Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Kolaborácia vo virtuálnej realite: vstupno-výstupná časť pre inteligentné telefóny

Diplomová práca

2020 **Bc. Tomáš Balluch**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Kolaborácia vo virtuálnej realite: vstupno-výstupná časť pre inteligentné telefóny

Diplomová práca

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

Košice 2020 **Bc. Tomáš Balluch**

Názov práce: Kolaborácia vo virtuálnej realite: vstupno-výstupná časť pre inteligentné telefóny

Pracovisko: Katedra počítačov a informatiky, Technická univerzita v Košiciach
Autor: Bc. Tomáš Balluch
Školiteľ: Ing. Štefan Korečko, PhD.
Konzultant: Ing. Marián Hudák
Dátum: 15. 4. 2020
Kľúčové slová: VR, virtuálna realita, webová stránka, javascript, A-Frame, sledovanie pohybu

Abstrakt: Hlavným cieľom tejto práce je navrhnúť a implementovať prostredie kolaboratívnej virtuálnej reality určené pre inteligentné telefóny. Toto prostredie má byť charakteristické možnosťou pripojenia viacerých užívateľov, podporou sledovania orientácie či polohy telefónu a má byť vyvinuté použitím webových technológií. Užívateľ by teda po navštívení webovej stránky tohto VR prostredia bol schopný vidieť ostatných užívateľov a mohol by sa v ňom pohybovať meneným pozície smartfónu. Na vývoj bol použitý framework A-Frame, pomocou ktorého boli vyriešené určité aspekty prostredia. Nepodporoval však sledovanie pohybu bez značiek, ktoré muselo byť do prostredia doimplementované. Práca sa venuje analýze možných spôsobov sledovania pohybu zariadenia, výberu vhodnej technológie, návrhu riešenia, implementácií tejto technológie do webovej VR aplikácie prostredníctvom vhodne zvoleného webového vývojového prostredia a zhodnoteniu výsledného riešenia. Vznikne tak webová aplikácia umožňujúca vykresľovať virtuálne prostredie a sledovať polohu telefónu prostredníctvom jeho kamery v reálnom čase, kombinujúca tak vlastnosti virtuálnej a rozšírenej reality. Práca je súčasťou kolaborácie troch prác, kde ďalšia rieši podporu VR prostredia pre headset Microsoft HoloLens a posledná má na starosti riadiacu a komunikačnú časť aplikácie.

Thesis title:	Collaboration in Virtual Reality: Input-Output Component for Smartphones
Department:	Department of Computers and Informatics, Technical University of Košice
Author:	Bc. Tomáš Balluch
Supervisor:	Ing. Štefan Korečko, PhD.
Tutor:	Ing. Marián Hudák
Date:	15. 4. 2020
Keywords:	VR, virtual reality, website, javascript, A-Frame, spatial tracking

Abstract: The main purpose of this thesis is design and implementation of a collaborative virtual reality environment built for smartphones. It should be characterized by the ability of connecting several users, support for spatial tracking and being developed by web technologies. A user would be able to see other users by visiting the website of the VR application and could move around freely by changing the position of the phone. A-Frame was used to develop the application, which helped solving some aspects of the environment. However, it did not support markerless tracking that had to be implemented additionally. This thesis is devoted to analysis of possible ways of spatial tracking of a smartphone, selection of a suitable technology, design of a solution, implementation of the technology into web VR application through properly selected web development environment and evaluation of the resultant solution. As a result, a web-based application will be created that is able to render a virtual environment and track phone´s position through its camera in real time, combining features of virtual and augmented reality. This thesis is part of a collaboration of three papers, where the others are dedicated to implementing support for Microsoft HoloLens headset and the management part of the application.

57826

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

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

Názov práce:

Kolaborácia vo virtuálnej realite: vstupno-výstupná časť pre inteligentné telefóny

Collaboration in Virtual Reality: Input-Output Component for Smartphones

Študent: **Bc. Tomáš Balluch**
Školiteľ: **Ing. Štefan Korečko, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce: **Ing. Marián Hudák**
Pracovisko konzultanta: **Katedra počítačov a informatiky**

Pokyny na vypracovanie diplomovej práce:

1. Oboznámiť sa s možnosťami moderných inteligentných telefónov a ich vstupno-výstupných zariadení a analyzovať možnosti a existujúce prístupy ich využitia v kolaboratívnej virtuálnej realite (KVR).
2. Na základe analýzy navrhnúť a implementovať klientskú časť systému KVR, určenú pre inteligentné telefóny a to s použitím webových technológií.
3. Pri návrhu a implementácii spolupracovať s riešiteľmi/kami záverečných prác, pracujúcich na ďalších častiach systému KVR.
4. Experimentálne overiť použiteľnosť implementovaného riešenia a porovnať ho s podobnými existujúcimi riešeniami.
5. Vypracovať dokumentáciu podľa pokynov vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 04.05.2020
Dátum zadania diplomovej práce: 31.10.2019

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY

12.

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

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

Košice, 15.4.2020 .....

Vlastnoručný podpis

Poďakovanie

Na tomto mieste by som sa chcel poďakovať môjmu vedúcemu práce, rovnako ako aj môjmu školiteľovi za ich čas, nápady, užitočné rady a celkovú spoluprácu počas písania tejto práce.

Takisto sa chcem poďakovať mojej rodine, kamarátom, učiteľom, profesorom a známym ktorí pri mne stáli počas celého môjho štúdia.

Obsah

Zoznam obrázkov

Zoznam zdrojových kódov

Slovník Termínov

VR Virtuálna realita - počítačom generované prostredie simulujúce skutočnosť

IDE Integrated Development Environment - prostredie určené na vývoj softvéru

AR Obohatená realita - prostredie spájajúce virtuálny svet a skutočnosť

Headset Okuliare pre virtuálnu realitu, často obsahujúce aj slúchadlá, senzory a ovládače

CAVE Cave Automatic Virtual Environment - prostredie virtuálnej reality v tvare kocky veľkosti miestnosti, v ktorej sú steny tvorené monitormi alebo projektormi

Engine Vývojové prostredie určené na vytváranie počítačových hier

Nickname Prezývka na internete

Renderovanie Počítačové vykresľovanie

Framework Softvér slúžiaci ako podpora pri programovaní s cieľom zjednodušenia vývoja, taktiež nazývaný softvérový rámec

API Application Programming Interface - zbierka procedúr, funkcií alebo tried určitej knižnice, ktoré môže vývojár používať

Bug Chyba v programe, kedy nepracuje očakávaným spôsobom

FPS Frames per Second - Počet obrázkov za sekundu, frame rate

Úvod

Dopyt po prostrediach virtuálnej reality za posledné desaťročie výrazne vzrástol a VR aplikácie sa rozšírili do početnej skupiny domén či priemyslu, ako napríklad vzdelávanie, medicína, architektúra či zábava. Tak ako sa v 80-tych rokoch stal počítač postupne dostupným pre širokú verejnosť, sa v súčasnosti vyrába čoraz viac zariadení podporujúcich VR obsah určených pre domáci sektor. Účelom tejto diplomovej práce je v spolupráci s ďalšími dvoma prácami vytvoriť a otestovať multi-platformové virtuálne prostredie použitím dostupných a moderných technológií, zdieľané viacerými používateľmi prostredníctvom internetu, ktorí môžu toto prostredie vnímať pomocou monitora alebo headsetov pre virtuálnu alebo zmiešanú realitu, ako je Microsoft Hololens, Google Cardboard, smartfón alebo počítač. Táto práca, ako časť kolaborácie, je zameraná na stále sledovanie orientácie a pozície zariadenia, spracovanie vstupov od používateľa a vykresľovanie grafického výstupu virtuálnej scény, a to pomocou podporovaných inteligentných mobilných telefónov s operačným systémom Android s možnosťou kombinácie s VR okuliarmi typu Google Cardboard. Virtuálne prostredie je vytvorené spolu s ďalšími dvoma riešiteľmi záverečných prác, ktorých úlohou je ho optimalizovať, otestovať a poskytnúť podporu pre systém rozšírenej reality Microsoft HoloLens, a poskytnúť základnú riadiacu a komunikačnú časť aplikácie, ktorá bude po dokončení prostredia spustená na serveri v LIRKIS laboratóriu nachádzajúcom sa na Technickej univerzite v Košiciach. Nakoniec tak vznikne prostredie virtuálnej reality, v ktorom sa budú môcť používatelia pohybovať pomocou senzorov zariadenia a budú schopní vzájomnej interakcie. Aj keď headsety virtuálnych realít ako je Microsoft HoloLens nie sú zatiaľ tak populárne kvôli vysokým obstarávacím nákladom, inteligentné telefóny s operačným systémom Android majú vysoký podiel na celosvetovom trhu a VR podporu má čoraz viac týchto zariadení, čím sa virtuálne prostredia stávajú dostupnejšími čím ďalej tým širšiemu spektru ľudí.

1 Formulácia úlohy a ciele práce

Ako bolo spomenuté v úvode, hlavným cieľom tejto práce je rozšíriť prostredie virtuálnej reality s názvom LIRKIS G-CVE pre inteligentné telefóny pomocou moderných technológií s vlastnosťami ako podpora pripojenia viacerých používateľov prostredníctvom internetu alebo schopnosť neustáleho sledovania orientácie a relatívnej pozície zariadenia v priestore. Táto hlavná úloha je delená na niekoľko podcieľov, ktorým sa budú venovať nasledujúce kapitoly tejto práce. Prvým podcieľom je získanie teoretických poznatkov ohľadne možností moderných inteligentných telefónov a ich vstupno-výstupných zariadení v systémoch virtuálnych realít a analýza možností rôznych druhov detekcie pohybu týchto zariadení. Ďalším podcieľom je analýza požiadaviek aplikácie, rozbor a výber vhodného vývojového prostredia pre VR systém. Taktiež sa bude práca venovať experimentálnej analýze kvality rôznych metód sledovania polohy mobilného zariadenia, ich ohodnoteniu, vzájomnému porovnaniu a výberom tej najvhodnejšej pre použitie v samotnej aplikácii. Následne bude potrebné na základe doposiaľ nazbieraných poznatkov vytvoriť návrh riešenia pre klientskú časť viac-používateľského VR prostredia určeného pre inteligentné telefóny. Nasledujúcim podcieľom je samotná implementácia tohto prostredia podľa návrhu riešenia s použitím vhodne zvolených webových technológií. Keďže je táto práca súčasťou kolaborácie viacerých záverečných prác, bude potrebné pri návrhu a implementácii spolupracovať s ďalšími riešiteľmi, ktorých cieľmi sú poskytnúť podporu tohto VR prostredia pre headset Microsoft HoloLens a vytvoriť riadiacu a komunikačnú časť samotnej VR aplikácie. Ďalším podcieľom je po vyhotovení experimentálne overiť dôležité funkcie a vlastnosti, ohodnotiť celkovú použiteľnosť implementovaného riešenia a porovnať ho s podobnými existujúcimi riešeniami z hľadiska jeho dôležitých aspektov. Posledným podcieľom je vypracovanie dokumentácie výsledného riešenia vo forme používateľskej a systémovej príručky VR prostredia.

2 Detekcia pohybu vo virtuálnom prostredí

Táto kapitola sa zameria na analýzu problematiky detekcie pohybu prostredníctvom inteligentných telefónov a VR headsetov vo virtuálnej alebo zmiešanej realite a rozoberie možnosti a využitie moderných telefónov a ich vstupno-výstupných zariadení v systémoch VR. Zameria sa na analýzu a vysvetlenie pojmov úzko súvisiacich s virtuálnou realitou a takisto opíše rôzne spôsoby detekcie pohybu celého tela alebo headsetov či ovládačov. Bližšie opíše metódy sledovania pohybu ako sú sledovanie na základe značiek alebo bez značiek a neskôr vysvetlí pojmy ako gyroskop, akcelerometer či magnetometer, čo sú hlavné senzory umožňujúce sledovanie polohy headsetov a smartfónov. Nakoniec kapitola vysvetlí pojem stupne voľnosti v systémoch virtuálnych realít.

2.1 Úvod do virtuálnej reality

Virtuálna realita je technológia ktorá umožňuje prepojiť používateľa so simulovaným prostredím, často dovoľujúca ich vzájomnú interakciu. Táto technológia vytvára väčšinou ilúziu skutočného alebo fiktívneho sveta. Tento svet je tvorený hlavne vizuálnym, ale aj sluchovým či hmatovým vnemom vytvárajúcich virtuálny svet, a to prostredníctvom špecializovaných zariadení ako sú hlavne VR headsety alebo miestnosti tvorené veľkým počtom obrazoviek. Súčasťou môžu byť aj rôzne periférne zariadenia, ktoré slúžia buď na sledovanie polohy používateľa, ako rôzne senzory či kamery, alebo poskytujú interakciu s virtuálnym prostredím, ako napríklad ovládače. Systémy, ktoré pôsobia na používateľa pomocou vibrácií sa nazývajú hmatové (haptické) systémy. Hmatová informácia sa často označuje ako silová spätná väzba, z angl. "force feedback". Výsledkom tých najprepracova-

nejších VR systémov je, že používateľ je nie len schopný sa vo virtuálnom prostredí rozhliadať, ale aj ho počuť, pôsobiť naňho alebo ho do istej miery ovplyvňovať, čím sa vytvára realistický zážitok podobný skutočnému svetu. Pre čo najvernejšie pohltenie sú pre VR systémy veľmi dôležité faktory ako rozlíšenie obrazovky, zorné pole, obnovovacia frekvencia, oneskorenie pohybu alebo synchronizácia výstupov, najčastejšie obrazu a zvuku [1]. Virtuálna realita sa začala komerčne používať a pomaly stávať masovo dostupnejšou od roku 2010, keď bol predstavený prototyp VR headsetu s názvom Oculus Rift [2]. Určitou formou virtuálnej reality sú aj systémy rozšírených realít, často označované ako AR systémy (z angl. augmented reality). Takéto systémy kombinujú virtuálny a skutočný svet, kedy sú schopné „pridať" virtuálny objekt do skutočnej scény, ktorá je snímaná pomocou kamery zariadenia.

2.1.1 Formy virtuálnej reality

V dnešnom svete existuje viacero foriem virtuálnej reality, z ktorých sú niektoré viac prepracovanejšie ako ostatné, čím ponúkajú kvalitnejší zážitok. Medzi formy VR patria:

- virtuálna realita založená na simulácií,
- virtuálna realita založená na projekcii,
- virtuálna realita založená na stolových počítačoch,
- displeje (headsety) virtuálnych realít,
- a zmiešaná realita.

Prvou je virtuálna realita založená na simulácií. Tu patria napríklad simulátory vedenia motorového vozidla, ktoré poskytujú používateľovi dojem šoférovania skutočného vozidla tým, že systém predpovedá pohyb vozidla spôsobený používateľovým vstupom, a poskytuje mu zodpovedajúce podnety, ako obraz či zvuk. Ďalšou formou je VR založená na projekcii, ktorá vytvára prostredie pomocou 3D projekcií v systéme CAVE. Používatelia na sebe zvyčajne musia mať 3D okuliare, a môže ich byť v prostredí viacero. Tento VR prístup sa využíva vo vizualizácií dizajnu, architektonických príručkách alebo v priemyselnom školení. Jeden používateľ má ovládač, ktorým interaguje s virtuálnym prostredím a jeho

objektami. Podobnou formou je virtuálna realita založená na stolových počítačoch, ktorá je založená na zobrazovaní 3D prostredia na bežnom monitore bez pomoci akéhokoľvek špecializovaného zariadenia na sledovanie polohy. Ako príklad možno uviesť moderné videohry z pohľadu prvej osoby. Tento typ VR však neponúka veľké možnosti oproti ostatným, a má nevýhody ako slabé priestorové videnie. Jednou z najpoužívanejších foriem VR sú displeje pripevnené na hlave, alebo HDM (z angl. head mounted displays, alebo headset).Tieto headsety poskytujú kvalitnejší zážitok ako stolové počítače, a využívajú sa vo viacerých sférach, vrátane hier, simulácií [3], letectva, medicíny [4] alebo inžinierstva [5]. Väčšinou obsahujú dve menšie obrazovky s vysokým rozlíšením, ktoré poskytujú samostatný obraz pre každé oko, čo slúži na vykreslenie stereoskopického obrazu, zvukový systém a senzory pre sledovanie polohy a rotácie hlavy v reálnom čase. Niektoré headsety poskytujú hardvér pre sledovanie polohy očí. Existujú aj typy okuliarov s polopriehľadnými sklami, ktoré umožňujú používateľovi pozerať cez ne [6], alebo také ktoré obsahujú dodatočné príslušenstvo ako ovládače s hmatovou spätnou väzbou. Medzi túto formu virtuálnej reality patria aj headsety navrhnuté pre mobilné zariadenia. Narozdiel od headsetov s integrovanými zobrazovacími jednotkami spomínanými vyššie, je takýto typ skôr krytom alebo puzdrom, do ktorých je možné vložiť inteligentný telefón. Ten sa potom chová ako počítač, ktorý zobrazuje obsah cez jeho displej a využíva svoje integrované senzory pre získavanie polohy používateľa alebo iné dáta. Na displej telefónu sa používateľ pozerá skrz šošovky, ktoré pôsobia ako stereoskop a poskytujú mu široké zorné pole a ostrý pohľad na smartfón aj z veľmi krátkej vzdialenosti. Súčasťou prepracovanejších high-end headsetov sú zväčša aj ovládače. To platí pre samostatne fungujúce headsety, ale aj tie, ktoré pre správne fungovanie potrebujú telefón. Sú to bezdrôtové diaľkové ovládače, ktoré obsahujú senzory na sledovanie svojej polohy alebo orientácie. Používajú sa na interakciu s VR aplikáciami a hrami, kde slúžia hlavne ako ukazovatele. Tieto ovládače sú štandardne dodávané dva, jeden pre každú ruku. Takto môžu byť simulované obidve ruky vo virtuálnom prostredí. Ovládače sú rozmerovo podobné kancelárskej myši a obsahujú obvykle niekoľko tlačidiel, ako tlačidlo Spať alebo Návrat na domácu obrazovku, spúšťače slúžiace na uchopenie predmetu alebo na strieľanie a joystick alebo dotykovú obrazovku, ktorá môže nadobúdať funkciu dotykovej plochy smartfónov. Obvykle tieto ovládače obsahujú polohové senzory a systém pre detekciu gest prstov a rúk, ktoré môže

používateľ robiť pri ich držaní. Niektoré ovládače disponujú sadou infračervených LED diód, ktoré sú sledované kamerami napríklad na headsete, a umožňujú presne sledovať ich pozíciu v priestore. Pre poskytovanie hmatovej spätnej väzby a vyššej formy pohltenia môžu obsahovať aj vibračné motorčeky. Na obrázku 2.1 je znázornený ovládač Oculus Go, obsahujúci dotykovú plochu, dve predné tlačidlá a spúšť na zadnej strane.

Obr. 2.1: Ovládač Oculus Go [7]

Zmiešaná realita (z angl. Mixed reality) je typ virtuálnej reality, ktorá spája digitálny, počítačom generovaný obsah s tým, čo používateľ v skutočnosti vidí vo svojom okolí. Zmiešaná realita sa delí na rozšírenú realitu a rozšírenú virtualitu. Rozšírená virtualita (z angl. Augmented virtuality) je technológia, ktorá vkladá skutočné objekty do syntetického sveta. Pri pohybovaní sa v umelo vytvorenom svete môže používateľ interagovať s virtuálnymi alebo reálnymi objektami [8]. Na druhej strane, rozšírená realita, skrátene AR, je opačná technológia, teda pridáva počítačom vytvorené objekty do reálneho sveta, ktorý sa sníma zvyčajne pomocou kamery telefónu. Výsledné digitálne obrázky zmiešavajú trojrozmerné virtuálne objekty so skutočným prostredím, čím vznikne dojem existencie týchto syntetických objektov v reálnom svete.

2.1.2 Nedostatky headsetov virtuálnych realít

Aj keď sú VR headsety jedným z najpoužívanejších foriem virtuálnej reality, majú určité nedostatky oproti ostatným technológiám. Musia mať napríklad výrazne nižšiu latenciu, teda čas potrebný na prejavenie vizuálneho efektu zo vstupu používateľa, ako bežné videohry. Ak headset nestihne dostatočne rýchlo reagovať na zmenu pohybu používateľa, tak mu môže prísť nevoľno. Ideálna latencia vo VR headsetoch je 7-15 milisekúnd, a jej hlavnou súčasťou je obnovovacia frekvencia obrazovky alebo obrazoviek, pri ktorých by táto hodnota mala byť pre ideálny herný zážitok v rozsahu od 90 do 120 hertzov. Ďalším problémom je, že grafický čip musí byť dostatočne výkonný na to, aby vykreslil požadované množstvo snímok potrebnej kvality, a to na obidvoch displejoch VR headsetu. Preto sú niektoré headsety prepojené so stolným počítačom, aby boli schopné renderovať náročný grafický obsah pomocou výkonných grafických kariet [9]. Na vyriešenie tohto problému bola vyvinutá technika, ktorá znižuje grafické zaťaženie zariadenia pri renderovaní použitím hardvéru, ktorý sleduje pohyb očí používateľa na určenie bodu, do ktorého sa používateľ pozerá. Táto technológia potom vykreslí obrázok s vysokým rozlíšením v tomto bode a v jeho blízkosti, ale znižuje rozlíšenie obrázku ďalej od tohto bodu. Znížená kvalita obrazu je pre používateľa väčšinou nepovšimnuteľná, pretože periférne videnie človeka je významne menej citlivé ako bod do ktorého sa díva [10]. Displeje VR headsetov sú veľmi blízko očiam pozorovateľa, z dôvodu aby poskytli široké zorné pole. To má za následok, že kvôli šošovkám sa obraz síce bude zdať prirodzený, avšak kvôli faktoru priblíženia sa značne zviditeľnia medzery medzi jednotlivými pixelmi displeja. Preto sú potrebné displeje s veľmi vysokým rozlíšením. Šošovky môžu spôsobiť tiež skreslenie, no to sa dá zvyčajne opraviť v softvéri. Môžu byť tiež upravené tak, aby zodpovedali predpisu dioptrických okuliarov používateľa, takže aj používateľ zo zlým zrakom môže použiť headset a vidieť v ňom ostro [11].

