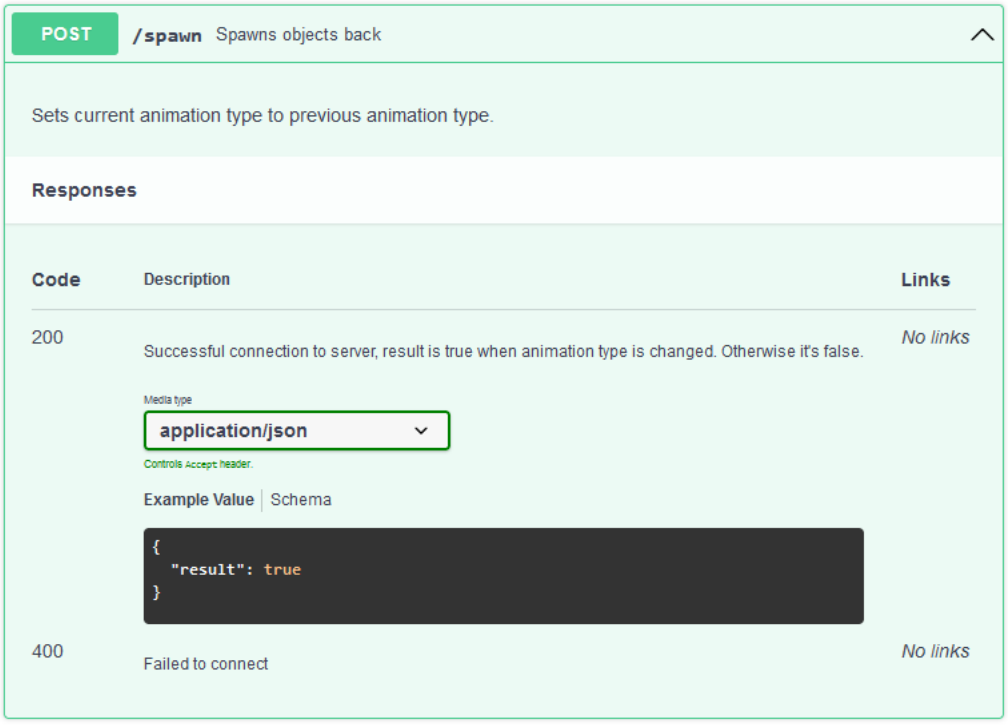
Obr. 3: API dokumentácia - /training/move



Obr. 4: API dokumentácia - /move



Obr. 5: API dokumentácia - /rest



Obr. 6: API dokumentácia - /spawn

Publikovanie aplikácie

Aktuálne aplikácia nie je publikovaná žiadnym spôsobom. Meta ponúka možnosť využitia platformy App Lab. App lab umožňuje využitie Alpha release, podľa našich informácií tento release neprechádza procesom overovania zo strany Mety (nemáme to ale otestované), a teda bolo by možné ho použiť počas vývoja. Platforma SideQuest je taktiež možnou alternatívou (nie je to ale oficiálna platforma). Posledná možnosť je publikácia vlastnou cestou (napríklad využitím githubu). Výhoda je možnosť vydávania viacerých paralelných verzií aplikácie, prípadne vydávanie veľkého množstva verzií pri malých zmenách. Ostatné platformy by mohli mať obmedzenia. Problém ale nastáva v prípade inštalácie, keďže musíme aplikáciu nainštalovať ručne. Je potrebné sa do budúcnosti zamýšľať nad touto témou.

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Neurorehabilitácia v zdieľanej virtuálnej realite
Používateľská príručka

Príloha B

Košice 2023 **Bc. Peter Nehila**

Inštalácia softvéru

Aplikácie aktuálne nie je na žiadnej platforme, bežne používanej na šírenie softvéru. Je preto potrebné zabezpečiť prístup k potrebným súborom cez priame spojenie. Kontakt: `peter.nehila@student.tuke.sk` alebo `stefan.korecko@tuke.sk`. Pre fungovanie je potrebné mať zabezpečený funkčný server a správne nastavené klientske inštancie (IP adresa servera). Aktuálne server nie je verejne dostupný.