2.2 Spôsoby detekcie pohybu

Sledovanie pohybu, teda proces digitalizácie pohybu určitého predmetu, je jedným z najdôležitejších aspektov virtuálnej reality. Pre dosiahnutie vysokej miery pohltenia musí byť používateľ schopný sa vo virtuálnom svete obzerať a pri naj-

pokročilejších systémoch sa aj vedieť pohybovať. Detekcia pohybu sleduje presnú polohu headsetov, ovládačov, iných predmetov alebo celého tela či jeho časti v priestore. Presná poloha je zaznamenávaná vďaka rozpoznávaniu rotácie a translačných, teda posuvných pohybov. To sa deje prostredníctvom rozličných senzorov alebo markerov. Senzory zaznamenávajú signál z reálnych objektov pri ich pohybe a prenášajú prijaté informácie ďalej na spracovanie. Vďaka tomuto spôsobu je možné sledovať, ako sa objekty ako hlava alebo ruky pohybujú v skutočnom svete, aby ich bolo možné reprezentovať virtuálne, čo je pre VR veľmi dôležité. Aj napriek tomu, že táto technológia sa stále vyvíja a nie je dostatočne implementovaná, má vo VR a AR veľmi veľké možnosti uplatnenia. Podľa typu senzorov sa systémy s takouto technológiou delia na tieto typy:

- optické metódy,
- bezdrôtové metódy,
- akustické metódy,
- magnetické metódy,
- inerciálne metódy,
- kombinácie viacerých sledovacích metód,
- a ostatné metódy.

Pri inteligentných telefónoch a ich headsetoch sú najpoužívanejšie optické a inerciálne metódy a ich kombinácie. Tieto druhy detekcie pohybu budú ďalej popisovať nasledujúce podkapitoly.

2.2.1 Optické metódy

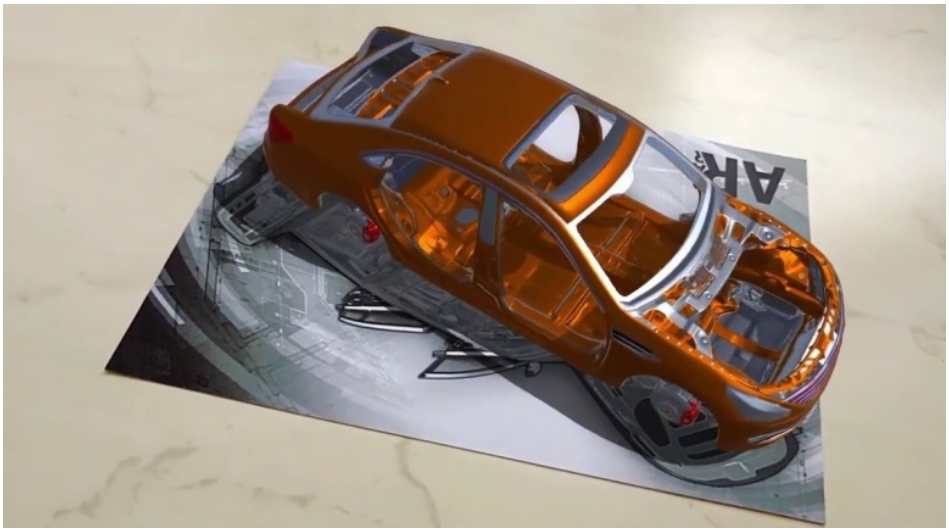
Optické metódy používajú kamery rozličných druhov na sledovanie pohybu tela [12]. Môžu to byť kamery schopné detekcie infračerveného alebo viditeľného svetelného spektra, hĺbkové kamery alebo stereofónne kamery, teda sústavy kamier. Optické sledovanie stereofónnymi kamerami je založené na podobnom princípe ako stereoskopické ľudské videnie. Tieto kamery sú schopné určiť vzdialenosť od sledovaného objektu a teda aj jeho polohu v priestore. Na to však potrebujú vedieť, ako ďaleko sú od seba vzdialené, preto je potrebná ich kalibrácia. Optické systémy

poskytujú niekoľko výhod. Sú relatívne spoľahlivé, široko dostupné a schopné sledovať polohu viacerých objektov súčasne. Takisto nie sú náchylné na okolité prostredie a majú malé odchýlky. Avšak pre čo najpresnejšie sledovanie je potrebná čo najlepšia kalibrácia. Rovnako je nevýhodou že systém vyžaduje priame svetelné spojenie medzi kamerami a sledovanými objektami bez kolízií, inak hrozí riziko získavania nepresných údajov. Optické metódy je možné deliť do dvoch kategórií, a to na systémy ktoré sledujú polohu objektov na základe sledovania značiek (z angl. "markers"), a na systémy ktoré nepotrebujú značky pre správne sledovanie polohy, nazvané markerless systémy. Týmto dvom typom optických metód sledovania polohy sa budú detailnejšie venovať nasledujúce dve časti tejto kapitoly.

Systémy sledovania na základe značiek

Pri týchto systémoch má sledovaný používateľ na svojom tele značky, ktoré tvoria známy vzor. Sú to väčšinou QR kódy, LED diódy alebo krúžky z vysoko reflexného materiálu na určitých známych miestach jeho tela alebo na headsete či ovládači [13]. Sústava kamier potom neustále hľadá tieto značky a pomocou nich vypočítava polohu sledovaného objektu. Tieto systémy sú väčšinou schopné aj dopočítavať údaje v prípadoch, keď je jedna alebo viac značiek mimo dohľadu kamier alebo sú dočasne zatienené. Komerčne sa používajú systémy, ktoré potrebujú relatívne málo značiek v porovnaní s profesionálnymi systémami, keďže sú požiadavky na sledovanie pohybu menšie. Jedným z problémov tejto metódy je riziko zámeny značiek, keď sa dve rôzne značky vyhodnotia ako jedna. Tento stav následne poskytuje nesprávne údaje a nepresné sledovanie pohybu. Tieto systémy sledovania polohy sa rozdeľujú do dvoch skupín, na systémy s pasívnymi a systémy s aktívnymi značkami. Systémy s aktívnymi značkami využívajú LED diódy kontrolované počítačom, ktoré poskytujú väčšiu presnosť a riešia niektoré nevýhody systémov s pasívnymi značkami. Tieto diódy môžu byť rozličných farieb alebo môžu rýchlo blikať v synchronizácií so snímacím systémom. Systémy s aktívnymi značkami sú presnejšie, ale prinášajú aj nevýhody zo strany používateľa. Ten buď musí nosiť zdroj energie alebo musí mať na sebe priviazaný tento systém diód, čo so sebou nesie určité nepohodlie. V súčasnosti sa systémy aktívnych značiek požívajú hlavne v profesionálnej sfére, napríklad v kinematografii alebo počítačových hrách, kde sa prostredníctvom optických metód vytvárajú počítačom generované obrázky. Zariadenia pre domáci trh disponujú naopak pasívnymi

značkami. Ďalšie možné delenie toho systému je sledovanie inside-out a outside-in. Pri sledovaní typu inside-out je kamera upevnená na sledovanom zariadení a značky sú umiestnené na statických miestach okolo objektu. Táto technológia sa používa napríklad pri sledovaní polohy značiek telefónom. Na obrázku 2.2 je značka umiestnená na stole, ktorú keď smartfón deteguje a zistí jej polohu, tak nad ňou vymodeluje karosériu auta. Naopak, pri sledovaní typu outside-in je sústava kamier umiestnená stacionárne a značky sú umiestnené na sledovanom zariadení. V tomto prípade kamery vypočítavajú polohu objektu prostredníctvom pohybujúcich sa značiek. Táto technológia sa zvyčajne používa v high-end zariadeniach.

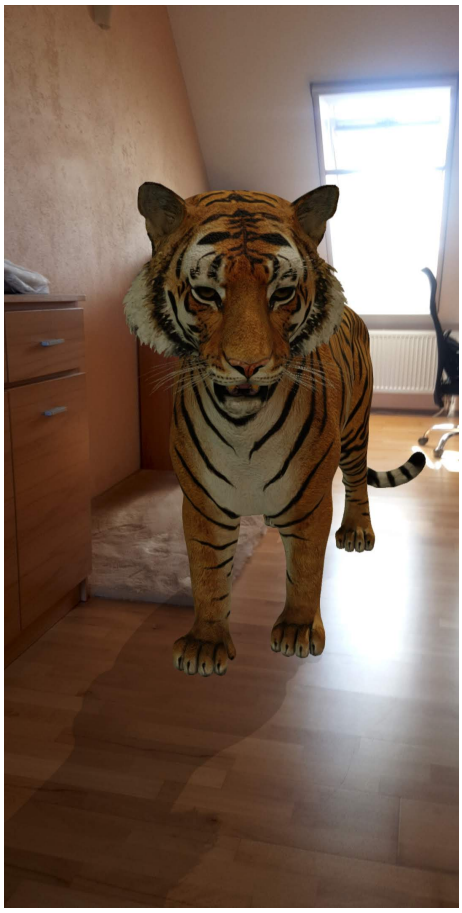


Obr. 2.2: Reprezentácia sledovania pohybu pomocou značky [14]

Systémy sledovania bez značiek

Zatiaľ čo systémy spomínané vyššie potrebovali špeciálne optické značky na určenie polohy, systémy sledovania bez značiek (z angl. markerless) ich nevyžadujú, čo ich robí oveľa flexibilnejšími v skutočnom nasadení. Pri tejto metóde napríklad nie je potrebné mať pripravené prostredie, v ktorom by boli vopred umiestnené značky na špecifických miestach. Na rozdiel od systémov sledovania založených na značkách tiež táto metóda umožňuje používateľovi voľný pohyb v miestnosti alebo inom prostredí a stále dostávať pozičnú spätnú väzbu, čím poskytuje širší rozsah uplatnenia [15]. Pri tejto metóde sa pri výpočte polohy a orientácie využívajú iba dáta získané z prirodzeného prostredia používateľa, kedy sa vykonáva spracovanie obrazu s cieľom zistiť jeho vlastnosti a tvary. Sledovanie bez

značiek je technológia ktorá sa vyvíja a uplatňuje hlavne v oblasti mobilnej AR a VR, avšak v súčasnosti vyžaduje kompromis medzi presnosťou a efektívnosťou, pretože čím viac informácií aplikácia zhromažďuje a využíva, tým je sledovanie presnejšie. Avšak s čím menej dátami výpočty aplikácie pracujú, tým je sledovanie efektívnejšie. Efektívnosť je tak pri mobilných zariadeniach problém, pretože dostupné zdroje sú relatívne obmedzené a sledovanie polohy ich nemôže využívať všetky, pretože zvyšok procesov tiež potrebuje procesorový výkon zariadenia [16]. Na obrázku 2.3 je znázornená aplikácia využívajúca systém sledovania bez značiek, ktorá na vhodné miesto zmapovaného priestoru vykreslí model tigra.



Obr. 2.3: Reprezentácia sledovania pohybu bez značiek

Aplikácia sleduje pomocou toku obrázkov z kamery zmenu polohy zariadenia, kedy následne upraví polohu virtuálneho objektu v priestore, budiac dojem že sa objekt v priestore skutočne nachádza. Najmodernejšie aplikácie už dokážu vykresľovať aj pohybujúce sa modely do priestoru, ktoré budia dojem že kráčajú napríklad po podlahe. Okrem kompromisu medzi presnosťou a efektívnosťou exis-

tuje niekoľko faktorov, ktoré ovplyvňujú výslednú kvalitu merania polohy sledovaním polohy bez značiek:

- **Tvar a textúra objektu:** Sledovanie je jednoduchšie a presnejšie, ak má objekt jedinečný tvar aj textúru.
- **Pozadie objektu:** Farba pozadia objektu je faktorom určujúcim kontrast medzi objektom a jeho prostredím. Sledovanie je presnejšie pri väčšom kontraste medzi prostredím a objektom.
- **Osvetlenie miestnosti:** Intenzita osvetlenia prostredia značne ovplyvňuje sledovanie polohy, keďže kamera musí správne zachytiť špecifické vlastnosti objektov a prostredia s čo najmenšou stratou detailov, pre čo najpresnejšie dáta na spracovanie.
- **Odraz svetla:** Keďže osvetlenie je dôležitým faktorom sledovania, odraz svetla, ako zrkadlá alebo sklenené povrchy, môžu brániť správnemu sledovaniu.
- **Typ a pozícia svetiel:** Určité typy svetelných zdrojov, ako napríklad žiarovka, tiež negatívne ovplyvňujú sledovanie polohy [17].

2.2.2 Inerciálne metódy

Inerciálne metódy používajú na sledovanie polohy údaje z akcelerometrov a gyroskopov. Akcelerometre sú schopné merať lineárne zrýchlenie. Deriváciou polohy podľa času je rýchlosť a deriváciou rýchlosti je zrýchlenie. Údaje z akcelerometra teda môžu byť integrované na výpočet rýchlosti a znova integrované na výpočet polohy, relatívne ku nejakému počiatočnému bodu. Gyroskopy sú zariadenia na meranie uhlovej rýchlosti. Uhlová rýchlosť sa rovnako môže integrovať na určenie uhlovej polohy vzhľadom ku počiatočnému bodu. Väčšina komerčných VR zariadení, ako headsety alebo ich ovládače používajú MEMS gyroskopy na sledovanie orientácie v troch rozmeroch v priestore s krátkym oneskorením a vysokou rýchlosťou aktualizácie. Presnejšiemu popisu týchto senzorov sa bude venovať nasledujúca podkapitola. Vďaka vývoju v automobilovom, leteckom či počítačovom priemysle a hlavne kvôli ich úspechu na trhu s inteligentnými mobilnými zariadeniami a tabletmi sa tieto senzory časom stali dostatočne malými, presnými a lacnými na to, aby mohli byť použité vo VR zariadeniach. Kombinácia týchto typov

senzorov v zariadení môže poskytnúť relatívne presné údaje o polohe s nízkou latenciou [18]. Tento spôsob môže byť kombinovaný s optickými metódami ako napríklad infračervené sledovanie alebo sledovanie prostredníctvom pasívnych značiek, aby sa dosiahlo robustné sledovanie pohybu, ale pre jednoduché, samostatné systémy menších rozmerov sú tieto senzory sami o sebe postačujúce. Avšak tieto senzory môžu byť časom nespoľahlivé, pretože ich odchýlka časom rastie, preto sa v niektorých VR systémoch zvyčajne používajú s inou formou sledovania polohy.

2.2.3 Kombinácie viacerých sledovacích metód

Pomocou kombinácie rôznych metód sledovania polohy sa dajú získať presnejšie výstupy. Jednou z možností je kombinácia inerciálneho a optického merania polohy. Optické sledovanie je použité ako hlavná metóda, avšak keď dôjde k chybe, ako je napríklad zatienenie sledovaného objektu alebo iná forma chyby neumožňujúca opticky zistiť polohu objektu, inerciálne sledovanie bude sledovať polohu, až kým to nebude znova schopný spraviť optický systém. Inerciálne sledovanie polohy má tiež nižšiu latenciu ako optické, takže sa tieto údaje môžu použiť počas čakania na aktualizáciu polohy cez optický systém. Optické sledovanie polohy taktiež pomáha stabilizovať odchýlku inerciálneho systému.

2.3 Senzory umožňujúce sledovanie pohybu telefónu

2.3.1 Gyroskop

Gyroskopický senzor je zariadenie, ktoré využíva zemskú príťažlivosť na sledovanie orientácie zariadenia. Existuje veľa druhov gyroskopov založených na rôznych operačných princípoch, avšak v elektronických zariadeniach, a hlavne vo VR headsetoch a telefónoch sa používajú vibračné MEMS gyroskopy [19]. Vibračné gyroskopy sú typom gyroskopov, ktoré používajú vibračnú štruktúru na určovanie rotácie. MEMS je skratka z aglického Micro-Electro-Mechanical Systems a označujú sa ňou elektromechanické konštrukcie a zariadenia veľmi malých rozmerov, často menej ako 1 milimeter. MEMS gyroskopy sú tak jediným typom gyroskopov s dostatočne malými rozmermi na to, aby mohli byť použité v zariadeniach virtuálnych realít. Tento typ gyroskopov sa vďaka svojim rozmerom a cene stal široko

dostupným. MEMS vibračné gyroskopy poskytujú buď analógový alebo digitálny výstup a merajú natočenie v troch na seba kolmých osiach. Okrem už spomínaných zariadení sa vibračné gyroskopy využívajú aj v automobilovom a zábavnom priemysle, priemyselnej robotike alebo v systémoch stabilizácie obrazu [20].

2.3.2 Akcelerometer

Akcelerometer je elektromechanický senzor schopný detekcie celkového zrýchlenia, respektíve preťaženia zariadenia, ktoré zvyčajne meria túto veličinu v metroch za sekundu za sekundu. Na rozdiel od súradnicového zrýchlenia berie celkové zrýchlenie v úvahu aj gravitačnú tiaž pôsobiacu na senzor, takže ak by bol senzor v pokoji, tak by na zemskom povrchu meral hodnotu preťaženia približne 1 g, alebo $9,81\ m/s^2$. Naopak, pri voľnom páde by tento senzor meral nulu vo všetkých osiach [21]. Akcelerometre snímajú buď statické zrýchlenie, ako napríklad gravitačnú silu, alebo dynamické zrýchlenie, ako vibrácie a pohyb. Okrem detekcie a smeru celkového zrýchlenia sa môžu použiť aj na sledovanie súradnicového zrýchlenia, vibrácií, nárazov či pádu. Prostredníctvom akcelerometra sú smartfóny a tablety schopné zistiť, či sú uchopené v portrait alebo landscape móde. Akcelerometre merajúce zrýchlenie v troch osiach sa stávajú čoraz viac používanejšími kvôli znižujúcim sa obstarávacím nákladom, avšak existujú aj tie merajúce zrýchlenie v jednej alebo dvoch osiach. Pri mikromechanických akcelerometroch použitých v telefónoch platí, že čím vyššie preťaženie sú schopné detegovať, tým sú menej citlivé [22].

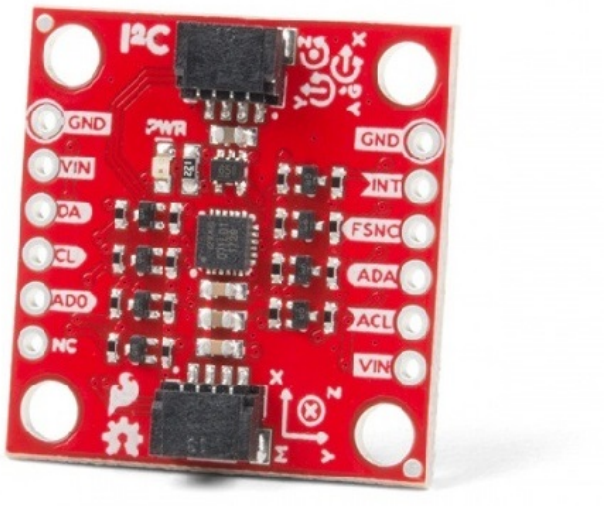
2.3.3 Magnetometer

Magnetometer je zariadenie schopné merať magnetizmus, teda silu, smer alebo relatívnu zmenu magnetického poľa. Jedným z takýchto zariadení je aj kompas, ktorý meria len smer okolitého magnetického poľa, najčastejšie magnetické pole Zeme. Existuje viacero druhov magnetometrov, no v spotrebiteľskej technike sa používajú vďaka svojím malým rozmerom, dostupnosti a relatívnej presnosti MEMS magnetometre. Väčšina inteligentných telefónov obsahuje tieto mikroelektromechanické senzory, ktoré slúžia hlavne ako kompas zariadenia. Pri týchto zariadeniach sa používajú magnetometre schopné merať magnetické polia v troch na seba kolmých osiach. Výhodou trojosích magnetometrov je, že sú schopné správne me-

rať magnetické pole bez ohľadu na to, ako je telefón v priestore natočený, alebo v akej nadmorskej výške sa nachádza. Magnetometre v mobilných zariadeniach už boli použité aj na bezdotykovú interakciu s týmito zariadeniami, kde boli prostredníctvom týchto senzorov sledované zmeny magnetického poľa v okolí telefónu na identifikáciu rôznych gest, kedy používatelia držali alebo na sebe mali magnety [23].

2.3.4 Inerciálna meracia jednotka

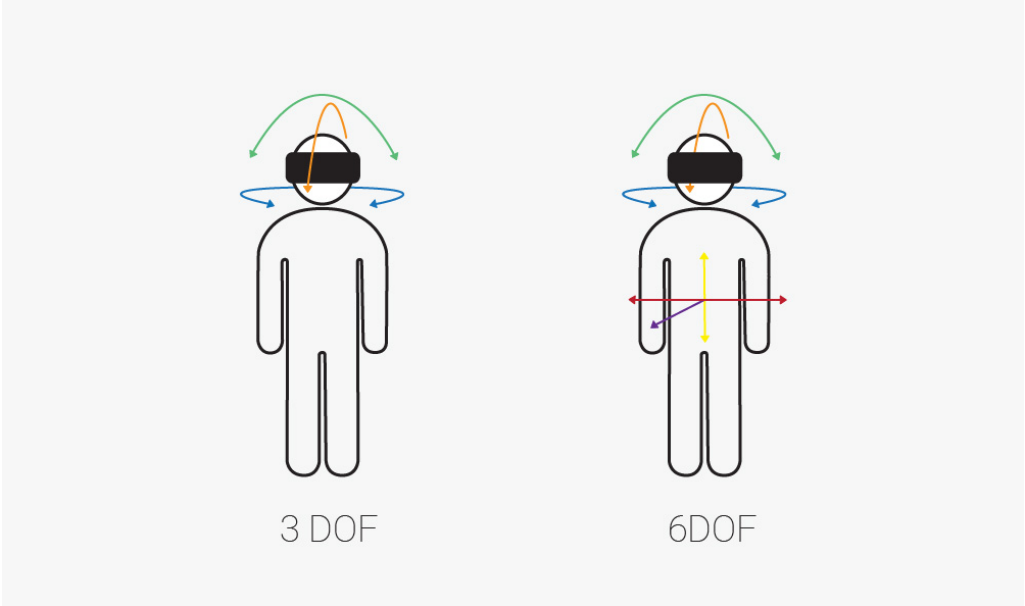
Inerciálna meracia jednotka (z angl. Inertial measurement unit), skrátene IMU, je elektronické zariadenie, ktoré je zložené z už spomínaného akcelerometra, gyroskopu a niekedy magnetometra. Pomocou týchto senzorov je zariadenie schopné merať veličiny ako zrýchlenie, orientácia, sily pôsobiace na toto zariadenie či veľkosť magnetického poľa. Inerciálne meracie jednotky sú použité v headsetoch virtuálnych realít a v mobilných zariadeniach na sledovanie ich rotácie. Na obrázku (Obr. 2.4) je zobrazené IMU zariadenie obsahujúce všetky tri senzory schopné merať zrýchlenie, orientáciu a magnetické pole zariadenia. Tieto zariadenia sú väčšinou zodpovedné za 3 stupne voľnosti v systémoch sledovania polohy.



Obr. 2.4: Inerciálna meracia jednotka [24]

2.4 Stupne voľnosti v systémoch virtuálnych realít

Stupne voľnosti (z angl. DoF - Degrees od Freedom) sú spôsoby, akými sa môže objekt pohybovať v priestore. V trojrozmernom priestore existuje dokopy šesť stupňov voľnosti. Tie je možné deliť do dvoch kategórií - na rotačné pohyby a na translačné, teda posuvné pohyby. Každá kategória má 3 stupne voľnosti. Rotačné pohyby možno definovať ako rotáciu okolo 3 na seba kolmých priamok prechádzajúcich stredom objektu. Rotačný pohyb sledujú inerciálne meracie jednotky headsetu, pozostávajúce z akcelerometra, gyroskopu a magnetometra. Tieto senzory sledujú zrýchlenie, orientáciu a gravitačné sily pôsobiace na headset na vypočítanie rotačného pohybu zariadenia. Na sledovanie troch translačných pohybov (vľavo/vpravo, hore/dole, dopredu/dozadu) sa pri inteligentných telefónoch zvyčajne používa externá kamera [25]. Na obrázku (Obr. 2.5) je možné vidieť rozdiel medzi VR headsetmi schopnými sledovať iba rotačné pohyby, teda 3 stupne voľnosti, a tými, čo umožňujú sledovanie aj posuvných pohybov, teda 6 stupňov voľnosti v priestore.



Obr. 2.5: Rozdiel medzi troma a šiestimi stupňami voľnosti [25]

3 Analýza aplikácie, jej súčastí a komponentov

Táto kapitola sa bude venovať rozboru požiadaviek práce a analýze súčasných riešení problematiky vývoja multi-platformového VR prostredia. Opíše a rozoberie všetky požiadavky časti aplikácie určenej pre inteligentné telefóny a takisto vysvetlí dôležité pojmy týkajúce sa týchto požiadaviek. Poskytne analýzu ako aj dôvody vybratia vývojového prostredia a softvérového rámca, ktoré budú použité pri vývoji VR aplikácie.

3.1 Analýza požiadaviek

Táto diplomová práca je súčasťou kolaborácie troch diplomových prác, ktorých hlavným cieľom je rozšíriť prostredie virtuálnej reality poskytujúce niekoľko dôležitých vlastností. Okrem tohto cieľa má práca viacero požiadaviek, ktorými sú:

- oboznámenie sa z možnosťami sledovania polohy prostredníctvom telefónov a ich využitia vo virtuálnych realitách,
- navrhnúť a implementovať multi-platformové kolaboratívne virtuálne prostredie pre inteligentné telefóny použitím webových technológií,
- a overiť použiteľnosť tohto prostredia a porovnať ho s podobnými existujúcimi riešeniami.

Nasledujúce podkapitoly sa budú venovať rozboru jednotlivých požiadaviek.

3.1.1 Oboznámenie sa z možnosťami sledovania polohy prostredníctvom telefónov

Prvou požiadavkou práce je oboznámiť sa s možnosťami moderných inteligentných zariadení a ich senzorov pri sledovaní polohy zariadenia alebo používateľa a analyzovať ich možnosti a využitie vo virtuálnej realite. Tejto problematike sa detailne venuje predchádzajúca kapitola, ktorá opisuje spôsoby sledovania polohy a orientácie zariadenia v reálnom čase, ktoré sa najčastejšie používajú v aplikáciách virtuálnych a rozšírených realít postavených na mobilnej platforme.

3.1.2 Návrh a implementácia multi-platformového kolaboratívneho virtuálneho prostredia pre inteligentné telefóny použitím webových technológií

Ďalšou požiadavkou je návrh a implementácia prostredia s troma vlastnosťami, musí byť multi-platformové, viac-používateľské a postavené na webových technológiách. Prvá vlastnosť znamená možnosť spustenia tohto prostredia na viacerých platformách poskytujúcich VR zážitok. Nevyhnutne musí byť podporované modernými inteligentnými telefónmi, v ideálnom prípade s čo najväčšou škálou smartfónov pre dosiahnutie čo najširšej kompatibility. Z tohoto dôvodu by mala byť pre prostredie použitá VR platforma pre mobilné telefóny poskytujúca širokú podporu zariadení. Príkladom takejto VR platformy je Google Cardboard, ktorý bol predstavený už v roku 2014 ako nízkonákladový systém určený pre podporu vývoja VR aplikácií a záujmu o mobilnú virtuálnu realitu. Najprv bol podporovaný operačným systémom Android, ale neskôr bol vďaka svojej popularite vyvinutý aj pre iOS a nakoniec aj pre web, prostredníctvom WebGL API a knižnice Three.js. Tento softvér tiež podporuje priestorový zvuk, teda simuláciu zvuku prichádzajúceho odkiaľkoľvek zo simulovaného prostredia [26]. Druhým príkladom VR platformy podporujúcej veľké množstvo zariadení je Google Daydream. Táto platforma vznikla po veľkom úspechu Cardboardu koncom roka 2016, ako jeho zdokonalený nástupca. V porovnaní s Cardboardom, ktorý ponúkal len obmedzenú funkcionalitu, bol Daydream súčasťou samotného operačného systému a obsahoval zdokonalenú funkcionalitu, ako napríklad podporu VR ovládačov alebo nový algoritmus sledovania hlavy používateľa, ktorý kombi-

noval vstup z rozličných senzorov zariadenia. Tento softvér bol dostupný len vybraným telefónom so systémom Android, ktoré spĺňali softvérové aj hardvérové požiadavky tejto platformy. Daydream však nemal u spotrebiteľov a vývojárov taký veľký úspech ako jeho predchodca a preto bola koncom roka 2019 ukončená jeho podpora s tým, že pre túto VR platformu už nebudú certifikované žiadne nové zariadenia. Ďalšou nevýhodou tohto systému je, že nepodporuje zariadenia s operačným systémom iOS, iba Android 7 a vyššie. Avšak od jeho predstavenia bolo certifikovaných veľa výkonných smartfónov a vzhľadom na jeho výhody oproti Cardboardu je vhodným adeptom pre využitie v kolaboratívnej realite. Okrem smartfónov musí toto prostredie podporovať aj platformu HoloLens. HoloLens sú samostatné okuliare pre zmiešanú realitu vyvinuté spoločnosťou Microsoft, ktoré boli predstavené v roku 2016. Tieto okuliare boli špeciálne vyvinuté pre zmiešanú realitu, kvôli čomu obsahujú senzory ako inerciálnu meraciu jednotku (obsahujúcu kompas, akcelerometer a gyroskop), hĺbkovú kameru, fotoaparát, sústavu mikrofónov a snímač okolitého svetla. Takisto poskytujú koprocesor vyvinutý špecificky pre toto zariadenie ktorý pomáha spracovať údaje so senzorov, mapovanie prostredia, rozpoznávanie gest alebo rozpoznávanie hlasu a reči. Zariadenie sa ovláda gestami rúk a prstov, ktoré sleduje v reálnom čase [27]. V roku 2019 bola predstavená druhá generácia okuliarov s označením HoloLens 2, ktoré oproti predchodcovi priniesli viacero vylepšení, ako ergonomickejší a pohodlnejší dizajn, značne vyšší výkon, vylepšené sledovanie očí a priestorový zvuk alebo detailné sledovanie rúk. Obidve generácie okuliarov používajú operačný systém z platformy Windows 10 [28]. Kolaboratívne prostredie by mohlo okrem týchto platforiem voliteľne podporovať aj iné, ako sú HTC Vive, Oculus Rift alebo Quest, Samsung GearVR alebo desktopové počítače. Čím viac platforiem by bolo podporovaných, tým by bolo prostredie dostupnejšie a univerzálnejšie. Táto diplomová práca je súčasťou kolaborácie troch prác, a venuje sa vývoju prostredia pre inteligentné telefóny. Vývoj prostredia pre headset HoloLens je riešený v druhej z týchto diplomových prác [29]. Bližšie informácie poskytne kapitola Návrh riešenia.

Ďalšou vlastnosťou prostredia je už spomínaná kolaborácia, teda spolupráca. V tomto kontexte sa pod týmto pojmom rozumie možnosť pripojenia sa viacerých používateľov naraz do prostredia. To znamená, že centrálny uzol, teda server, by nevytváral inštanciu nového prostredia zvlášť pre každého používateľa, ale na-

opak, pripojil by každého novo prihláseného používateľa do jedného zdieľaného sveta. Pripojení používatelia by tak mali možnosť vnímať ostatných používateľov v podobe ich avatarov a vedeli by sledovať ich pohyb a do určitej miery sa podieľať na vzájomnej interakcii. Pripojenie sa používateľa do virtuálneho prostredia by bolo možné realizovať napríklad objavením sa avatara daného používateľa na statickom alebo náhodnom mieste v simulovanom svete a odpojenie sa by bolo uskutočnené jeho zmiznutím zo scény.

Treťou vlastnosťou kolaboratívneho multi-platformového prostredia je jeho vývoj použitím webových technológií. Výhodou tohto prístupu je absencia nutnosti inštalácie vlastnej aplikácie VR prostredia. Keďže pôjde o internetovú aplikáciu, používateľovi stačí otvoriť jej stránku prostredníctvom webového prehliadača, ktorý má väčšina platforiem poskytujúca VR zážitok už predinštalovaný. Tento prístup môže mať však aj nevýhody, akými sú potencionálne obmedzenia vývojového prostredia alebo frameworku. Tie sú väčšinou väčšie ako pri samostatných aplikáciách a môže sa to prejaviť napríklad na horšej úrovni grafických detailov, zvuku nižšej kvality, obmedzenej funkcionalite alebo príliš oneskorenej spätnej väzbe spôsobenej nedostatočne rýchlym internetovým pripojením. Avšak po analýze dostupných riešení a pri použití najviac vyhovujúcich technológií by sa tieto možné nedostatky webových aplikácií mohli značne minimalizovať.

Okrem týchto troch esenciálnych vlastností by prostredie malo disponovať jednoduchým používateľským rozhraním, ktoré by bolo intuitívne pre používateľa. Ak by používateľ otočil hlavou alebo sa posunul v určitom smere v skutočnom svete, mal by sa tento pohyb čo najvernejšie preniesť do syntetického prostredia, teda používateľov avatar by mal pohnúť hlavou v čo najpresnejšom uhle alebo sa posunúť o čo najpresnejšiu vzdialenosť vzhľadom na virtuálny svet v porovnaní so skutočnosťou. Rozhranie aplikácie tiež musí disponovať všetkou očakávanou funkcionalitou a tiež by malo poskytovať rýchlu spätnú väzbu alebo používateľovi známe alebo zaužívané gestá. Celé používateľské prostredie by malo byť responzívne a prispôsobené displejom rôznych veľkostí, keďže samotná aplikácia bude spustiteľná na viacerých druhoch platforiem, ako sú smartfóny rozličných veľkostí, VR okuliare HoloLens a potencionálne ďalšie systémy.

Rôzne podporované platformy by takisto mali ponúkať čo najpodobnejšie vlastnosti a funkcie a to z dôvodu, aby používateľov rôzne platformy nijako neobmedzovali. Takže ak by napríklad HoloLens ponúkal otáčanie hlavy avatara vo VR

svete otáčaním hlavy používateľa pomocou senzorov, alebo zmenu pohybu pomocou mapovania reálneho prostredia a sledovania zmeny polohy headsetu, tak by tieto funkcie mali podporovať aj smartfóny. Samozrejme, každá platforma môže podporovať určité vlastnosti len v rámci svojich možností. Ak smarfón nebude disponovať napríklad gyroskopom, tak nebude schopný sledovať svoje natočenie a podľa toho prispôsobiť avatara. Týmito senzormi nebudú disponovať hlavne desktopové počítače, ktoré budú musieť byť tiež staticky na jednom mieste, takže v tomto prípade sa bude pohyb riešiť napríklad myškou a klávesnicou. Z týchto dôvodov bude potrebné vybrať také vývojové prostredie a technológie, ktoré budú brať ohľad na rozdiely všetkých platforiem a prispôsobia tomu všetky funkcie a zároveň budú poskytovať čo najpodobnejší herný zážitok medzi týmito platformami.

3.1.3 Overenie použiteľnosti prostredia a jeho porovnanie s podobnými existujúcimi riešeniami

Poslednou požiadavkou je overiť použiteľnosť virtuálneho prostredia z hľadiska faktorov ako funkčnosť, plynulosť, presnosť sledovania pohybu alebo dĺžka latencie. Tomuto rozboru sa bude venovať predposledná kapitola tejto práce, ktorá otestuje a ohodnotí výsledné prostredie. Veľa spomínaných faktorov, ako je presnosť sledovania pohybu, odozva na akciu používateľa alebo stabilita celkovej aplikácie je v značnej miere závislá na použitých technológiách, ako sú programovací jazyk alebo framework, na ktorom bude postavená. Preto je nevyhnuté použiť také technológie, ktoré najlepšie vyhovejú týmto faktorom. Nasledujúca podkapitola sa bude venovať analýze softvérového rámca, ktorý bude použitý pri vývoji aplikácie.

3.2 A-Frame

A-Frame je open-source webový softvérový rámec určený pre vývoj aplikácií a hier vo virtuálnej a zmiešanej realite, zo začiatku vyvíjaný spoločnosťou Mozilla a neskôr Googlom. Je to komponentový softvérový rámec založený na Three.js a WebVR, v ktorom môžu developeri vyvíjať 3D a VR scény prostredníctvom HTML jazyka a Javascriptu [30]. Bol vytvorený na urýchlenie vývoja softvéru vo WebVR,

a poskytol webovým developerom platformu na vytváranie VR prostredí. Vďaka webovej aplikácií nie je potreba žiadnej inštalácie, prostredie je spustiteľné vo webovom prehliadači. A-frame funguje na veľkom počte VR headsetov, mobilných zariadeniach a aj desktopoch. Medzi jeho výhody patrí multiplatformová podpora, vysoký výkon v široko používaných prehliadačoch ako Firefox a Chrome, podpora viacerých VR ovládačov, jednoduchý vývoj, real-time online podpora či dostupná dokumentácia. Keďže je založený na jazykoch HTML a Javascript, tak ho podporuje väčšina online vývojových prostredí určených na vývoj webových aplikácií, ako napríklad Glitch. A-frame je založený na architektúre Entity-Component-System, čo znamená že každý objekt v scéne je entita a každá entita sa skladá z jedného alebo viacerých komponentov určujúcich jej správanie alebo funkciu. Entita bez akýchkoľvek komponentov sa nevykreslí, lebo nemá definovanú vizuálnu reprezentáciu ani správanie. Týmto spôsobom je možné meniť správanie alebo vzhľad entít za behu programu prostredníctvom pridávania alebo odoberania komponentov. Tento architektonický vzor je často používaný pri vývoji počítačových hier a odstraňuje problémy hierarchií a dedičnosti, ktoré býva obtiažne pochopiť, udržiavať či rozširovať [31]. A-frame obsahuje niekoľko primitív, čo sú vlastne entity s preddefinovaným vzhľadom alebo funkcionalitou. Primitíva, ale aj všeobecné entity môžu byť pridané do scény prostredníctvom HTML súboru, kde ich syntax pripomína základné HTML elementy.

```
<a-scene>
    <a-sphere position="0 1.25 -5" rotation="0 45 0" radius="1.25"
        color="#EF2D5E"></a-sphere>
    <a-plane position="0 0 0" rotation="-90 0 0" width="4" height="4"
        color="#7BC8A4"></a-plane>
    <a-sky color="#ECECEC"></a-sky>
</a-scene>
```

Výpis 3.1: Kód jednoduchej A-frame scény

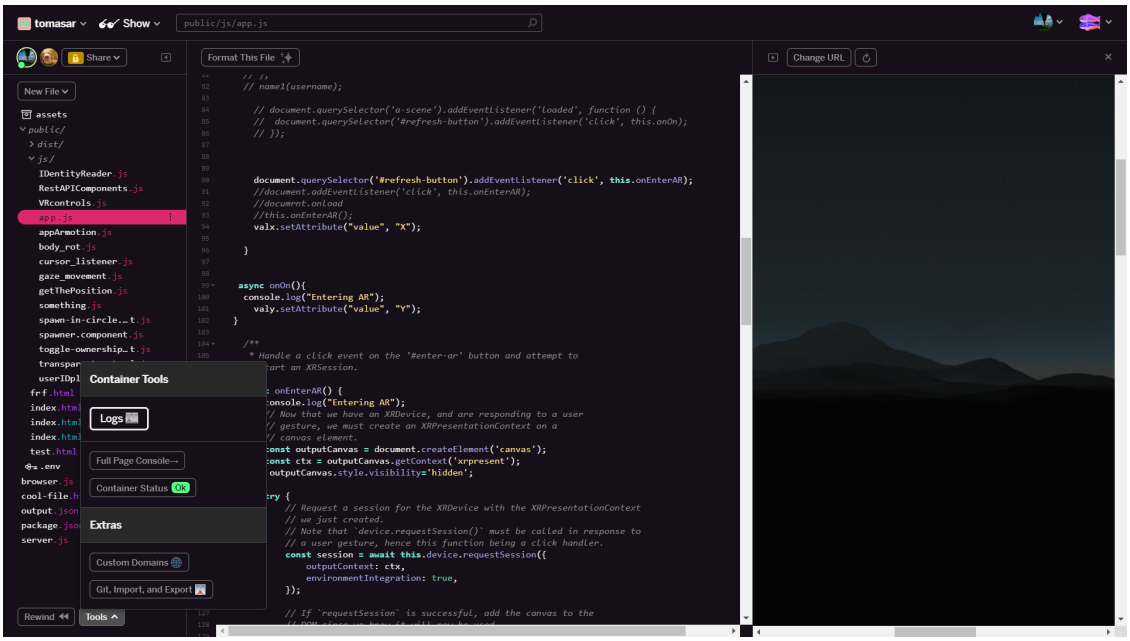
Kód vypísaný vyššie pozostáva z niektorých základných entít obsahujúcich niekoľko komponentov. Najdôležitejšou entitou je HTML element <*a-scene*> , ktorý definuje virtuálnu scénu a sú doňho vkladané všetky ostatné entity. Okrem neho scénu tvoria tri primitíva, ktoré majú dodefinované určité komponenty, ako napríklad pozíciu v troj-rozmernej karteziánskej súradnicovej sústave scény, rotáciu

v príslušných osiach tejto súradnicovej sústavy, polomer v jednotkách súradnicovej sústavy, farbu, šírku a výšku. Táto scéna sa vizuálne podobá na testovaciu scénu opísanú v nasledujúcej kapitole. Softvérový rámec A-Frame bol vybratý, pretože najviac vyhovuje požiadavkám aplikácie, ako sú webové VR prostredie, ktoré je vyvinuté a prispôsobené pre viacero platforiem. Networked-Aframe je rámec podporujúci zdieľanie HTML entít a ich atribútov, ktorý umožňuje vývoj viac-používateľských prostredí pomocou synchronizácie entít a komponentov A-Frame scény. Takisto je voľne dostupný a ponúka relatívne ľahkú implementáciu a vytvorenie prostredia. Z hľadiska smartfónov však A-Frame počas písania tejto práce ponúka iba sledovanie polohy smartfónu prostredníctvom značiek. To je relatívne veľká limitácia, pretože by to znamenalo, že by táto funkcionalita bola k dispozícií len v určitých striktných podmienkach, kedy by kamera sledovala špeciálnu značku. To by značne obmedzovalo pohyb hráča vo fyzickom prostredí, ako aj jeho otáčanie. Kvôli tomu bude experimentálne overená funkcionalita sledovania polohy zariadenia cez jeho polohové senzory a kameru prostredníctvom rôznych technológií. Následne bude snaha implementovať do aplikácie technológiu poskytujúcu čo najpresnejšie výsledky. Nasledujúca kapitola sa bude venovať analýze kvality sledovania polohy telefónu cez senzory a kameru.