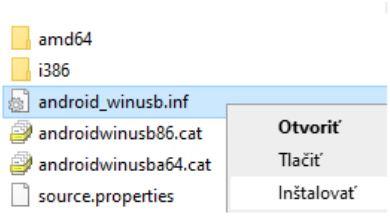
Osobný počítač

Aplikáciu nie je potrebné inštalovať. Spúšťa sa otvorením súbora s názvom `NeuroRehabVR.exe`, ktorý je súčasťou priečinka so všetkými súbormi.

Oculus Quest 2

Aktuálne nie je možné spúšťať a inštalovať aplikáciu cez platformu na šírenie softvéru (ako napríklad Oculus Store). Je potrebné ju ručne nainštalovať. Postup inštalácie obsahuje niekoľko krokov, na inštaláciu softvéru je potrebné použitie stolového počítača. Postup je nasledovný:

1. **USB debugging** - na inštaláciu aplikácií je potrebné zapnúť developer mode. Je možné ho zapnúť dvoma spôsobmi: použitím aplikácie v smartfóne alebo zmenou nastavení priamo v zariadení Oculus. Pri použití aplikácie Meta Quest na smartfóne je potrebné spárovať zariadenia, následne vybrať zariadenie a prejsť do nastavení zariadenia, kde je možné povoliť developer mode. V prípade nastavenia na headsete priamo, je postup nasledovný: je potrebné prejsť do nastavení "Settings", následne otvoriť kartu "System". Ako posledný krok je potrebné prejsť do karty "Developer" na ľavej strane. V tejto karte je potrebné povoliť možnosť "USB Connection Dialog". Následne sa po pripojení zariadenia do počítača na headsete objaví dialóg "Allow USB debugging?". Je potrebné povoliť túto možnosť. Ak plánujete používať počítač v spojení s helmou častejšie, odporúčame povoliť možnosť "Always allow from this computer",



Obr. 1: Inštalácia ovládača Oculus Go

2. **inštalácia ovládačov** - na stolovom počítači je potrebné nainštalovať Oculus Go ovládač[3]. Po stiahnutí adresára, je potrebné ho rozbaliť. Po rozbalení je možné nainštalovať ovládač. Pravým tlačidlom myši na súbor `android_winusb.inf` sa zobrazí možnosť "Inštalovať" (obr. 1),

3. **Android Debug Bridge** - skrátene ADB, slúži na komunikácia s android zariadeniami, pripojenými k počítaču. Keďže Oculus Quest využíva android ako operačný systém, je možné tento nástroj použiť na inštaláciu softvéru. ADB je možné stiahnuť z oficiálnej stránky android[4]. Po stiahnutí a rozbalení priečinka je potrebné prejsť do cieľového priečinka, kde sú potrebné súbory rozbalené (odporúčame použiť iný priečinok než ten z predošlého kroku),

4. **inštalácia aplikácie** - inštaláciu je potrebné uskutočniť cez príkazový riadok. Prejdite do priečinka, kde ste rozbalili ADB nástroje v predošlom kroku. Podržaním tlačidla Shift a stlačením pravého tlačidla myši otvoríte dialógové menu. Zvoľte možnosť "Otvoriť tu okno prostredia PowerShell", prípadne "Otvoriť tu príkazový riadok". Môže sa to líšiť v závislosti od zariadenia. Následne je potrebné zadať príkaz:
   ```
   .\adb.exe install -r .\Build.apk
   ```
   ".\Build.apk" je potrebné nahradiť celou cestou k súboru aplikácie. Príkaz s celou cestou by mohol vyzerať nasledovne:
   ```
   .\adb.exe install -r D:\Unity\NeuroRehabVR\Oculus\Build.apk
   ```
   Pre jednoduchosť je preto možné uložiť inštalačný súbor "Build.apk" do rovnakého priečinka, v ktorom sú uložené ADB nástroje. Následne je možné použiť relatívnu cestu, ako je ukázané v prvom príklade príkazu,

5. **spustenie aplikácie** - po inštalácii sa aplikácia objaví medzi nainštalovaným softvérom. Je možné ju nájsť v priečinku "Unknown sources".

[3] `https://developer.oculus.com/downloads/package/oculus-go-adb-drivers/`
[4] `https://developer.android.com/studio/releases/platform-tools`

Použitie a ukážka aplikácie

Po spustení aplikácie sa používateľ nachádza v offline lobby scéne. Používateľ môže meniť nastavenia, prispôsobiť vzhľad avatara, ako aj resetovať výšku avatara. Pohyb v prostredí závisí od platformy, na ktorej sa nachádzame.