3.3 Glitch

Glitch ako webové prostredie vznikol v roku 2017 pod rovnomennou spoločnosťou, kde si používatelia mohli vytvárať webové aplikácie pomocou Javascriptu a Node.js. Glitch ponúka niekoľko výhod a funkcií, kvôli ktorým bol zvolený ako prostredie pre vývoj aplikácie. Jednou z hlavných výhod je, že je to online editor s automatizovaným nasadením, čo znamená že poskytuje server pre vyvíjanú aplikáciu, takže nie je nutné inštalovať vlastné IDE ani lokálny server. Ďalšou výhodou je okamžité prejavenie sa zmien v kóde vo výslednej aplikácií bez nutnosti reštartu serveru. Okno s aplikáciu sa taktiež automaticky obnoví pri detekcii zmien v kóde. Keďže je aplikácia a jej zdrojový kód uložená v cloude, vývojár má možnosť po prihlásení sa pristupovať ku všetkým svojím projektom z akéhokoľvek počítača. Glitch takisto podporuje úpravu kódu viacerým používateľom súčasne a každý vidí zmeny v kóde ostatných vývojárov hneď ako boli vykonané. Vďaka tomu umožňuje simultánny vývoj bez nutnosti vytvárania alternatívnych vetiev a

ich následného zlučovania. Okrem toho má aj podporu Git-u, teda systému riadenia verzií a teda je možné sledovať zmeny a úpravy, či sa vrátiť na predchádzajúcu verziu aplikácie. Toto IDE ponúka podporu od jeho komunity, má vlastné fórum a od jeho spustenia ho používa viac ako milión ľudí pre vývoj webových aplikácií [32]. Na obrázku 3.1 je znázornené toto IDE, v ktorom je otvorený projekt obsahujúci VR scénu. Úplne naľavo je vidieť hierarchickú štruktúru projektu so všetkými súbormi a priečinkami, ktoré obsahuje. V strede sa nachádza textový editor, podobne ako u väčšiny vývojových prostredí. Výhodou je, že editor ukazuje na akom riadku sa nachádza iný používateľ a čo upravuje v prípade, že má otvorený ten istý súbor. Naľavo hore je tlačidlo spustenia aplikácie, ktoré ponúka spustiť aplikáciu v novom okne prehliadača alebo vedľa prostredia, ako je to znázornené na pravej strane obrázka. Sú tu takisto možnosti vytvorenia nového projektu, zmazania projektu, zmeny projektu či nastavenia editora. Dole na ľavej strane sú nástroje ako import alebo export projektu, možnosť otvorenia terminálu a iné.



Obr. 3.1: Webové vývojové prostredie Glitch

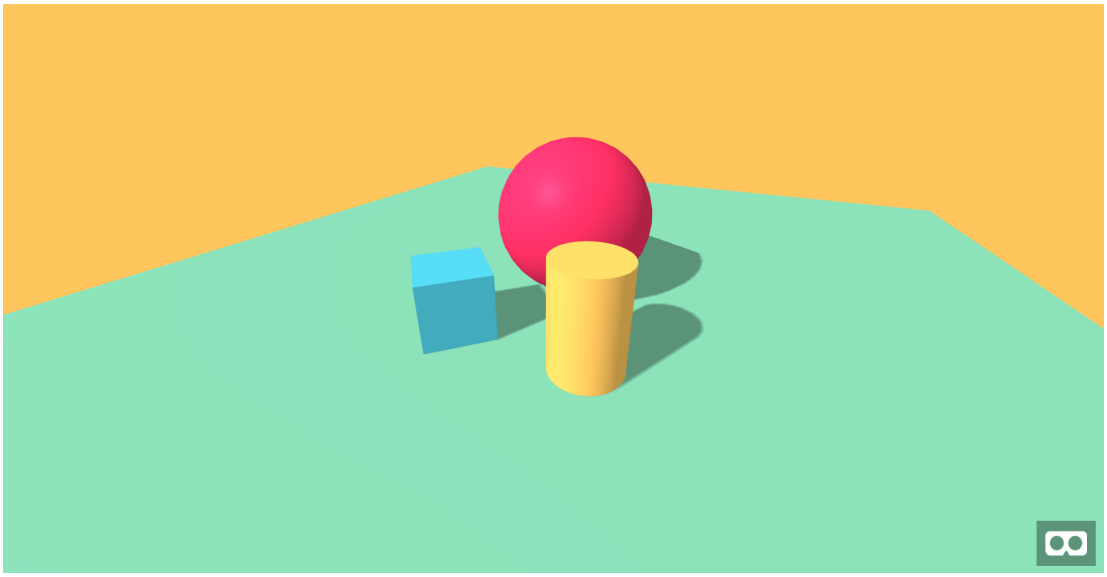
Toto prostredie bolo vybraté pre vývoj prostredia LIRKIS C-GVE a jeho rozšírení kvôli jeho viacerým výhodám spomenutým vyššie a takisto kvôli podpore softvérového rámca A-Frame.

4 Experimentálna analýza kvality sledovania polohy zariadenia prostredníctvom polohových senzorov a fotoaparátu

Ako bolo spomínané v predchádzajúcej kapitole, A-Frame nepodporuje sledovanie zariadenia bez značiek a preto bude musieť byť táto technológia vytvorená a implementovaná do scény. Táto kapitola sa z tohto dôvodu bude venovať experimentálnej analýze sledovania fyzickej polohy smartfónu v priestore. Prvá podkapitola opíše sledovanie cez pohybové senzory zariadenia, zatiaľ čo druhá kapitola sa bude venovať sledovaniu prostredníctvom kamery. Na záver sa tieto metódy porovnajú a vyberie sa najlepší adept pre implementáciu v samotnej aplikácií. Na testovanie bola vytvorená virtuálna scéna frameworkom A-frame vo webovom vývojovom prostredí Glitch, ktorá obsahovala hráča v podobe kamery a zopár jednoduchých predmetov v jeho blízkosti. Keďže A-frame je framework napísaný v jazyku Javaskript, budú aj experimentálne skripty napísané v tomto jazyku. Ďalším dôvodom použitia tohto jazyka je vysoká kompatibilita s rôznymi webovými prehliadačmi a jednoduché pristupovanie k údajom zo senzorov. Skripty na zisťovanie polohy budú potom do tejto scény implementované a pomocou nich bude menená pozícia kamery v scéne. Podrobnejšiemu vysvetleniu implementácie, tvorbe scény a skriptov sledovania polohy vo frameworku A-frame sa bude venovať šiesta kapitola „Implementácia". Táto kapitola sa síce venuje implementácií rôznych skriptov, avšak len z dôvodu analýzy rôznych techník a výberu tých najvhodnejších, ktoré budú implementované do samotnej aplikácie.

4.1 Testovacia scéna

Na obrázku (Obr. 4.1) je znázornená VR scéna použitá pri testovaní skriptov. Obsahuje len 3 objekty okrem podlahy a kamery, a to z dôvodu, aby sa prostredníctvom nich dala sledovať zmena pozície kamery voči scéne, teda týmto objektom. Údaje zo senzorov o fyzickej zmene polohy smartfónu budú priamo alebo nepriamo späté s pozíciou kamery, takže pri detekcii zmeny polohy by mal byť tento efekt viditeľný v scéne v reálnom čase.



Obr. 4.1: Ukážka scény použitej pri testovaní

4.2 Overenie sledovania polohy zariadenia prostredníctvom polohových senzorov

Prvým spôsobom sledovania polohy smartfónu je cez spomínané polohové senzory ako gyroskop, akcelerometer alebo kompas. K prístupu k dátam o polohe či orientácií zariadenia sa dá vo webovom prostredí prostredníctvom jazyka Javascript dostať viacerými spôsobmi. Prvou možnosťou je prístup pomocou Javascriptom natívne podporovaných udalostí „devicemotion“ a „deviceorientation“ a následného pridania listenera na tieto udalosti, alebo pomocou knižníc tretích strán, ktoré zväčša ponúkajú väčší počet funkcií a možností, alebo sa softvérovo

snažia tieto údaje spresniť pomocou rôznych techník. Jednou z takýchto knižníc je motion-sensors.js, ktorá implementuje Generic Sensor API. Toto nové API slúži na pristupovanie k dátam z rozličných senzorov a je novodobou alternatívou pre udalosti deviceorientation a devicemotion. Knižnica implementuje rôzne rozhrania ako Accelerometer, LinearAccelerationSensor, GravitySensor a iné [33], ktorých funkcionalita bude testovaná. Postupne bolo vytvorených a implementovaných do scény viacero skriptov pristupujúcim k dátam buď pomocou spomínaných natívnych funkcií alebo pomocou objektov a metód tejto knižnice. Tieto skripty riešia sledovanie polohy cez vyššie uvedené senzory každý iným spôsobom, pričom niektoré skripty sa líšia len menšími zmenami. Skripty boli testované a implementované do scény samostatne a cieľom bolo ich porovnávať a zistiť ktorý ponúka najlepšie výsledky, a teda je schopný najpresnejšie zisťovať zmeny polohy. Všetky skripty však v scéne navonok budú vykonávať rovnakú funkciu a tou je vertikálne posúvanie kamery scény na základe vertikálnej zmeny polohy zariadenia. V prípade, ak sa nájde skript alebo skripty ktoré budú túto zmenu sledovať v dostatočnej kvalite, bude vytvorená funkcionalita sledovania polohy vo zvyšných dvoch osiach, kedy sa aj táto zmena porovná medzi skriptami. Všetky skripty vytvorené v tejto kapitole budú dostupné na priloženom CD médiu v priečinku *CD/usr/*. Fungovanie jednotlivých skriptov je vysvetlené v nasledujúcich podkapitolách, pričom sa z hľadiska získavania dát, ich spracovania a následnej zmeny polohy kamery rozdeľujú na tieto:

- prístup k polohe prostredníctvom natívnych funkcií a listenerov,
- prístup k polohe cez natívne funkcie a výpočet približnej odchýlky šumu,
- prístup k polohe cez objekt Accelerometer z knižnice motion-sensors.js,
- prístup k polohe cez objekt LinearAccelerationSensor z knižnice motion-sensors.js,
- prístup k polohe cez objekt GravitySensor z knižnice motion-sensors.js,
- prístup k polohe cez objekt LinearAccelerationSensor z knižnice motion-sensors.js a vypočítanie približnej odchýlky šumu,
- a prístup k polohe cez objekt LinearAccelerationSensor z knižnice motion-sensors.js a aplikovanie filtrov pre potlačenie šumu.

4.2.1 Pristupovanie k polohe cez natívne funkcie

Sledovanie zmeny orientácie a zmeny polohy zariadenia je možné zavolaním týchto funkcií:

```
window.addEventListener("deviceorientation", handleOrientation, true);
window.addEventListener("devicemotion", handleMotion, false);
```

Výpis 4.1: Funkcie pre sledovanie sklonu a pozície

pričom „handleOrientation“ je funkcia, ktorá sa zavolá pri každej zmene orientácie zariadenia a funkcia „handleMotion“ je volaná pri každej detekcii zmeny polohy. Zmena orientácie a polohy je sledovaná vo všetkých podporovaných osiach a preto sú tieto funkcie volané pri akejkoľvek zmene v akejkoľvek sledovanej osi. Ak však zariadenie nedisponuje polohovými senzormi, nie sú funkcie volané nikdy a tým pádom sa poloha kamery v scéne nezmení. Latencia senzorov je minimálna, až nepostrehnuteľná. Sledovanie orientácie je vo frameworku A-frame predvolene podporované, kedy vie používateľ natívne točiť kamerou pri natáčaní telefónu. Sledovanie polohy akcelerometrom je nutné doimplementovať a potom je jednoducho možné napríklad meniť polohu objektu pohybom telefónu. V tomto skripte sa za vertikálnu súradnicu kamery priamo dosadzuje údaj z akcelerometra rovnakej osi.

```
function motion(event){
    let pos = camera.getAttribute("position");
    //pos.x += event.acceleration.x.toFixed(1)/100;
    pos.y += event.acceleration.y.toFixed(1)/20;
    //pos.z += event.acceleration.z.toFixed(1)/100;

    camera.setAttribute("position", pos);
}
```

Výpis 4.2: Kód zmeny pozície kamery podľa polohy zariadenia

Pri každom zavolaní tejto funkcie sa získa pozícia kamery, a zmení sa hodnota príslušnej osi. Výsledky tohto a ostatných prístupov ako aj ich porovnanie bude zhrnuté v poslednej sekcii tejto podkapitoly.

4.2.2 Pristupovanie k polohe cez natívne funkcie a výpočet šumu

Tento skript vytvára nový komponent vo frameworku nazvaný „gyroacc“, ktorý tiež pristupuje k polohe cez udalosť devicemotion. Avšak skript sa narozdiel od predchádzajúcej implementácie na začiatku snaží eliminovať šum z akcelerometra spôsobom, že prvé dve sekundy meria hodnoty y-novej súradnice z akcelerometra, ktoré potom spriemeruje vydelením počtom nameraných hodnôt, aby sa zistila približná odchýlka. Požívateľ by nemal hýbať telefónom počas tohto merania aby bol výsledok čo najpresnejší. Skript následne už len odčítava túto odchýlku od aktuálnych hodnôt, čím by sa mali dostávať výsledky veľmi blízke nule.

```
movement: function () {
  if (counter != MAX_COUNT){
    counter++;
    avgBias += event.acceleration.y;
    if (counter == MAX_COUNT) avgBias /= MAX_COUNT;
    return;
  }
  velocity += event.acceleration.y - avgBias;
```

Výpis 4.3: Kód na výpočet priemerného šumu

Pri plynulom pohybe, teda ak sa dá zrýchlenie, rýchlosť a vzdialenosť definovať ako spojitá funkcia závislá na čase, je okamžité zrýchlenie rovné derivácií rýchlosti. Pri známom zrýchlení je teda možné vypočítať rýchlosť ako jeho integrál. Avšak akcelerometer nesleduje zrýchlenie v nekonečne malých intervaloch, a preto je rýchlosť daná súčtom všetkých nameraných zrýchlení vynásobených časovým rozdielom medzi jednotlivými meraniami. Podobne je celková vzdialenosť rovná súčtu všetkých rýchlostí vynásobených týmto časovým rozdielom. Keďže prvotné veličiny by mali byť nulové kvôli používateľovi, čo držal telefón v statickej pozícií, netreba brať do úvahy žiadne počiatočné konštanty. Skript teda týmto spôsobom mení polohu kamery vo vertikálnom smere. Aj keď je tento spôsob matematicky správny, úroveň odchýlky sa extrémne zväčšuje. Pretože malá odchýlka v zrýchlení spôsobí relatívne veľkú odchýlku v rýchlosti a podobne, malá odchýlka v rýchlosti spôsobí relatívne veľkú odchýlku vo vzdialenosti. Vďaka tomu môže mať relatívne malý šum z hodnôt akcelerometra veľký dopad na výslednú vzdialenosť.

4.2.3 Pristupovanie k polohe cez objekt Accelerometer z knižnice motion-sensors.js

Jednoduchý skript, ktorý takisto vytvára nový komponent v A-frame nazvaný „accelerometer“ a pristupuje k polohovým údajom prostredníctvom objektu typu Accelerometer tým, že sa vytvorí jeho inštancia, zavolá sa funkcia start() tohto objektu a následne sa funkcii onreading priradí metóda, ktorá nastavuje pozíciu kamery na údaj, ktorý dostala pri zavolaní.

```
AFRAME.registerComponent('acc',{
update: function () {
  let accelerometer = new motion.Accelerometer({frequency: 60});
  this.movement = this.movement.bind(this);
 accelerometer.onreading = this.movement;
 accelerometer.start();
```

Výpis 4.4: Kód vytvorenia komponentu vo frameworku A-Frame

Skript však nefunguje pretože telo funkcie registerComponent frameworku Aframe slúžiace na vytvorenie nového komponentu do scény, konkrétne jeho základné udalosti update a init, sa nespustia ak je v tomto skripte importovaný iný skript, v tomto prípade motion-sensors.js. Bez importu by nebolo možné vytvoriť objekt Accelerometer, čím by skript stratil význam. Preto sa pri importovaní iných skriptov nebudú vytvárať nové komponenty do frameworku, ale ku A-frame entitám sa bude pristupovať cez ich jedinečné ID volaním funkcie *document.getElementById*.

4.2.4 Pristupovanie k polohe cez objekt LinearAccelerationSensor z knižnice motion-sensors.js

Tento skript sa od predchádzajúceho líši v dvoch veciach, a to že nevytvára nový komponent vo frameworku a získava údaje o polohe smartfónu cez objekt typu LinearAccelerationSensor. Použitím objektu toho typu sa získajú údaje zrýchlenia bez vplyvu zemskej tiaže na telefón, ktorá je automaticky na pozadí odpočítaná z nameraných hodnôt pomocou gyroskopu alebo magnetometra. Pri vytváraní inštancie LinearAccelerationSensor sa poskytujú konštruktoru 2 argumenty, prvým je „frequency“, teda frekvencia získavania nových údajov z akcelerometra, čo je ar-

gumentom konštruktorov aj iných typov senzorov a druhým je „referenceFrame“, teda referenčný rámec. Ten môže mať 2 hodnoty, a to screen a device. Device značí súradnicový systém zariadenia a screen zas súradnicový systém obrazovky zariadenia. Obidva sú to trojrozmerné karteziánske súradnicové systémy, rozdiel je že súradnicový systém zariadenia zostáva statický vzhľadom na samotné zariadenie, zatiaľ čo súradnicový systém obrazovky mení svoje x-ové a y-ové súradnice podľa natočenia obrazovky. Teda y-ová súradnica mieri vždy smerom hore, bez ohľadu na natočenie zariadenia. Takže pri nesprávnom argumente sa môže stať, že pri otočení zariadenia do landscape módu bude skript čítať údaje z nesprávnej osi zariadenia. Tento skript meria y-ovú súradnicu v oboch natočeniach smartfónu.

4.2.5 Pristupovanie k polohe cez objekt GravitySensor z knižnice motion-sensors.js

V tomto skripte sa sleduje poloha pomocou objektu GravitySensor, čím sa získavajú hodnoty so senzorov vrátane zemskej tiaže. Gravitácia teda nie je vôbec filtrovaná a týmto spôsobom je skript schopný merať zmenu polohy len jedným smerom – kolmým na zemský povrch, teda rovnobežným k zemskej tiaži. Pomocou vzorca

```
gravity = ((Math.sqrt((sensor.x * sensor.x) + (sensor.y * sensor.y) +
    (sensor.z * sensor.z))) - avgBias);
```

Výpis 4.5: Vzorec na výpočet absolútneho zrýchlenia

bude do premennej gravity dosadená veľkosť absolútneho aktuálneho zrýchlenia bez informácie o jeho smere. Tento skript, rovnako ako v poradí druhý skript, tiež na začiatku vypočíta priemernú odchýlku ktorú potom odpočítava od hodnôt, ako je to vo vzorci vyššie vidieť. Do tejto odchýlky je rátané samozrejme aj tiažové zrýchlenie, aby bol v kľudnom stave výsledok podobný nule, a nie zemskej tiaži. Avšak naviac sa tu počíta aj hodnota threshold, čo je maximum z prvotne nameraných hodnôt, z ktorých sa vypočítava priemerná odchýlka. Ak sa neskôr zistí, že aktuálna nameraná hodnota je menšia ako threshold, tak sa usudzuje, že používateľ nepohol zariadením, lebo výkyv nameranej hodnoty je v rámci oblasti šumu. Ak nameraná hodnota prekročí threshold, vypočíta sa rýchlosť a vzdialenosť ako vo vyššie spomenutom skripte a aplikuje sa na polohu kamery.

4.2.6 Pristupovanie k polohe cez objekt LinearAccelerationSensor z knižnice motion-sensors.js a výpočet šumu

Tento skript pristupuje k polohe cez už vyššie spomenutý a použitý objekt LinearAccelerationSensor knižnice motion-sensors.js. Touto metódou sa teda získavajú údaje už bez tiažového zrýchlenia. Funkcionalitou je veľmi podobný predchádzajúcemu skriptu, ktorý ku polohe pristupoval cez objekt GravitySensor, akurát tu sa k vertikálnemu zrýchleniu pristupuje len cez jednu, x-ovú súradnicu. Tiež sa tu z tejto hodnoty počíta priemerná odchýlka a threshold spôsobom, akým to bolo navrhnuté v predchádzajúcich riešeniach. Následne sa vypočíta rýchlosť a vzdialenosť, ktorá sa aplikuje na pozíciu kamery.

```
      maxTr = maxTr < currentForce ? currentForce : maxTr;
      minTr = minTr > currentForce ? currentForce : minTr;

      counter++;
      avgBias += currentForce;
      if (counter == MAX_COUNT){
        avgBias /= MAX_COUNT;
        treshold = Math.abs(maxTr-avgBias) > Math.abs(avgBias-minTr) ?
            Math.abs(maxTr-avgBias)*1.2 : Math.abs(avgBias-minTr)*1.2;
      }
      return;
    }
  acceleration = currentForce - avgBias;
  if (Math.abs(acceleration) <= treshold) return;
```

Výpis 4.6: Kód získania premennej treshold

Obidva skripty majú definovanú minimálnu a maximálnu výšku kamery, takže po jej potenciálnom prekročení sa nastaví rýchlosť na nulu, čím kamera zostane na pôvodnom mieste. Skript bol vytvorený pre porovnanie rozdielov s predchádzajúcim skriptom, ale žiadne pozorovateľné odlišnosti neboli zistené.

4.2.7 Pristupovanie k polohe cez objekt LinearAccelerationSensor z knižnice motion-sensors.js a aplikovanie filtrov pre potlačenie šumu

Posledný skript tejto experimentálnej analýzy je postavený na rovnakom základe ako predchádzajúci, teda ku údajom pristupuje cez objekt LinearAccelerationSensor neberúc do úvahy gravitáciu. Na rozdiel od ostatných skriptov však používa pre potlačenie šumu dva typy filtrov. Tieto filtre sa snažia redukovať šum odstránením vysokých alebo nízkych frekvencií, respektíve hodnôt, odtiaľ názov low-pass a high-pass filtre, a tak minimalizujú účinok nežiaducich frekvencií. Tieto dva typy filtrov sú:

- low-pass filter,
- high-pass filter.