V prípade **osobného počítača** je možné ovládať pohyb pomocou klávesov "WASD" alebo kurzorových šipiek, otáčanie funguje pomocou myši, interakcia s objektmi (UI aj 3D objekty) pomocou ľavého tlačidla myši. Menu je možné ovládať priamo stlačením tlačidiel v 3D priestore, alebo je možné zobraziť 2D menu na vrchu obrazu. Po stlačení skratky na zobrazenie menu (klávesa "M") sa odomkne myš a ovládanie menu je možné štandardným spôsobom. Na obr. 4.20 je možné vidieť vzhľad menu v tomto móde. Presúvanie objektu je implementované jeho presúvaním myšou a zároveň držaním ľavého tlačidla myši. Presúvať objekt je možné v bežnom móde pohybu, stlačením klávesy "G" je možné uzamknúť rotáciu kamery. Zároveň sa odomkne kurzor myši. Pred pripojením má používateľ možnosť voľby roly, v ktorej bude vystupovať počas rehabilitácie. Vzhľad úvodného menu je na obr. 2.

Počas používania **VR helmy** je možný pohyb pomocou chôdze, alebo použitím páčky ľavého ovládača. Otáčanie je možné pomocou páčky pravého ovládača, ale vzhľadom na pocit nevoľnosti, ktorý spôsobuje odporúčame otáčať sa len celým telom. Interakcia s menu elementami je možná stlačením hlavného tlačidla ovládača, alebo použitím "Trigger" tlačidla. Interakcia funguje oboma ovládačmi. Presun objektov je možný zamierením ovládača na objekt, alebo dotknutia sa objektu a držaním tlačidla "Grip" ovládača (V prípade Oculus Quest ovládačov to sú tlačidlá na stranách). Zobrazenie menu nad ovládačom je možné pomocou tlačidla menu na ľavom ovládači. Podržaním tohto tlačidla sa objaví menu pred používateľom, ktoré zostane prilepené na danej pozícii.

Obr. 2: Úvodné menu

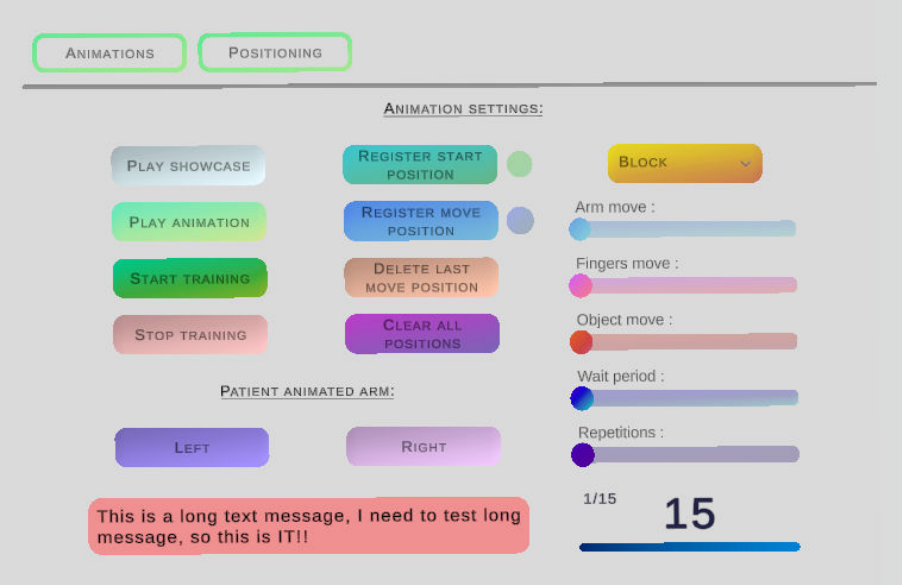
Rola pacienta

Pacient má počas cvičenia značne obmedzené možnosti. Môže sa hýbať v prostredí, no jeho prácou je len počkať na pokyny terapeuta. Po pripojení je automaticky zvolená ruka, ktorou sa vykonáva cvičenie pravá. Toto nastavenie môže meniť terapeut po pripojení do online prostredia.

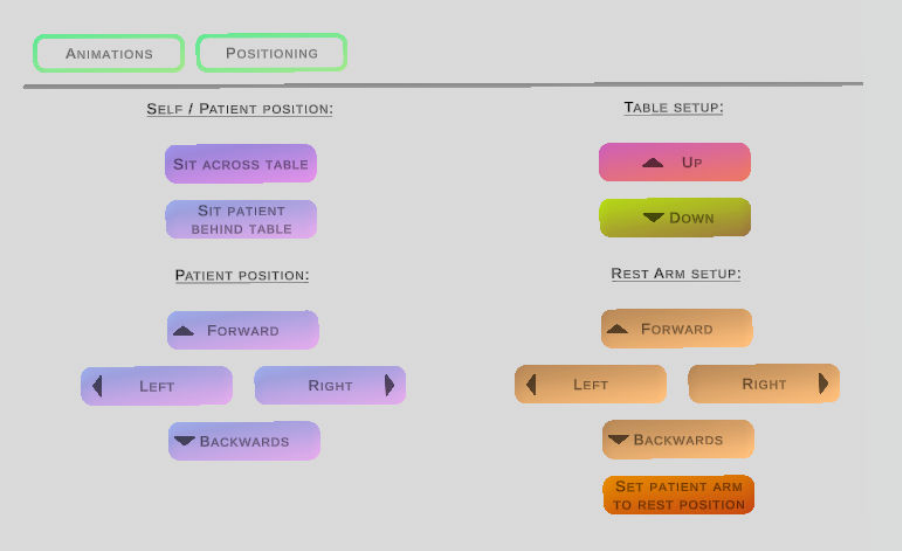
Rola terapeuta

Terapeut má za úlohu viesť celú rehabilitáciu. V porovnaní s rolou pacienta má tiež po pripojení svoje vlastné menu, kde sa nachádzajú nastavenie týkajúce sa rehabilitácie, ako aj pozícií počas cvičenia. Terapeut môže zmeniť aktívnu ruku pacienta, ktorá je rehabilitovaná. Na obr. 3 je možné vidieť nastavenia, ktoré sú dostupné počas cvičenia terapeutovi. Môže meniť typ pohybu, ktorý je aktuálne používaný. Na výber je zo štyroch rôznych pohybov. Každý pohyb je špecifický predmetom, ktorý sa pri pohybe používa. Terapeut je taktiež schopný presúvať predmet a pridávať ďalšie pohyby pre rehabilitáciu. Dôležité je aj nastavenie časovaní pre animácie, čas čakania alebo aj počet opakovaní. Tieto nastavenia je možné meniť v pravej časti menu. Pred spustením rehabilitácie môže terapeut tiež aj zobraziť ukážkovú animáciu. Po spustení rehabilitácie bude následne server čakať na udalosť z prostredia OpenVibe. Po prijatí správy sa spustí animácia na každom klientovi. Ak sa nepošle animácia v rozmedzí čakania, celé aktuálne cvičenie sa preruší a je potrebné ho znova spustiť.

Podstatnú časť nastavení ešte pred rehabilitáciou tvorí prispôsobenie správnych pozícií. Na obr. 4 je možné vidieť možnosti, ktoré terapeut má. Môže presu-



Obr. 3: Menu nastavení rehabilitácie



Obr. 4: Menu s nastaveniami pozícií

núť svoju postavu bez nutnosti pohybu za stôl. Taktiež dokáže presunúť za stôl aj pacienta. To znamená, že pacient a terapeut sa nemusia fyzicky presúvať. Nie je potrebné sa presúvať ani pomocou ovládačov. Terapeut taktiež dokáže precíznejšie meniť pozíciu pacienta jeho posúvaním vpred, vzad a do strán. Systém taktiež ponúka aj možnosť kalibrácie výšky stola a zmenu oddychovej polohy ruky pacienta.

Terapeut je schopný nastaviť viacero pozícií pre pohyb počas animácie. Systém ale má nastavených niekoľko obmedzení. Maximálny počet pozícií pre pohyb je 10, animácia potrebuje mať nastavenú aspoň jednu pozíciu (začiatočnú) aby ju bolo možné spustiť, pozícia sa použije len v prípade, že je v dosahu ruky pacienta.