Low-pass filter funguje na princípe, že nová hodnota zrýchlenia je daná pomerom predchádzajúcej hodnoty a hodnoty, ktorú aktuálne dostal filter ako vstupný údaj z akcelerometra. To znamená, že filter si pamätá predchádzajúcu upravenú hodnotu, a tlmí alebo vyhladzuje nezvyčajné hodnoty, ktoré sú najčastejšie dôsledkom šumu. Pomer novej a starej hodnoty sa dá nastaviť ako vstupný parameter konštruktora filtra. Nevýhodou tohto riešenia je, že ak používa veľké percento existujúcej hodnoty na vypočítanie nasledujúcej, tak to predstavuje oneskorenie v registrácií skutočných udalostí [33].

```
    update(reading) {
  this.x = this.x * this.bias + reading.x * (1 - this.bias);
  this.y = this.y * this.bias + reading.y * (1 - this.bias);
  this.z = this.z * this.bias + reading.z * (1 - this.bias);
}
```

Výpis 4.7: Princíp fungovania low-pass filtra

High-pass filter funguje podobne ako low-pass filter, ale prepúšťa iba vysoké hodnoty. Toto je užitočné pre minimalizáciu odchýlky, ktorá môže časom lineárne narastať. Úroveň tlmenia je tiež možné nastaviť ako vstupný parameter konštruktora, v zdrojovom kóde je to premenná „cutoff“.

```
update(reading) {
  let dt = reading.timestamp - this.timestamp / 1000;
  this.timestamp = reading.timestamp;
  for (let i of ["x", "y", "z"]) {
    let alpha = this.cutoff / (this.cutoff + dt);
    this[i] = this[i] + alpha * (reading[i] - this[i]);
  }
}
```

Výpis 4.8: Princíp fungovania high-pass filtra

Použitím prvého alebo druhého typu filtra alebo ich kombináciou by mal skript produkovať o niečo presnejšie výsledky s potlačeným šumom. V skripte sa takisto ako v predchádzajúcich vypočíta priemerná odchýlka, threshold, zrýchlenie a vzdialenosť. Tieto hodnoty ale nie sú vyrátané na základe čistých dát zo senzorov, ale z hodnôt vypočítaných po ich spracovaní filtrami. Výsledná vzdialenosť sa nakoniec aplikuje na vertikálnu os kamery a ošetrí sa stav, ak výška kamery prekračuje hranice.

```
let sensor = new motion.LinearAccelerationSensor({frequency: 90});
const filter = new filters.LowPassFilterData(sensor, 0.7);
...
acceleration = filter.x - avgBias;
```

Výpis 4.9: Kód získavania pozície cez filtre a potlačenie šumu

4.2.8 Ohodnotenie kvality sledovania polohy cez polohové senzory

Po implementácií, dlhšom testovaní a experimentovaní s vyššie vymenovanými skriptami je možné urobiť záver, že nanešťastie všetky sú prakticky nepoužiteľné pre účely sledovania drobných zmien polohy telefónu v rozsahu štandardne niekoľkých centimetrov. Jedným z dôvodov je najpravdepodobnejšie fakt, že akcelerometre, ktoré sú inštalované do súčasných smartfónov, vedia dobre detegovať len náhle a prudké zmeny polohy, ako je trasenie telefónom. Avšak nevedia zisťovať pomalší, pomerne konštantný pohyb, ako je presun zariadenia napríklad z výšky očí do výšky kolien v čase približne 1,5 sekundy. Dôkazom tejto teórie je fakt, že

všetky skripty sa pri implementácií do scény správali až na drobné detaily rovnako. V prípade, keď bol telefón v statickej polohe a používateľ ním nehýbal, tak sa nemenila ani poloha kamery v scéne, teda pohľad do virtuálneho prostredia zostal tiež statický. Hneď ako používateľ pohol smartfónom v smere kolmom na zemský povrch (len tento smer bol sledovaný), tak sa zmenila aj pozícia kamery. Prvý problém bol, že počas toho, ako používateľ posúval smarfónom, či už vyššie alebo nižšie od počiatočnej polohy, tak to zariadenie nebolo schopné sledovať a pozícia kamery sa nemenila. Akonáhle uviedol používateľ telefón znova do pokoja, tak sa kamera opäť pohla. Senzory tak boli schopné zistiť len začiatok pohybu, aj to na počiatočný krátky moment, a koniec pohybu. To mohlo byť spôsobené tým, že na začiatku a konci zmeny polohy pôsobilo na zariadenie najväčšie preťaženie, ktoré bolo možné sledovať, avšak počas zmeny polohy nebolo možné toto preťaženie sledovať alebo odlíšiť od šumu. Samotný šum bol takisto značne výrazný, keďže zariadenie prostredníctvom výpisu hodnôt z akcelerometra do konzoly detegovalo konštantné drobné zmeny polohy aj keď bolo staticky položené na stole. Podobný šum bol sledovaný práve počas posunu telefónu hore alebo dole, okrem počiatku a konca zmeny polohy. Práve kvôli šumu bolo implementovaných viacero skriptov s rôznymi technikami na jeho potlačenie ale žiadna nebola dostatočne efektívna na to, aby sa získali dostatočne presné dáta. Druhý problém bol, že relatívne pomalý a jemný pohyb senzory neboli schopné sledovať vôbec. Čiže zmena polohy, teda jej začiatok a koniec, museli byť dostatočne razantné na to aby to senzory zachytili. Posledným problémom bol fakt, že pri začiatku a konci zmeny polohy pôsobilo na zariadenie preťaženie, respektíve zrýchlenie opačného smeru, čo malo dopad na zmenu polohy kamery v scéne. Čiže ak používateľ posunul telefón smerom hore, na začiatku pohybu pôsobilo na zariadenie preťaženie smerujúce do zeme a na konci pohybu zas preťaženie opačného smeru. To sa vo výsledku v scéne odzrkadlilo tak, že na začiatku pohybu sa kamera posunula jedným smerom (dole) a na konci zas opačným (hore). Kombinácia týchto problémov zapríčinila, že takéto správanie skriptov nebolo vyhovujúce pretože nesimulovalo korektne skutočný pohyb. Posledný skript aplikujúci low-pass či high-pass filtre vykazoval značne presnejšie výsledky ako predchádzajúce riešenia, avšak ani tento spôsob tlmenia šumu a vyhladzovania hodnôt nebol dostatočne účinný na reálne použitie v konečnom riešení.

4.3 Overenie sledovania polohy zariadenia prostredníctvom fotoaparátu

Druhý spôsob sledovania polohy je pomocou kamery. Vzhľadom na to, že A-frame počas písania tejto práce nepodporuje sledovanie polohy bez značiek, bude musieť byť použitá technológia tretej strany, ktorá bude integrovaná do tohto softvérového rámca a poskytovať mu údaje o polohe. Existuje veľa natívnych aplikácií ponúkajúcich AR zážitok, avšak existuje oveľa menšie množstvo riešení, ktoré ho ponúkajú použitím webových technológií s otvoreným alebo znova použiteľným kódom. Jedným takýmto riešením je AR webová aplikácia od spoločnosti Google, ktorá používa knižnice a skripty písané v Javascripte pre generovanie 3D modelov, ktoré sú vložené do výstupu z kamery. Takisto používa WebXR Device API, čo je nástupca pre WebVR API a je zatiaľ (v čase písania) stále vo vývoji. Toto API kombinuje VR a AR funkcionalitu vo webovom prostredí pre podporované smartfóny. Okrem tejto technológie využíva aplikácia ešte Google ARCore. Samotná aplikácia je celkom jednoduchá - po prejdení na jej webovú stránku a po stlačení tlačidla vstupu do AR sa spustí kamera a aplikácia začne sledovať prostredie, kam po krátkej chvíli vygeneruje virtuálne kocky. Na internete sa nachádza návod na vytvorenie tejto aplikácie, ktorý obsahuje postup, požiadavky, zdrojový kód aplikácie a iné informácie [34]. Vzhľadom na to, že WebXR API je počas písania experimentálna technológia, ktorá nie je vo finálnej podobe, treba tomu prispôsobiť aj určité požiadavky. Požiadavky pre správne fungovanie aplikácie teda sú:

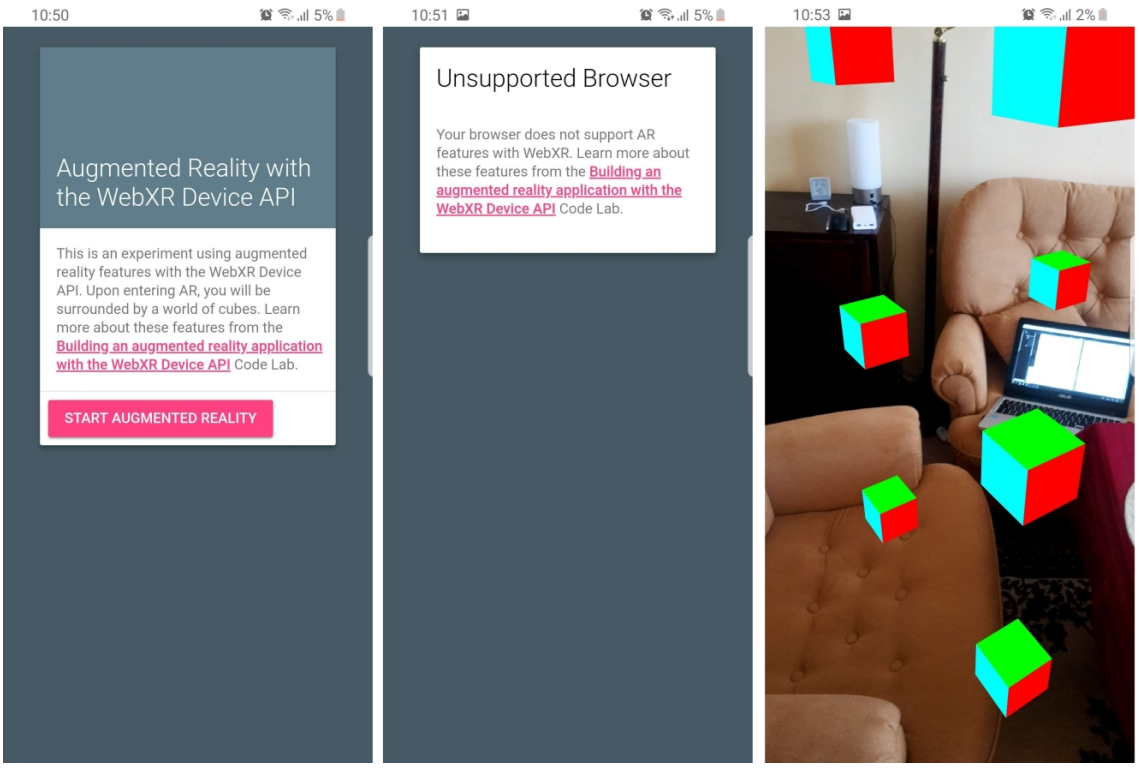
- smartfón s OS Android 8.0 a vyšší,
- smartfón s oficiálnou podporou pre ARCore,
- nainštalovaná aplikácia "Služby Google Play pre RR",
- nainštalovaný prehliadač "Chrome Canary"vo verzii 70 až 72 (len tieto verzie podporujú správne fungovanie WebXR API),
- a webový server a vývojové prostredie.

Ako vývojové prostredie bude znova použitý Glitch, ktorý ponúka aj funkcie servera a ktorému sa bude bližšie venovať kapitola Implementácia. Z webovej

stránky návodu na vytvorenie tejto AR aplikácie bol stiahnutý a detailne preštudovaný jej zdrojový kód. Tento kód okrem CSS súborov a pomocných knižníc pozostával z dvoch podstatných súborov. Prvým súborom bol *index.html*, teda HTML súbor webovej stránky. Samotná aplikácia pozostávala z dvoch obrazoviek - na prvej boli základné informácie a tlačidlo vstupu do AR, druhá obsahovala výstup z kamery, kam boli dokresľované virtuálne objekty. Tento HTML súbor reprezentoval prvú obrazovku. Odkazoval sa na niekoľko súborov, medzi ktorými boli CSS súbory na úpravu vzhľadu stránky a 3 Javascript súbory. Prvý skript obsahoval pomocné metódy pre ďalší skript slúžiace pre vykresľovanie kociek, druhý sa odkazoval na knižnicu Three.js. Three.js je API napísané v jazyku Javascript slúžiace na vytvorenie a vykreslenie dvoj a trojrozmerných objektov vo webovom prehliadači bez potreby inštalácie pluginov, a to vďaka používaniu WebGL API. Prostredníctvom Three.js API je v aplikácií renderovaná grafika. Posledný, tretí skript je druhým dôležitým súborom celej aplikácie. Má názov *app.js* a obsahuje kód, ktorý je spustený po tom, čo používateľ stlačí tlačidlo vstupu do rozšírenej reality. Obsahuje niekoľko dôležitých funkcií, od overenia kompatibility až po samotné renderovanie. Celý skript je jedna veľká trieda a na konci sa vytvorí inštancia tejto triedy nad webovou stránkou. Ako prvá sa v skripte zavolá funkcia, ktorá overí či zariadenie, respektíve webový prehliadač podporujú WebXR API. Ak ho prehliadač nepodporuje, teda overenie nebolo úspešné, stránka to oznámi používateľovi na prvej obrazovke a skryje tlačidlo vstupu do AR. V prípade úspešného overenia skript ponechá tlačidlo viditeľné, a naopak skryje text o nepodporovanom prehliadači. Následne skript čaká na používateľa, kým stlačí tlačidlo vstupu do AR. Ak to urobí, spustí sa ďalšia funkcia, ktorá vytvorí plátno, na ktoré bude renderovaný výstup z kamery v kombinácií s virtuálnymi objektami. Taktiež znova overí, či sa dá generovať WebXR obsah na toto plátno, a ak nie, tak znova zobrazí hlášku o neúspešnom overení. V prípade úspechu pridá toto plátno do webstránky a zavolá ďalšiu funkciu. Táto funkcia pripraví všetko potrebné a vytvorí scénu. Táto scéna obsahuje veľké množstvo kociek vygenerovaných za pomoci Three.js ktoré sa neskôr zobrazia používateľovi v priestore, ktorý natáča. Takisto sa v tomto kroku skryje všetok webový obsah z prvej obrazovky, a ponechá sa len plátno. Vytvorí sa kamera na sledovanie virtuálnych kociek pod správnym uhlom a zavolá sa posledná funkcia. Posledná funkcia mení animáciu na plátne a volá sa pri každej zmene animácie, čim je volaná rekurzívne dookola niekoľko krát za sekundu. Ako

prvé sa funkcia pokúša zistiť polohu telefónu v priestore prostredníctvom feedu z fotoaparátu. Táto poloha je relatívna vzhľadom na počiatočnú polohu zariadenia. Ak túto polohu nájde, prispôsobí kameru vo virtuálnej scéne a vyrenderuje ju pre každý pohľad. Pohľad môže byť jeden alebo dva, podľa toho či je zariadenie smartfón alebo VR headset. V prípade headsetu sú pohľady dva, keďže každé oko má svoj vlastný displej. Tento algoritmus sa opakuje dookola, teda zakaždým sa skript snaží nájsť pozíciu zariadenia v priestore a podľa toho generovať virtuálnu scénu, až kým používateľ nevypne prehliadač alebo nezatvorí kartu. Skript a jeho metódy sú takisto podrobnejšie vysvetlené vo vyššie spomenutom návode.

Súbory *index.html* a *app.js* boli pridané do prostredia Glitch pre overenie funkcionality aplikácie a potencionálnych možností využitia kódu pre účely VR prostredia. Pre úspešné spustenie aplikácie ešte musí byť správne nakonfigurovaný prehliadač Chrome Canary. Tento postup je bližšie špecifikovaný v nasledujúcej kapitole venujúcej sa návrhu aplikácie. Po splnení všetkých požiadaviek je možné navštíviť stránku aplikácie, ktorá zobrazí uvítaciu stránku s možnosťou vstúpiť do AR prostredia (Obr. 4.2 vľavo). Testovaním bolo zistené, že pre fungovanie aplikácie je potrebné použiť protokol *HTTPS* namiesto Glitchom predvoleného *HTTP* pri zadávaní URL adresy. Ak bude použitý HTTP protokol, skript bude považovať zariadenie a prehliadač za nespĺňajúce podmienky pre WebXR obsah a tým pádom vypíše hlášku o nepodporovanom prehliadači, ako je to znázornené na obrázku (Obr. 4.2 v strede). Šifrovaný protokol je tak podmienkou fungovania aplikácie a pri jeho použití má používateľ možnosť pokračovať na ďalšiu obrazovku. Po stlačení tlačidla zmiznú zo stránky grafické elementy a objaví sa prenos zo zadnej kamery zariadenia. Na prvých pár sekúnd sa vizuálne nič nedeje - vtedy sa skript snaží „zorientovať sa ", mapuje priestor a vytvára na pozadí trojrozmernú mapu. Po zmapovaní sa na obrazovke objaví niekoľko kociek (Obr. 4.2 napravo). V tejto chvíli je už všetko korektne nastavené a dá sa smartfónom pohybovať s tým, že virtuálne kocky sa budú správne pohybovať v priestore, akoby v ňom boli. Po minimalizovaní prehliadača a jeho následného otvorenia aplikácia ponechá túto obrazovku v pamäti, a znova sa na chvíľu bude snažiť zmapovať priestor. Za pár sekúnd sa tak kocky znova objavia.



Obr. 4.2: Prvá obrazovka AR aplikácie pri podporovanom (naľavo) a nepodporovanom (v strede) zariadení a druhá obrazovka aplikácie (napravo)

Aplikácia bola testovaná vo viacerých svetelných podmienkach s veľmi dobrými výsledkami. Vo všetkých prípadoch bola schopná kvalitne mapovať prostredie a správne vykresľovať virtuálne objekty. Bližší opis testovania tohto spôsobu sledovania polohy pri rôznych svetelných podmienkach je detailne vysvetlený v poslednej kapitole tejto Diplomovej práce. Pomocou testovania a rozboru skriptu bolo zistené, že poloha zariadenia (nie jeho orientácia) v priestore je uchovávaná v troch hodnotách jednorozmerného poľa čísel. Tieto tri hodnoty reprezentovali relatívnu zmenu polohy voči počiatočnej hodnote v 3 kolmých osiach, preto 3 čísla. Prostredníctvom konzoly boli vypísané všetky tieto vypočítané čísla o polohe zariadenia v reálnom čase.

```
let pose = frame.getDevicePose(this.frameOfRef);
      if (pose) {    //ak bola zistena poloha
         console.log(pose.poseModelMatrix[12]); //os X
         console.log(pose.poseModelMatrix[13]); //os Y
         console.log(pose.poseModelMatrix[14]); //os Z
         ...
```

```
    }
...
}
```

Výpis 4.10: Výpis zaznamenaných údajov do konzoly

Teda zakaždým, keď skript úspešne našiel pozíciu zariadenia, tak bola vypísaná a zaznamenaná. Takto sa dala jednoducho overiť presnosť a latencia výpočtu. Všetky čísla boli mapované na kvalitnej úrovni s nízkou dobou odozvy.

4.4 Porovnanie kvality sledovania pozície cez polohové senzory a kameru

V závere tejto kapitoly sa dá jednoznačne konštatovať, že výrazne presnejšie výsledky ponúka druhý spôsob, teda sledovanie pozície prostredníctvom fotoaparátu zariadenia. Tento spôsob nevykazuje žiadne nedostatky ani problémy, ktoré obsahuje spôsob sledovania prostredníctvom senzorov. Práve prvý spôsob je úplne nepoužiteľný pre sledovanie pozície vo VR aplikácií, pretože nedokáže rozpoznať jemné zmeny a naopak, sníma príliš veľký šum. Jedinou výhodou prvej metódy je kompatibilita, teda že väčšina súčasných telefónov disponuje gyroskopom, akcelerometrom a kompasom, takže sú schopné pomocou nich sledovať zmeny ich pozície. Na druhej strane, metóda sledovania prostredníctvom fotoaparátu je schopná sledovať aj veľmi malú zmenu stavu, stačí že sa zmení výstup z kamery. Nevýhodou je práve kompatibilita, a to z dôvodu že zariadení, čo podporujú použité technológie, je relatívne málo a sú to práve tie najvýkonnejšie. Ďalšia nevýhoda je, že pre správne fungovanie treba počas doby písania tejto práce používať špeciálny webový prehliadač v konkrétnej verzii a nastaviť niektoré prepínače (z angl. „flags"). Avšak akonáhle bude WebXR API dokončené a v stabilnej verzii, tieto kroky už nebude nutné vykonávať. Aj napriek týmto slabším stránkam ponúka technika zisťovania pozície cez kameru presné sledovanie a veľký potenciál a preto prebehne pokus o použitie tejto technológie na sledovanie pozície na pozadí vo VR aplikácií, kde bude podľa jej výstupu prispôsobená kamera vo virtuálnom prostredí. Implementácia tohto spôsobu do VR aplikácie bude vysvetlená v nasledujúcich kapitolách.

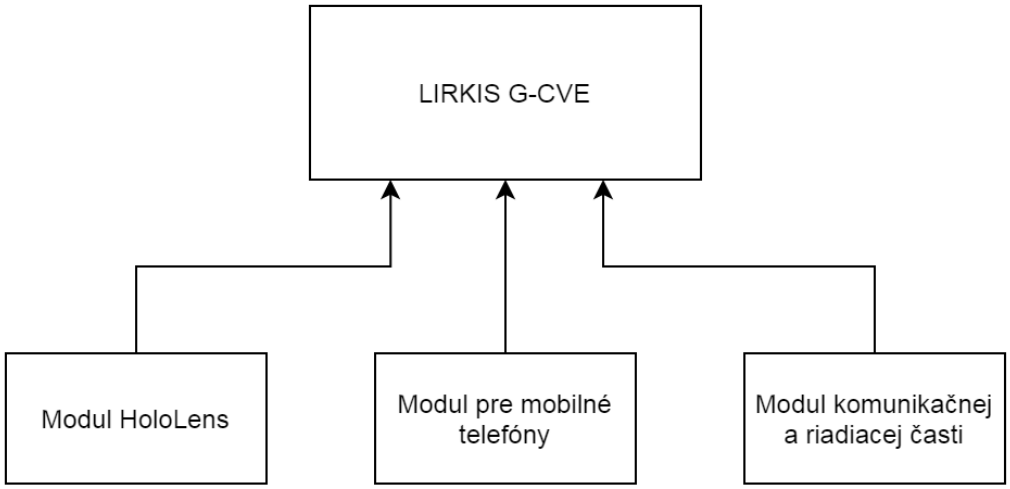
5 Návrh riešenia aplikácie

Táto kapitola sa bude venovať analýze spôsobu, ako by mala výsledná aplikácia fungovať a ako by sa mala správať a reagovať na akcie používateľa. Rozoberie návrh všetkých častí aplikácie, ako je opis konceptuálneho modelu, návrh systému a spôsobu komunikácie a výmeny dát so serverom či návrh modulu poskytujúceho sledovanie pozície telefónu prostredníctvom kamery. Najprv však kapitola bližšie opíše spôsob kolaborácie medzi jednotlivými riešiteľmi VR prostredia.