Obr. 5: Prostredie s pohľadom na stôl

To znamená, ak sa pacient pohne počas rehabilitácie a pozícia už nie je v jeho dosahu, pozícia sa preskočí. Animácia s kľúčom a zámkou má trochu iné nastavenia. Animácia musí mať vždy dve pozície, aby ju bolo možné spustiť začiatočnú pozíciu a cieľovú pozíciu zámku.

Ukážka systému

V tejto časti ukážeme zopár obrázkov, kde je možné vidieť systém a niektoré vlastnosti, ktorými disponuje.

Vzhľad prostredia

Na obr. 5 a 6 je vyobrazené prostredie, v ktorom prebieha rehabilitácia. Na stole je možné vidieť objekt, ktorý sa aktuálne používa počas rehabilitácie. Pri zmene typu animácie sa nový objekt zobrazí na stole.

Ukážka animácií

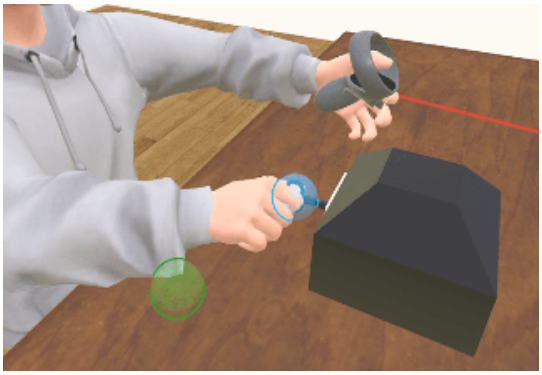
Na pod-obrázkoch obr.7 je možné vidieť rozdielne pohyby počas vykonávania animácií. Na obr. 7d je taktiež možné vidieť ukazovatele, ktoré vyznačujú pozície pre pohyby. Zelená guľa predstavuje štartovaciu pozíciu, modrá ostatné pozície. Tieto značky vidí len terapeut. Ako bolo spomenuté v predošlej kapitole, pozície animácií sú obmedzené dosahom ruky pacienta. Na obr. 8 je možné vidieť ukazovateľ dosahu. Tento ukazovateľ je viditeľný len terapeutom počas presúvania objektu. Na obr. 9 je vidieť animáciu z pohľadu pacienta.

Obr. 6: Zvyšok miestnosti

(a) Uchopenie a presun kvádra

(b) Uchopenie kocky

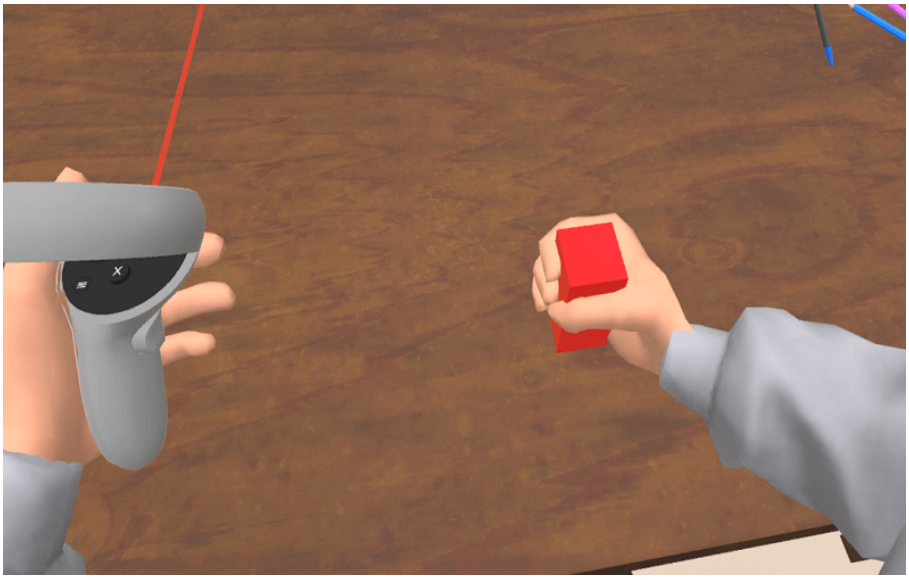
(c) Zdvihnutie pohára



(d) Presunutie kľúča do zámky

Obr. 7: Ukážka rôznych typov animácií

Obr. 8: Dosah ruky pacienta



Obr. 9: Animácia z pohľadu pacienta