5.1 Kolaborácia s ďalšími prácami

Táto práca je súčasťou kolaborácie troch prác, z ktorých každá má na starosti inú problematiku rozšírenia viac-užívateľského, multiplatformové-ho VR prostredia. Toto prostredie už začalo byť vyvíjané a výsledné časti týchto troch prác doňho budú implementované [35]. Výsledná aplikácia bude spustená na serveri v LIRKIS laboratóriu na Technickej Univerzite v Košiciach. Meno VR aplikácie je Global Collaborative Virtual Environment, skrátene G-CVE. V rámci tejto práce je riešený vývoj aplikácie pre inteligentné telefóny s operačným systémom Android. Zároveň sa ďalšia práca venuje vývoju a optimalizácii VR prostredia a jeho funkcionality s headsetom Microsoft HoloLens [29]. Pri tejto práci bude taktiež vyvinutá funkcionalita sledovania fyzického prostredia cez senzory headsetu a následnej aplikácií zmeny polohy na avatara. Taktiež bude možná interakcia s prostredím pomocou vstavaného rozpoznávania gest rúk. Tretia práca rieši problematiku riadiacej a komunikačnej časti [36]. Tu patrí celková logika scény, vývoj použitých entít a objektov, komunikácia medzi používateľmi alebo možnosť prihlásenia sa do scény ako administrátor alebo klient. Na obrázku (Obr. 5.1) je možné vidieť všetky tri moduly virtuálneho prostredia, ktoré sú v súčasnosti vyvíjané. Na konci vývoja budú všetky časti spojené, a to tým spôsobom, že táto a časť venujúca sa

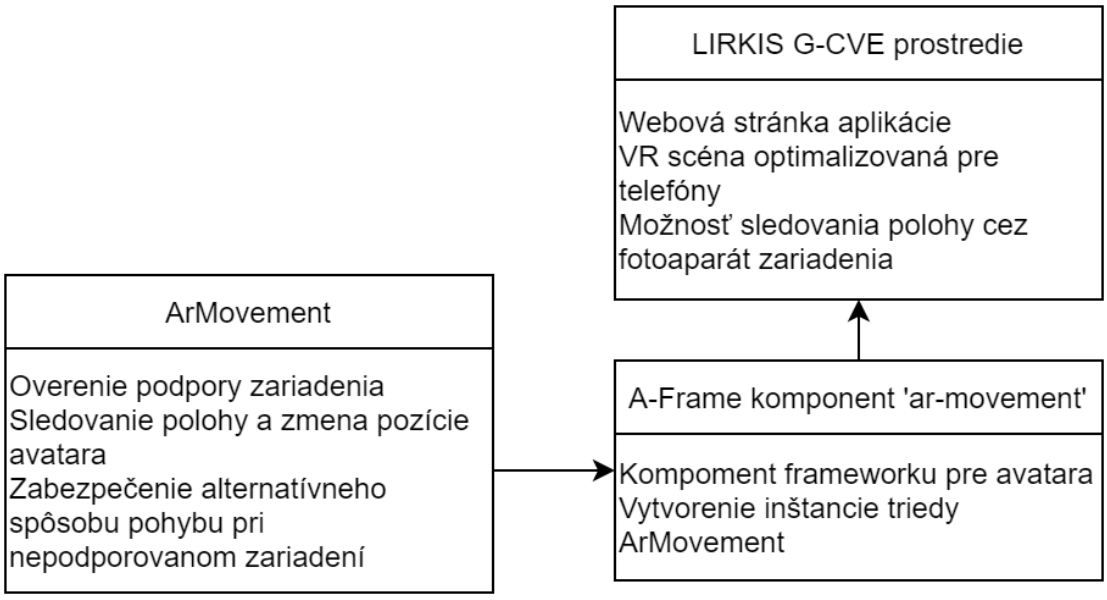
HoloLensu, respektíve ich esenciálna funkcionalita, budú importované do výslednej scény. Táto scéna pozostávajúca z oboch modulov pre smartfóny aj HoloLens bude nakoniec otestovaná a následne bude možné vyvodiť ohodnotenie výsledku tejto spolupráce.



Obr. 5.1: Kolaborácia pri vývoji G-CVE

5.2 Návrh riešenia

Na obrázku (Obr. 5.2) je znázornený návrh riešenia časti aplikácie pre mobilné telefóny. Časť aplikácie LIRKIS G-CVE určená pre mobilné telefóny bude pozostávať z HTML stránky, ktorá bude obsahovať samotnú scénu optimalizovanú pre použitie so smartfónmi s OS Android. Scéna bude tiež obsahovať HTML element hlavy avatara s novo vytvoreným komponentom s názvom *ar-movement*, cez ktorý bude možné pristupovať k HTML elementu, na ktorý bude naviazaný. Nakoniec bude vytvorená trieda *ArMovement* obsahujúca logiku samotného sledovania polohy telefónu. Tu bude vykonávané aj overenie, či zariadenie podporuje možnosť sledovania polohy. Trieda bude vykonávať sledovanie a cez komponent *ar-movement* bude pristupovať k HTML elementom scény.



Obr. 5.2: Návrh riešenia jednotlivých častí mobilnej VR aplikácie

5.3 Návrh systému

Multiplatformové kolaboratívne prostredie bude voľne dostupná webová aplikácia, čo znamená že používatelia sa na toto prostredie budú môcť pripojiť z podporovaného zariadenia cez internet. Keďže ide o VR aplikáciu, tak bude z dôvodu fyzického pohybu používateľov najčastejšie použité Wi-Fi pripojenie. Renderovanie virtuálnej scény a vypočítavanie zmeny polohy používateľa sa bude vykonávať na samotnom zariadení pre čo najrýchlejšiu odozvu systému a server bude slúžiť na nadviazanie spojenia, stiahnutie potrebných grafických modelov prostredia, zdieľanie polohy používateľov, potrebnú komunikáciu alebo iné pomocné výpočty. Existuje niekoľko dôležitých podmienok, ktoré budú musieť byť splnené pre úspešné nadviazanie spojenia, vykreslenie prostredia, sledovanie polohy a poskytnutie cieleného zážitku pri použití telefónu, a to sú:

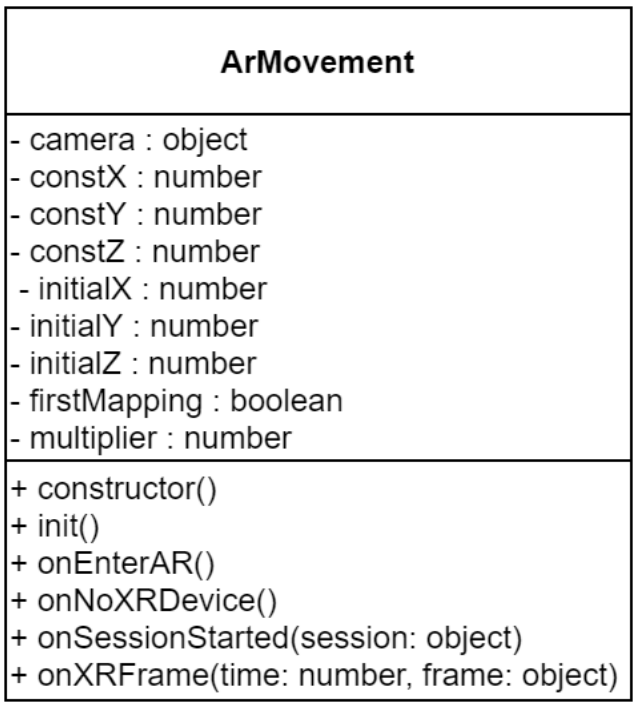
- zariadenia musí podporovať sledovanie svojej polohy a orientácie cez gyroskop a kameru,
- server musí byť dostupný aby sa naňho bolo možné pripojiť cez internet,
- zariadenie musí podporovať použité technológie, ako ARCore pre sledovanie polohy cez kameru.

Prvou podmienkou je, že zariadenie musí disponovať gyroskopom a kamerou. Kamera, respektíve fotoaparát, je súčasťou každého inteligentného zariadenia ľubovoľnej výkonnostnej triedy a gyroskop, ale aj akcelerometer či kompas sa dostávajú do čoraz väčšieho množstva telefónov. Tieto senzory sa počas minulých rokov stali oveľa lacnejšími a dostupnejšími a preto sú nimi vybavené veľa krát aj smartfóny nižšej triedy. Fotoaparát je nutný pre sledovanie relatívnej polohy smartfónu v priestore, zatiaľ čo gyroskop slúži na sledovanie orientácie v troch kolmých osiach. Druhá podmienka je podobne nenáročná ako prvá, a to že na webový server sa musí byť možné pripojiť, teda musí byť online. Celá aplikácia bude spustená na serveri so známou adresou, ktorú používateľ zadá do prehliadača. V prípade že prehliadač danú adresu nenájde, bude musieť používateľ overiť svoje internetové pripojenie alebo počkať kvôli potencionálnej chybe na serveri. Táto podmienka nezávisí na používateľovi a jeho zariadení, avšak stav servera môže mať dopad na celkový zážitok. Vzhľadom na to, že VR prostredie je viac-používateľské, sa môže stať že sa na web stránku prihlási priveľa používateľov, čo bude mať za následok preťaženie servera. V tomto prípade sa môže stať stránka nedostupnou alebo nestíhať spracovávať dáta od používateľov, ktorým môže VR scéna začať mrznúť alebo nebudú musieť vidieť aktuálnu polohu ostatných hráčov v scéne. Poslednou podmienkou je podpora použitých technológií vo VR prostredí. Keďže bolo v predchádzajúcej kapitole zistené, že pre sledovanie polohy zariadenia v priestore je najvýhodnejšie využiť technológiu ARCore, tak ju bude musieť zariadenie podporovať. Všetky zariadenia s touto technológiou by mali disponovať polohovými senzormi opísanými v predchádzajúcich kapitolách, čiže ak má zariadenie podporu pre ARCore, tak najpravdepodobnejšie obsahuje aj tieto potrebné senzory. Okrem tejto už nie je pri použití smartfónov špecificky nutná podpora žiadnej inej technológie.

5.4 Návrh modulu pre sledovanie pozície zariadenia prostredníctvom jeho kamery

Technológia sledovania polohy pomocou kamery opísaná v kapitole 4 je funkčný príklad a s malými zmenami je použiteľná vo vlastnej AR aplikácií. Avšak v aplikácií LIRKIS G-CVE z nej budú využité len určité aspekty. Preto bude musieť byť tento modul značne upravený a jeho funkcionalita pozmenená. Z predchádza-

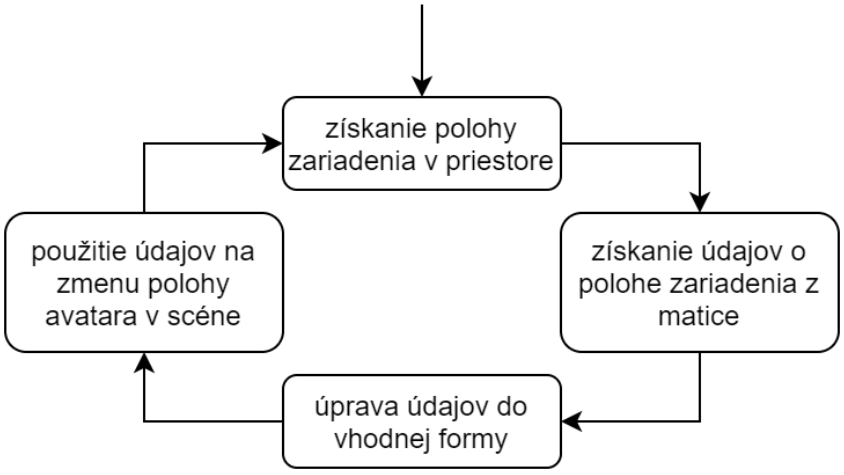
júceho riešenia bude použitý súbor *app.js* a pomocné knižnice pre jeho správne fungovanie. HTML súbor bude vynechaný, pretože vyvíjaná aplikácia bude obsahovať svoju vlastnú stránku obsahujúcu samotnú scénu. Skript bude musieť byť teda upravený aby spolupracoval s hlavnou stránkou a pristupoval k jej HTML elementom a zároveň aby neprekryl plátno scény so svojím, na ktorom renderoval AR scénu. Taktiež bude musieť byť odstránená časť zodpovedná za vykresľovanie virtuálnych objektov do AR scény kvôli možným problémom s výkonom. Časti nevyhnutné pre sledovanie polohy budú zachované. To môže mať potenciálne nežiadúci vplyv na výkon zariadenia, pretože telefón bude sledovať svoju polohu a zároveň aj vykresľovať virtuálnu scénu. Testovaniu a zisteniam sa bude venovať kapitola 7. Skript bude obsahovať všetky svoje pôvodné funkcie, ale pribudnú pomocné premenné, ako je to znázornené na obrázku 5.3.



Obr. 5.3: Diagram triedy sledovania polohy zariadenia

Keďže počiatočná pozícia, ktorú vypočíta tento skript na začiatku sledovania nie je nulová, musia byť pomocou pomocných premenných tieto údaje odpočítané od každého ďalšieho merania. Takisto k nim musí byť pri každom meraní pripočítaná konštantná hodnota počiatočnej pozície avatara v scéne, aby sa nestalo, že pri prvotnom meraní zmení svoju pozíciu v ktoromkoľvek smere. Nakoniec bude

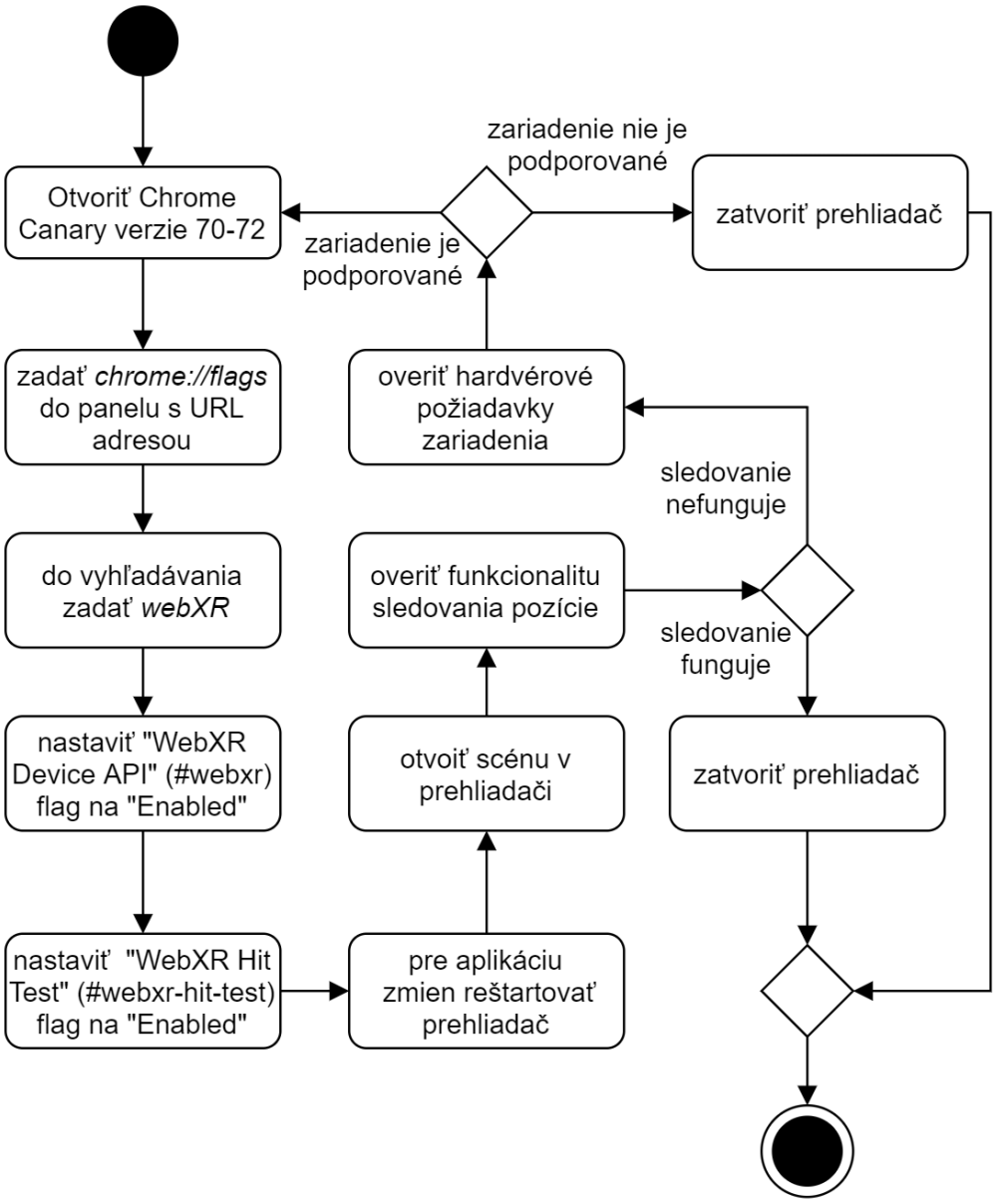
výsledný rozdiel hodnôt aktuálneho a počiatočného merania prenásobený konštantou s cieľom, aby pola zachovaná mierka zmeny polohy v reálnom svete aj v scéne. Ak teda používateľ vstane a posunie telefón o napríklad 50 centimetrov smerom hore, v scéne by tento pohyb mal tiež predstavovať posun o takúto vzdialenosť. Hodnota tejto konštanty bude upravovaná pri testovaní scény. Tento modul bude teda vo výsledku vykonávať nekonečný cyklus, ako je to znázornené na obrázku 5.4, až kým nebude zastavený používateľom v podobe zatvorenia webovej stránky. Slučka sa začína získaním polohy telefónu. Túto funkcionalitu vykonával skript aj doteraz. Avšak následne sa pristúpi k týmto dátam vo forme matice, kedy budú uložené do pomocných premenných na ďalšie spracovanie. Ďalej sa tieto dáta upravia vyššie spomenutým spôsobom pomocou pomocných premenných a konštánt a budú pripravené na poslednú časť cyklu. V tejto časti sa nakoniec hodnoty polohy priradia avatarovi v scéne, prípadne sa vypíšu do konzoly a celý cyklus sa znova opakuje.



Obr. 5.4: Diagram cyklu sledovania polohy

Ako bolo spomínané v predchádzajúcej kapitole, existujú aj softvérové požiadavky, ktoré budú musieť byť splnené pre fungovanie sledovania pozície. Prvou je správna verzia prehliadača Chrome Canary. Aj keď samotný A-Frame je podporovaný vo viacerých mobilných prehliadačoch ako Chrome či Mozilla Firefox, sledovanie polohy využívajúce WebXR API a Google ARCore je v čase písania podporované len vo vývojárskej verzii aplikácie Chrome vo verzii 70-72. Keďže cez oficiálny obchod Google Play nie je možné stiahnuť staršie verzie hier ani aplikácií, je nutné stiahnuť aplikáciu prehliadača cez internet a manuálne ju nain-

štalovať. V aplikácií je ďalej potrebné nastaviť dva flagy (nastavenia) na správnu hodnotu, ostatné ponechať predvolené a reštartovať aplikáciu. Celý postup správnej konfigurácie je znázornený na diagrame (Obr. 5.5) nižšie. V prípade, že používateľ správne nakonfiguruje prehliadač a sledovanie stále nebude fungovať, treba overiť či telefón spĺňuje hardvérové požiadavky aplikácie. Tie boli vymenované v predchádzajúcej kapitole a je to napríklad minimálna verzia OS Android či oficiálna podpora Google ARCore a takisto aj nainštalovaná táto služba.



Obr. 5.5: Diagram aktivity konfigurácie prehliadača

6 Implementácia

Táto kapitola sa bude venovať implementácií časti aplikácie určenej pre smartfóny do VR prostredia LIRKIS G-CVE. Presnejšie opíše implementáciu technológie sledovania pozície prostredníctvom kamery do scény vytvorenej prostredníctvom frameworku A-Frame. Na implementáciu bolo použité vývojové prostredie Glitch, ktoré bolo použité aj skôr pri experimentálnej analýze. Takisto kapitola bližšie vysvetlí spôsob ladenia aplikácie prostredníctvom počítača.

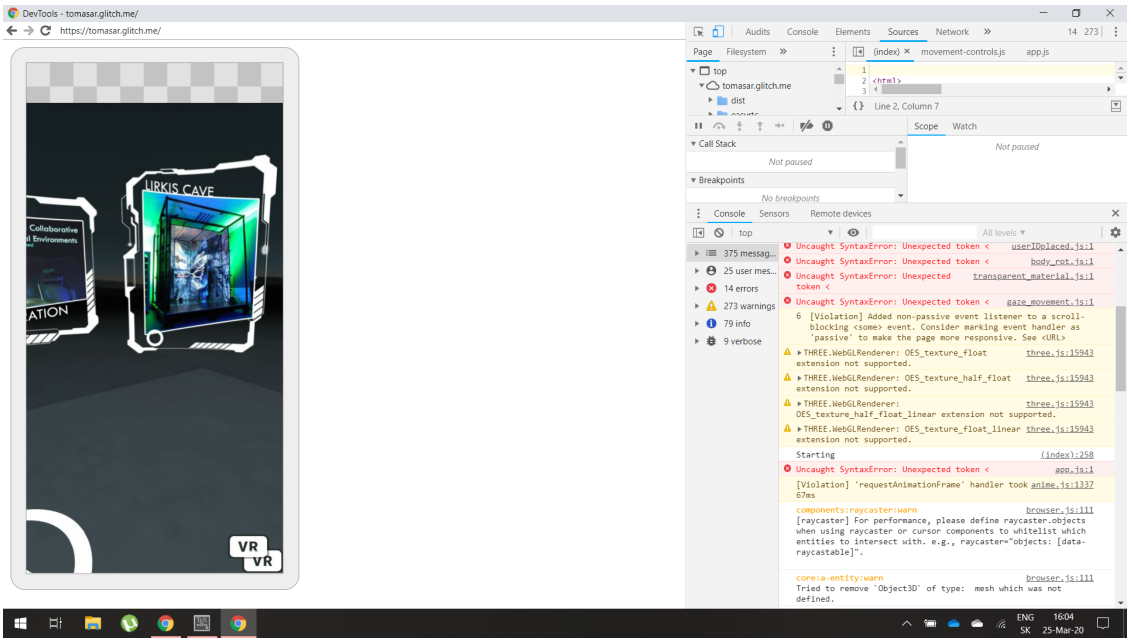
6.1 Ladenie

Pojem debuggovanie alebo ladenie programu znamená hľadať a znižovať počet chýb programu s cieľom, aby program pracoval tak, ako sa od neho očakáva. Debuggovanie softvéru prebieha najčastejšie prostredníctvom ladiacich nástrojov a výpisov. V Javascripte je možné vypísať reťazec znakov do konzoly webového prehliadača. Prehliadač Chrome pre smartfóny, ako aj jeho Canary verzia použitá pre sledovanie polohy cez kameru, nemajú priamu podporu konzoly a teda vývojár nemá možnosť vidieť výstup z konzoly mobilného prehliadača priamo na zariadení. Avšak pre takéto prípady existuje možnosť prepojiť tento prehliadač s jeho verziou určenou pre desktopy a notebooky prostredníctvom USB kábla. Pre prepojenie a sledovanie konzoly smartfónu na počítači je potrebné spraviť nasledujúce kroky:

1. v smartfóne povoliť „USB Debugging"cez „Možnosti pre vývojárov",
2. pripojiť telefón ku počítaču a pri autorizácií stlačiť „Vždy povoliť",
3. na PC prejsť na adresu *chrome://inspect* a na smartfóne otvoriť prehliadač Chrome alebo Chrome Canary,

4. v okne počítača sa zobrazí meno pripojeného zariadenia a všetky jeho otvorené karty v prehliadači, ktoré je možné obnoviť, prehliadať alebo zatvoriť.

Po stlačení tlačidla „Inspect", teda prehliadať, sa otvorí nové okno s vývojárskymi možnosťami, ako je to znázornené na obrázku 6.1.



Obr. 6.1: Okno s nástrojmi pre vývojárov

Na pravej strane je zrkadlenie obrazovky mobilu, teda je tu zobrazené všetko čo na jeho displeji. Na ľavej strane sú rôzne nástroje ako možnosť prezerania HTML elementov či zdrojov stránky alebo vyťaženosť telefónu. Jednou z možností je aj výstup z konzoly danej stránky, teda je možné týmto spôsobom sledovať výpisy zo skriptov, ktoré na danej stránke bežia. Takto sa budú sledovať rozličné dôležité údaje ako sú hodnoty premenných alebo sa takto bude zisťovať, či sa spustili rôzne metódy alebo skripty.

6.2 Opis použitej scény frameworku A-Frame

Scéna, do ktorej bude implementovaná technológia sledovania pozície cez kameru je značne komplexnejšia ako pôvodná scéna, v ktorej boli testované skripty zmeny polohy prostredníctvom akcelerometra. Nová scéna takisto obsahuje niektoré skripty, ktoré budú použité vo výslednej aplikácií. Táto scéna bola dodaná

pre túto Diplomovú prácu jej konzultantom a spolupracujúcim kolegom venujúcim sa problematike riadiacej a komunikačnej časti aplikácie LIRKIS G-CVE [36]. Dôvodom použitia tejto a nie finálnej scény bolo, že v čase implementácie bola výsledná scéna vo vývoji. Preto bude na konci výsledný zdrojový kód tejto časti a rovnako aj časti venujúcej sa implementácií scény do headsetu HoloLens skombinovaný s kódom finálnej scény, kedy budú testované aspekty ako komunikácia a oneskorenie medzi rôznymi použitými zariadeniami. Aktuálne použitá scéna pozostávala z HTML stránky a vyššie spomenutých skriptov. Základná HTML stránka obsahovala okrem esenciálnych tieto elementy a entity:

- odkazy na lokálne uložené skripty a skripty stiahnuté z internetu,
- element <a-scene> softvérového rámca A-Frame,
- assety ako grafické modely, obrázky či šablóny,
- entitu tela hráča pozostávajúcu z viacerých častí,
- statické objekty rozmiestnené v priestore scény,
- a schéma zdieľaných entít rámca Networked-Aframe slúžiaca na zdieľanie vlastností ako je poloha avatarov v scéne.

Okrem lokálnych skriptov stránka sťahuje niektoré z internetu. Sú to napríklad zdrojové kódy frameworkov A-frame a Networked-Aframe alebo iné knižnice. Assety (z anglického assets) sú modely, textúry alebo multimediálne dokumenty, ktoré framework dokáže dopredu načítať pred renderovaním samotnej scény kvôli lepšiemu výkonu. Tento spôsob zabezpečuje, že modely nebudú vizuálne chýbať v scéne kvôli pokusu o ich načítanie až pri vykresľovaní. Takže pri vstupe do scény sa na začiatku môže stať, že sa scéna vykreslí až po niekoľkých sekundách, avšak potom bude vykreslená so všetkými objektami. Šablóny (z anglického templates) sú tiež assety, ale sú to entity frameworku A-frame, a teda môžu obsahovať ďalšie entity alebo komponenty. Entita hráča pozostáva z entít jeho tela, hlavy, ukazovateľa, textu s jeho menom, rúk a systému šípok. Tieto šípky sa zobrazovali vždy pod avatarom a keď sa na nejakú pozrel, tak sa pohol smerom na ktorý ukazovala. Tento systém pohybu bude na telefónoch zmenený na prirodzený pohyb podľa polohy zariadenia. Obrázok tejto scény po menších vizuálnych úpravách bude znázornený v nasledujúcej podkapitole.

6.3 Úprava HTML stránky

Časť vyššie spomenutej HTML stránky scény bola pre podporu smartfónov zmenená, pričom jej štruktúra zostala zachovaná ako v kapitole 6.2. Keďže HoloLens aj smartfóny používajú iné komponenty, ich HTML stránky sa vo finálnej scéne budú líšiť. To bude riešené tak, že keď sa používateľ pripojí do scény, tak bude pomocou skriptov zistený typ zariadenia, ktorý používa. V závislosti od zariadenia bude potom presmerovaný na správnu stránku. Z testovacej scény bol odstránený komponent umožňujúci pohyb prostredníctvom vyššie spomenutých šípok a komponenty *look-controls* a *movement-controls*. Prvý komponent mal na starosti otáčanie hlavy podľa otáčania zariadenia prostredníctvom gyroskopu a druhý slúžil na pohyb smerom, ktorým používateľ pozeral. Tento pohyb bol realizovaný v telefónoch dotykom na displej a na počítači bol umožnený pohyb do štyroch smerov klávesami W,A,S,D. Ďalej bol do projektu pridaný skript sledovania polohy, ktorého implementácia a úprava bude vysvetlená v nasledujúcej kapitole. Odkaz na tento skript bol pridaný do stránky scény. Keďže je možné cez A-Frame pristupovať ku knižnici *Three.js*, tak odkaz na túto knižnicu už nemusel byť pridaný do HTML stránky. Skript sledovania polohy bude prerobený na A-frame komponent s menom „ar-movement" a preto bol priradený entite hlavy avatara, ktorá tiež obsahovala komponent kamery. Taktiež boli hlave z vývojárskych dôvodov pridané 3 textové polia so statickou pozíciou slúžiace na vypisovanie fyzickej polohy zariadenia. Nakoniec bolo do scény pridané tlačidlo po stlačení ktorého spustí skript algoritmus sledovania polohy. Dôvodom bolo, že sledovanie polohy musí používateľ vyvolať pomocou gesta, napríklad stlačením tlačidla. Táto požiadavka je nutná pre správne fungovanie algoritmu a nie je možné ju obísť.

```
<a-entity ... camera ar-movement position="0 1.6 0" >
   ...
      <a-text id="valx" value="X" position="-0.1 -0.2 -0.4" ></a-text>
      <a-text id="valy" value="Y" position="0.0 -0.2 -0.4"></a-text>
      <a-text id="valz" value="Z" position="0.1 -0.2 -0.4"></a-text>
  </a-entity>
     ...
     <button style="..." id="enter-ar" title="Enter AR.">AR</button>
```

Výpis 6.1: Pridanie komponentu, entít a tlačidla

Na obrázku 6.2 je znázornená scéna po úpravách HTML stránky.

Obr. 6.2: Testovacia scéna s textovými poliami a tlačidlom

6.4 Implementácia skriptu sledovania polohy

Do projektu bol pridaný skript sledovania s názvom „ArMovement" opísaný v podkapitole 5.4 (Obr. 5.3). Prvou zmenou bolo vytvorenie komponentu ar-movement, ktorý bol priradený entite hlavy. Keďže pôvodne skript tvorila jedna trieda, tak pri inicializácií komponentu sa vytvoril objekt tejto triedy obsahujúci logiku sledovania. Taktiež tu bol uložený HTML element viazaný na tento komponent, čo predstavovalo entitu hlavy hráča.

```
AFRAME.registerComponent('ar-movement', {
  init: function () {
    camera = this.el;
    window.arMovement = new ArMovement();
  }
```

```
});
```

Výpis 6.2: Vytvorenie komponentu ar-movement

Samotná trieda obsahujúca logiku sledovania bola premenovaná na „ArMovement". V konštruktore tejto triedy boli entity textových polí, kamera a jej pozícia priradené pomocným premenným pre ďalšie spracovanie. Na konci bola volaná asynchrónna funkcia *init*() tejto triedy. Asynchrónne funkcie vykonávajú svoje inštrukcie oddelene od zvyšku kódu, keďže je predpoklad že ich inštrukcie nemôžu byť vykonané v reálnom čase [37]. Slúžia ako prevencia voči zamrznutiu systému. Keďže tento skript vykonáva pri inicializácií a príprave na sledovanie časovo náročné výpočty, sú tu použité asynchrónne funkcie. Preto po stlačení tlačidla sledovania polohy trvá zariadeniu pár sekúnd, kým ju začne monitorovať. Táto asynchrónna funkcia na začiatku overuje, či zariadenie podporuje WebXR API a či bol povolený flag *webxr-hit-test*. Ak podmienky neboli splnené, zavolá funkciu *onNoXRDevice*(), prostredníctvom ktorej je zobrazené vyskakovacie okno informujúce používateľa, že sledovanie polohy nie je možné. V prípade úspechu vytvorí dva listenery, jeden viazaný na tlačidlo sledovania polohy, a druhý na stlačenie klávesy. Takto používateľ môže začať sledovať polohu aj prostredníctvom ovládača headsetu mobilného zariadenia. Po stlačení klávesy alebo tlačidla na obrazovke sa zavolá asynchrónna funkcia *onEnterAR*(). Tu sa vytvorí HTML element canvas slúžiaci na zobrazenie AR scény. Tento element je potrebný, pretože sa prostredníctvom neho vytvorí session, ktorá je použitá ako argument pri volaní ďalšej funkcie. Prostredníctvom premennej session je získavaná poloha zariadenia a v pôvodnom skripte aj renderovaná AR scéna. Element canvas je v tejto metóde nastavený na neviditeľný, pretože slúži len ako časť procesu získavania polohy. Takisto je skryté aj tlačidlo sledovania polohy, keďže ho už v tomto kroku používateľ stlačil a takisto sa zabezpečí väčšie pohltenie bez interaktívnych prvkov. V prípade, že inicializácia premennej session neprebehla úspešne, zavolá sa znova *onNoXRDevice*(). V opačnom prípade je canvas pridaný do tela HTML dokumentu a zavolá sa funkcia *onSessionStarted*(*session*). V tejto metóde je vytvorený WebGLRenderer, ktorý obsahuje druhé plátno. Po zistení jeho kompatibility so zariadením sa vytvorí pomocou neho základná vrstva spomenutej premennej session. Na túto vrstvu (z angl. layer) bude kreslená AR scéna, čo je v tomto prípade výstup z kamery bez virtuálnych objektov, ktorá sa zobrazí na HTML elemente canvas. V

tejto metóde bola pôvodne vytvorená virtuálna scéna obsahujúca objekty vmiešavané do reálneho sveta, avšak táto funkcionalita nie je potrebná pre vyvíjanú technológiu. Následne sa vytvorí objekt XRFrameOfReference, ktorý poskytuje priestorové informácie o bode, z ktorého sa uskutočňuje meranie. Nakoniec sa zavolá posledná metóda *onXRFrame*() pomocou funkcie requestAnimationFrame(). Táto metóda je volaná s dvoma argumentami. Prvým je premenná time predstavujúca ubehnutý čas od spustenia scény v milisekundách. Druhý objekt menom frame obsahuje spomínanú session scény. Tá vďaka objektu XRFrameOfReference z predchádzajúcej metódy obsahuje údaje o relatívnej pozícií zariadenia v priestore v podobe matice so 16 prvkami. Na začiatku metódy je tiež volaná funkcia requestAnimationFrame s argumentom *onXRFrame* čím sa zabezpečí volanie tejto metódy a sledovanie pozície v slučke. V prípade, že pozícia nebola nájdená sa metóda skončí. V druhom prípade sa pristúpi k matici polohy a údaje o pozícií sú dostupné. Teraz je možné s nimi pracovať a meniť polohu avatara v scéne. Ak ide o prvé meranie, a teda bola funkcia *onXRFrame* volaná prvýkrát a neexistujú dáta o predchádzajúcom meraní, uložia sa dáta o polohe zariadenia v troch osiach a dáta o pozícií kamery v scéne do pomocných premenných a funkcia sa ukončí. Je to tak z dôvodu, že údaje o predchádzajúcej polohe a pozícií kamery sú použité vo vzorci na výpočet aktuálnej polohy avatara. Pri všetkých nasledujúcich volaniach funkcie je pozícia avatara vo všetkých troch osiach vypočítaná podľa výpisu nižšie.

```
camera.object3D.position.x = constX + ((object.position.x - initialX) *
    multiplier);
 camera.object3D.position.y = constY + ((object.position.y - initialX) *
     multiplier);
camera.object3D.position.z = constZ + ((object.position.z - initialX) *
    multiplier);
```

Výpis 6.3: Funkcia zmeny pozície avatara

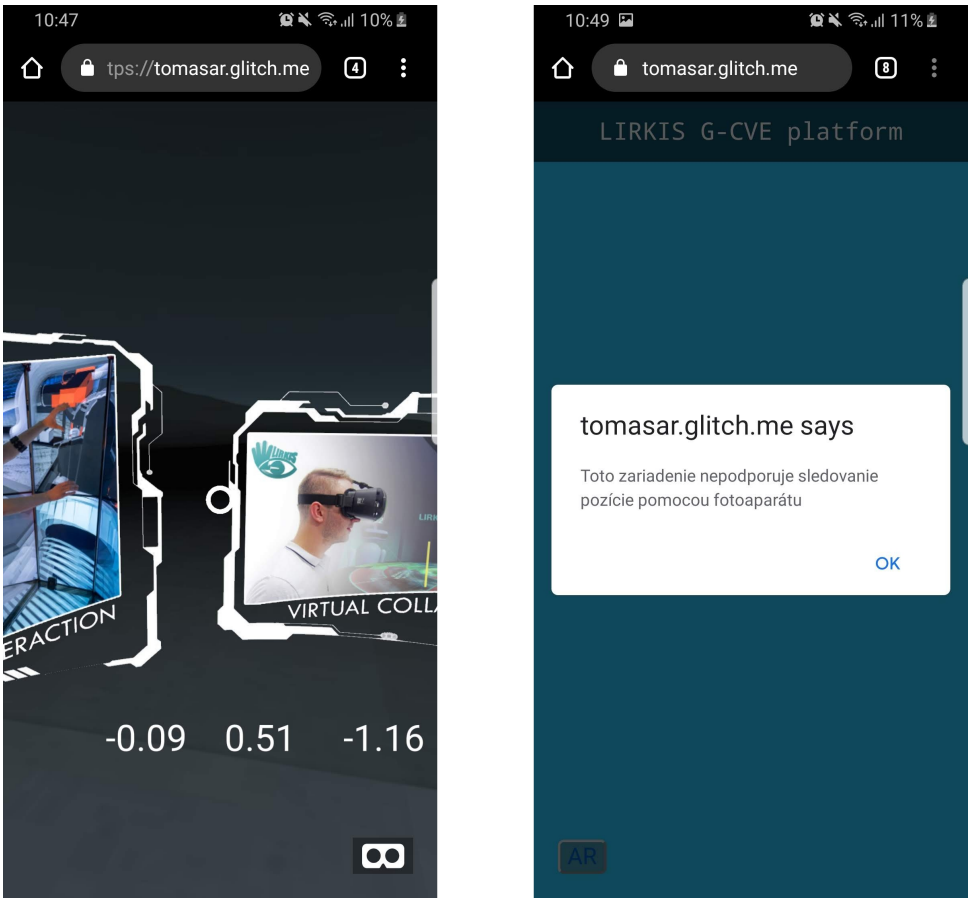
Pričom *camera.object3D.position.x* je nastavovaná pozícia avatara v scéne, *constX* je pozícia hlavy avatara pri prvom meraní polohy, *object.position.x* je aktuálne zistená relatívna poloha zariadenia v priestore, *initialX* je relatívna poloha zariadenia pri prvom meraní a *multiplier* je konštanta pre zmenu mierky veľkosti pohybu. To platí aj pre zvyšné dve osi. Na konci metóda ešte vypíše aktuálnu polohu zaria-

denia do textových polí pre každú os a ukončí sa.

6.5 Overenie algoritmu sledovania polohy a celkovej funkčnosti scény

Scéna bola v tomto štádiu pripravená na testovanie. Bola dostupná na webovej adrese *https://tomasar.glitch.me/*, čiže sa na ňu dalo pristúpiť z akéhokoľvek zariadenia či miesta. Na testovanie a overovanie bolo použité zariadenie Samsung Galaxy S8 s OS Android 9.0, procesorom Exynos 8895, 4GB pamaťou RAM, 12 MP fotoaparátom s optickou stabilizáciou a clonou f/1.7. Tento telefón patril medzi podporované ARCore zariadenia. Bol tu nainštalovaný Chrome Canary a zariadenie splňovalo všetky hardvérové aj softvérové požiadavky pre sledovanie polohy. Na testovanie bol po dohode s vedúcim a konzultantom práce použitý A-Frame verzie 0.9.0 a Networked-Aframe verzie 0.6.0. Po otvorení stránky a zadaní mena používateľa sa načítala scéna. V tomto momente sa nedalo v scéne pohybovať, keďže boli odstránené komponenty ktoré to umožňovali. Po stlačení tlačidla sledovania polohy sa približne 3 sekundy nič nedialo okrem zmiznutia tohto tlačidla. Následne sledovanie začalo fungovať, v troch textových poliach sa objavili hodnoty relatívnej polohy a bolo možné sa v scéne premiestňovať. Na obrázku (Obr. 6.3) vľavo je možné vidieť výstup zo scény pri úspešnom sledovaní polohy. V prípade nesplnenia určitých podmienok, ako je napríklad použitie *HTTP* protokolu namiesto *HTTPS* alebo obyčajnej verzie prehliadača Chrome aplikácia po zadaní prezývky vypísala okno s oznamom o nepodporovanom zariadení, ako je to možné vidieť na obrázku (Obr. 6.3) vpravo. Po stlačení tlačidla „OK"však bolo možné vstúpiť do scény.

Aktuálna implementácia prinášala dva nedostatky. Prvý bol, že skript sledoval len zmenu pozície a nie aj orientácie a tým pádom nebolo možné sa vo svete rozhliadnuť, teda sa pozrieť iným ako predvoleným smerom. Tento bug bol odstránený opätovným pridaním komponentu *look-controls* entite hlavy. Vtedy bolo možné už otáčať hlavou, avšak pri zapnutí sledovania polohy začala veľmi rýchlo kmitať kamera v rozličných osiach. Približne každý druhý obrázok mal posunutú pozíciu hlavy avatara asi o tretinu vzdialenosti medzi podlahou a kamerou v rôznych osiach. To spôsobovalo veľmi rýchle skákanie kamery medzi dvoma polohami a scéna sa stala nepoužiteľnou. Preštudovaním iných zdrojových kódov po-



Obr. 6.3: Testovacia scéna s funkčným a nefunkčným sledovaním polohy

užívajúcich túto technológiu bolo zistené, že matica spomínaná v predchádzajúcej kapitole obsahovala okrem údajov o polohe aj údaje o orientácií, ktoré boli sprístupnené po jej dekompozícií. Keďže táto technológia sledovania polohy neposkytovala dostatočne detailnú dokumentáciu, bolo nutné sa dostať ku potrebným informáciám štúdiom podobných skriptov alebo hľadaním v dokumentáciách frameworkov. Spomenuté údaje mohli byť získané prostredníctvom metódy decompose volanej na 4x4 matici, ktorá z nej extrahovala dáta o pozícií, rotácií a mierke.

```
poseMatrix.decompose(object.position, camera.object3D.rotation,
    object.scale);
```

Výpis 6.4: Pristupovanie ku polohe a orientácií

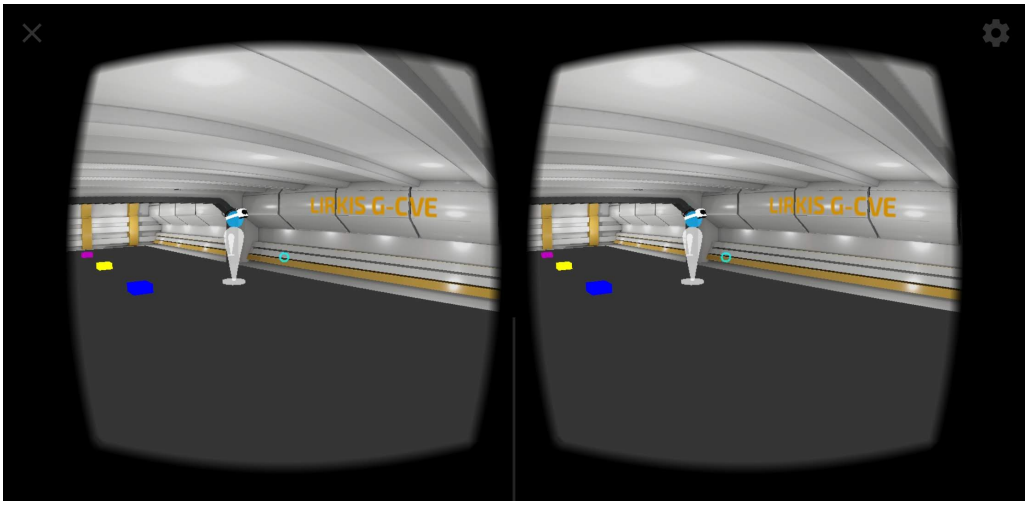
Vo výpise vyššie boli údaje o orientácií hneď priradené entite hlavy avatara a údaje o pozícií boli uložené pre dodatočné spracovanie. Výsledkom toho prístupu bola funkčná a plynulá zmena polohy aj orientácie hlavy avatara. Druhým

nedostatkom bola funkčnosť skriptu len v bežnom móde. To znamenalo, že ak používateľ prešiel do VR-módu znázorneného na obrázku vyššie (Obr. 6.3) tlačidlom v pravom dolnom rohu, sledovanie polohy bolo zastavené. V tomto móde fungovalo len sledovanie orientácie. Akonáhle bol tento mód vypnutý, aplikácia prešla do bežného módu kde sa opäť automaticky spustilo sledovanie zariadenia v nezmenenej kvalite. Toto správanie bolo testované na viacerých verziách frameworku A-Frame, ako 0.9.0, 0.9.2, 1.0.0 či 1.0.3 a nakoniec bolo zistené, že verzia 0.9.2 na rozdiel od ostatných podporuje sledovanie aj vo VR móde. To znamenalo, že používateľ môže mať telefón vo VR headsete ako je Google Cardboard na hlave, a pohybovať sa v scéne prostredníctvom sledovania pozície telefónu. Táto verzia však obsahovala bug, kedy pri vstupe do scény nebola orientácia hlavy avatara na predvolenej nulovej hodnote, ale získala sa aktuálna pozícia z gyroskopu. Keďže vo VR móde nebolo možné vypnúť predvolené sledovanie rotácie frameworkom, spôsobovalo to nesúlad medzi orientáciu hlavy avatara a telefónom v skutočnom prostredí. Táto chyba sa dala odstrániť použitím inej verzie frameworku A-Frame, avšak tie zastavili fungovanie skriptu pri VR-móde. Preto pre správne fungovanie skriptu vo VR-móde musí používateľ orientovať telefón do nulovej hodnoty Y-ovej osi a obnoviť stránku. Nulová hodnota Y-ovej osi je tá, na ktorú sa orientuje telefón pri začatí sledovania polohy (pravá stena scény).

6.6 Implementácia skriptu do finálnej scény

Po implementácii a prvotnom testovaní algoritmu sledovania polohy bola dodaná finálna scéna s názvom LIRKIS G-CVE [35]. Táto scéna bola vytvorená konzultantom tejto práce a riešiteľom riadiacej a komunikačnej časti. Scéna ponúkala na základnej obrazovke tri tlačidlá, a to vstúpiť do VR prostredia ako administrátor, robot alebo používateľ. Administrátor aj robot vyžadovali prihlasovacie heslo, kde administrátor mohol spravovať scénu a jednotlivých používateľov a robot mal v scéne vlastný pohyb a jeho používateľ mohol sledovať kam ide. Bežnému používateľovi bola teda prístupná posledná možnosť. Po výbere tejto možnosti bol presmerovaný na stránku, kde bolo potrebné zadať nickname a následne bolo možné vstúpiť do scény. Bližší opis tejto scény poskytuje diplomová práca komunikačnej a riadiacej časti tejto VR aplikácie [36]. Samotná scéna bude pozostávať z dvoch HTML súborov, jeden pre HoloLens a druhý pre mobilné telefóny.

Do hlavnej stránky bol pridaný skript, ktorý deteguje typ zariadenia. Ak zistí, že používané zariadenie je smartfón, otvorí príslušnú stránku. Do tejto stránky bol pridaný odkaz na skript sledovania *arMovement.js*. Takisto bolo pridané tlačidlo spustenia sledovania, pridaný komponent sledovania entite hlavy a odstránené komponenty *look-controls* a *movement-controls*. Na stránke bol použitý po dohode s konzultantom práce A-Frame vo verzii 0.9.2, pretože ponúkal možnosť sledovania polohy aj vo VR-móde. Skript sledovania pozície bol upravený tak, aby po detekcii nepodporovaného zariadenia pridal naspäť komponenty *look-controls* a *movement-controls*. Tým je umožnené používať scénu ľubovoľnému inteligentnému telefónu s podporovaným prehliadačom. Keďže A-Frame vo verzii 0.9.2 podporuje aj Chrome pre osobné počítače, ktoré skript deteguje ako nepodporované pre sledovanie polohy, je takto umožnené používať stránku aj akémukoľvek počítaču. Na obrázku 6.4 je znázornená finálna scéna použitím VR-módu telefónu.



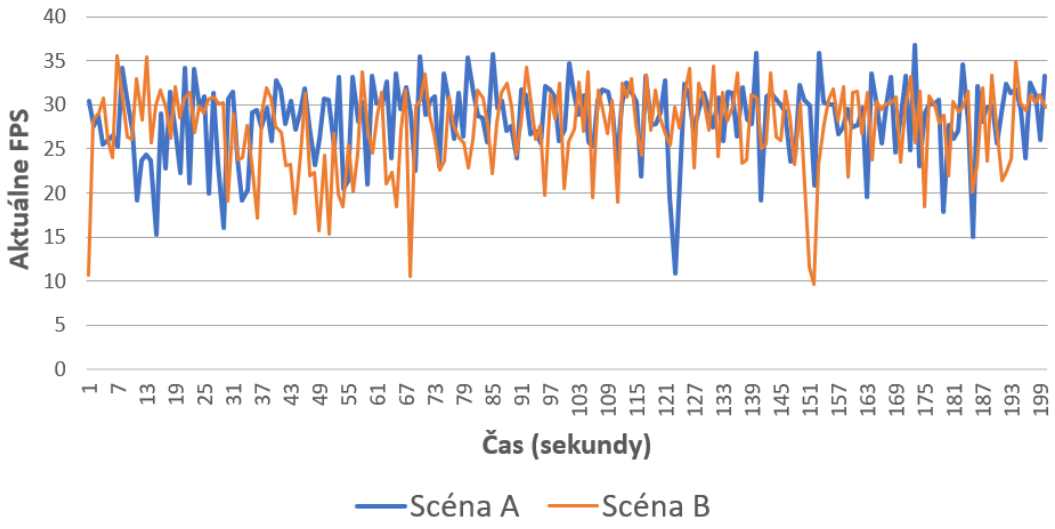
Obr. 6.4: Ukážka finálnej scény LIRKIS G-CVE

7 Zhodnotenie scény a jej komponentov formou testovania

Finálna VR scéna v tomto štádiu poskytovala možnosť sledovania polohy pre mobilné zariadenia a bola pripravená na testovanie. Na to bol použitý rovnaký smartfón ako v predchádzajúcej scéne. Celý proces testovania a dáta použité v tejto kapitole boli získané v domácom prostredí z dôvodu globálnej pandémie, kvôli ktorej bola vysoká škola a laboratórium LIRKIS zatvorené v marci a apríli 2020. Táto kapitola sa bude venovať rôznym druhom testovania, ako je sledovanie obnovovacej frekvencie obrazovky pri zapnutej scéne, porovnanie kvality sledovania pozície telefónu pri rôznych svetelných podmienkach alebo sledovanie kvality spojenia medzi viacerými pripojenými používateľmi. Nakoniec kapitola porovná aplikáciu s podobnými existujúcimi riešeniami.

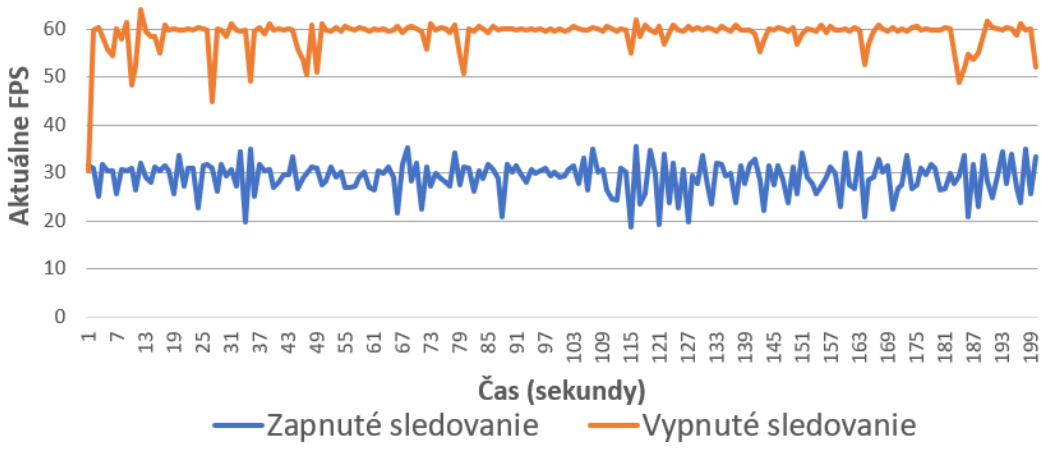
7.1 Testovanie obnovovacej frekvencie scény

Prvým druhom testovania bolo sledovanie FPS scény. Pre porovnanie bola otestovaná aj predchádzajúca scéna, na ktorej bol skript vyvíjaný. Na obidvoch scénach bolo na začiatku spustené sledovanie polohy a pozícia aj orientácia zariadenia boli počas testu konštantne menené. Scény boli spustené na približne 3 minúty za predpokladu, že sa FPS za takýto čas ustáli a predíde sa chybám merania. Na obrázku (Obr. 7.1) je znázornený graf s horizontálnou osou predstavujúcou ubehnutý čas a vertikálnou osou znázorňujúcou obnovovaciu frekvenciu obrazovky v aktuálnom čase. Scéna A v tomto grafe predstavuje testovaciu scénu a Scéna B finálnu scénu LIRKIS C-GVE.



Obr. 7.1: FPS v jednotlivých scénach

Z grafu je vidieť relatívnu nestabilitu počtu snímkov za sekundu, ktorá sa viac prejavuje vo finálnej scéne. V niektorých prípadoch kleslo jej FPS pod 20, v extrémnych prípadoch to bolo približne 10. Na druhú stranu, vo väčšine prípadov sa počet obrázkov za sekundu v Scéne B pohyboval v rozmedzí 23-33. Priemerné FPS Scény A bolo 28,3 a Scény B 27,35. Z týchto dát sa dá povedať, že rýchlosť vykresľovania nie je pri scénach značne odlišná, aj keď Scéna A bola graficky jednoduchšia. Z tohto dôvodu bol vytvorený druhý graf (Obr. 7.2) porovnávajúci finálnu scénu so zapnutým a vypnutým sledovaním polohy. Pri vypnutom sledovaní bol použitý klasický prehliadač Chrome a avatar sa takisto neustále pohyboval.



Obr. 7.2: FPS pri zapnutom a vypnutom sledovaní vo finálnej scéne

Z týchto dát sa dá sledovať, ako je pre telefóny výpočtovo náročné sledovať polohu cez kameru a zároveň vykresľovať grafický obsah. V tomto prípade sa frame rate pri zapnutom sledovaní najčastejšie pohyboval medzi 25-32 a pod 20 klesol len sporadicky. Priemerné FPS bolo v tomto meraní 29,1. Nedá sa to však porovnať s priemernou hodnotou 58,99 FPS dosiahnutej pri scéne s vypnutým sledovaním pozície. Keďže obnovovacia frekvencia displeja použitého telefónu bola 60Hz, je možné že teoretický výkon zariadenia je ešte vyšší. Pri týchto podmienkach je teda pri zapnutom sledovaní obnovovacia frekvencia približne polovičná oproti vypnutému. Aj keď je táto hodnota značne nižšia a ak by bolo možné prejsť do VR-módu tak by bol obraz trhavý, pri bežnom móde je táto hodnota FPS dostačujúca pre poskytnutie požadovaného zážitku z aplikácie. Výber vhodného telefónu je tiež dôležitý aspekt a dnešné high-end zariadenia poskytujú troj-generačný rozdiel vo výkone oproti použitému modelu, čím pri ich použití môže frame rate stúpnuť až o niekoľko desiatok percent.

7.2 Testovanie kvality sledovania telefónu pri rôznych svetelných podmienkach

Ďalší typ testovania bolo porovnanie kvality sledovania pozície pri rozdielnych svetelných podmienkach. Pri tomto testovaní bola tiež overovaná mierka zmeny pohybu, teda či relatívna vzdialenosť pohybu v scéne sa zhodovala so vzdialenosťou posunu v skutočnosti. Túto mierku bolo možné v skripte meniť úpravou spomínanej konštanty *multiplier*, a po viacerých pokusoch bola nastavená na hodnotu 2. Táto hodnota poskytovala subjektívne najpresnejšiu vzdialenosť pri zmene pohybu. Toto testovanie sledovania polohy bolo vykonané v úzkej izbe so zrkadlami pre vytvorenie menej ideálnych podmienok pre mapovanie priestoru. Za jasného počasia sa takisto silne odrážalo svetlo od podlahy, a to z dôvodu zistenia limít technológie, pri akých podmienkach bude skript poskytovať nepresné výsledky alebo bude mať príliš veľkú odozvu. Na obrázku (Obr. 7.3) sú znázornené štyri svetelné situácie, pri ktorých prebiehali testy. Tieto situácie znázorňujú približne taký jas, aký bol v danom okamihu viditeľný ľudským okom. Situácia A predstavovala výborné svetelné podmienky pri jasnom slnečnom počasí. Situácia B reprezentovala značne horšie podmienky pri šere bez zapnutého osvetlenia. Naj-

horšie svetelné podmienky predstavovala situácia C pri zapadnutom slnku a pri zapnutej slabej žiarovke osvetľujúcej približne tretinu miestnosti. Pri poslednej, situácii D, bola vonku úplná tma a v izbe svietilo jedno hlavné svetlo. Táto scéna predstavovala druhé najlepšie podmienky. Pri všetkých situáciách bola aplikácia spustená na približne 3 minúty a v prípade nekorektnej funkcionality bola otestovaná viackrát.

Obr. 7.3: 4 situácie s rôznymi svetelnými podmienkami

Výsledky meraní sú nasledujúce:

- sledovanie polohy bolo vykonávané pri situácii A na veľmi vysokej úrovni, bez akýchkoľvek mrznutí aplikácie alebo nepresného merania,
- situácia B ponúkala veľmi podobné, až rovnaké kvalitatívne výsledky ako predchádzajúci príklad napriek značne zhoršeným podmienkam,
- pri situácií C došlo k výraznému zhoršeniu kvality, veľkej odozve, mrznutiu aplikácie a k skokovým zmenám pozície a orientácie avatara,
- a situácia D poskytovala dostatočne dobré podmienky pre kvalitné sledovanie na úrovni situácie B bez mrznutia či odozvy.

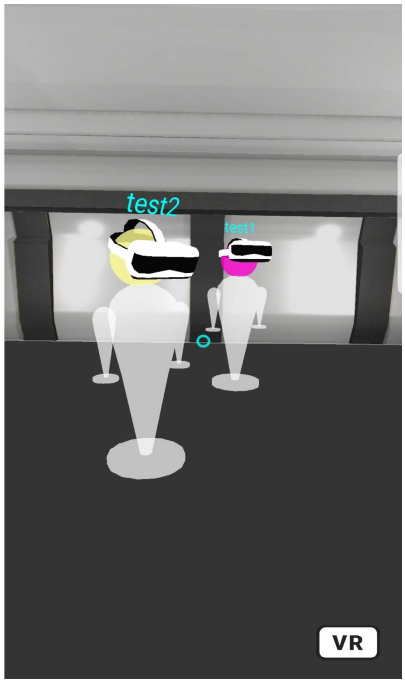
Z týchto záverov vyplýva, že použitá technológia poskytuje veľmi dobré výsledky aj v horších svetelných podmienkach a prestáva byť spoľahlivá až v značne

tmavých priestoroch akým bola situácia C. Za bežného dňa či pri umelom osvetlení miestnosti je sledovanie polohy a mapovanie priestoru vykonávané na veľmi vysokej úrovni. Sledovanie bolo tiež kvalitné aj napriek odrazu jasného svetla od podlahy alebo zrkadlám umiestneným v testovacej miestnosti.

7.3 Testovanie kvality spojenia pri viacerých používateľoch

Posledným testovaním bolo overenie spojenia. Tým sa zisťovalo či a s akým oneskorením vidí používateľ používajúci smartfón ostatných používateľov a naopak. Toto testovanie bolo realizované videohovorom s ostatnými kolegami riešiacimi zvyšné dve časti aplikácie alebo použitím telefónu a zároveň počítača, keďže stránka VR prostredia pre mobilné telefóny podporovala aj použitie počítačov. Týmto testom boli simulované skutočné podmienky pre ktoré bola aplikácia navrhnutá a pri niektorých testoch boli použité smartfón, počítač aj HoloLens súčasne. Pri zapnutej scéne mohol používateľ používajúci telefón vidieť ostatných avatarov a ich pohyb, ako je to možné vidieť na obrázku (Obr. 7.4), aj keď odozva bola niekedy nestála. Vo viacerých pokusoch bol pohyb veľmi plynulý, takmer so žiadnym oneskorením, no občas bol trhavý alebo s oneskorením niekoľko sekúnd. Ostatní používatelia mali veľmi podobné problémy. Tieto problémy boli najpravdepodobnejšie spôsobené preťaženým serverom, keďže vo väčšine prípadov fungovala scéna plynule. V prípade implementácie serveru do LIRKIS laboratória sa potencionálne vyriešia tieto občasné nedostatky. Pred týmto testovaním bol optimalizovaný pohyb avatara prostredníctvom funkcií *bodyAvatar.object3D.translateX*((*object.position.x* - *initialX*) * *multiplier*) a ku nej analogických, ktoré boli použité na zmenu jeho pohybu. Použitím nich sa zamedzilo možnosti avatara prechádzať stenami a inými pevnými objektami. Táto zmena nemala žiaden dopad na výkon a iné aspekty scény, ako bolo overené dodatočným testovaním. Druhou časťou tohto testu bolo zistiť, ako plynulo sú ostatní používatelia schopní vidieť avatara používateľa používajúceho smartfón. Prvým zistením bolo, že pri použití smartfónu nepodporujúceho sledovanie pohybu ostatní používatelia videli avatara rovnako ako jeho používateľ videl ich, teda často plynulý pohyb bez odozvy. Tu sa dalo pohybovať otáčaním telefónu a stláčaním displeja. V prípade použitia smartfónu s podporou sledovania pohybu ostatní používatelia neboli schopní vidieť hlavu ava-

tara jeho používateľa. Analýzou bolo zistené, že je to chýbajúcim komponentom *look-controls*, ktorý bol pre zariadenia podporujúce sledovanie odstránený, pretože spôsoboval nežiadúce skoky a odchýlky v orientácii hlavy pri zapnutom sledovaní pozície. Tento stav bol napravený pridaním komponentu do HTML stránky a jeho vypnutím pri štarte sledovania polohy. Následne bolo možné vidieť celého avatara rovnako ako ostatných a teda vizuálna reprezentácia avatarov bola rovnaká pri použití ľubovoľného telefónu alebo počítača.



Obr. 7.4: Finálna scéna s viacerými pripojenými používateľmi

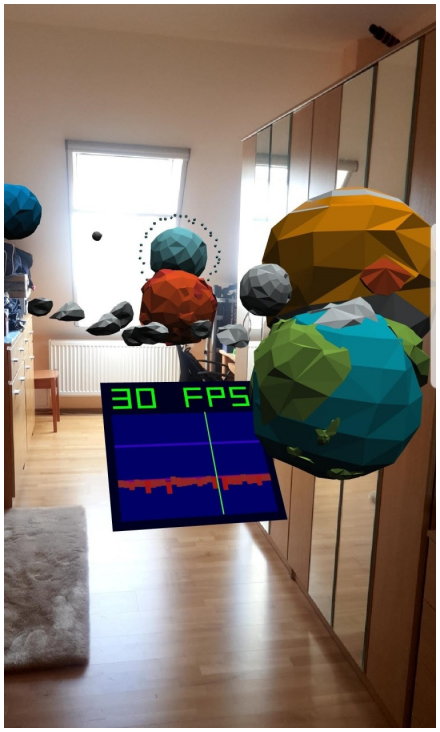
7.4 Porovnanie s podobnými existujúcimi riešeniami

Výsledná aplikácia, spolu s jej vlastnosťami a možnosťami sa s ostatnými v čase písania tejto práce porovnáva zložito. Je to webová VR aplikácia so sledovaním priestoru, čo je technológia, ktorá sa sa v súčasnosti používa takmer vždy pri AR aplikáciách. Takisto majú tieto dva typy aplikácií stúpajúcu tendenciu pri mobilných zariadeniach, avšak väčšinou majú podobu samostatných aplikácií postavených na platforme Google Cardboard, Daydream alebo ARCore namiesto webových stránok. Vzhľadom na to, že táto VR aplikácia obsahuje aj AR prvky, bude porovnaná s VR aj AR aplikáciou. Pre zachovanie čo najväčšej podobnosti budú

tieto aplikácie tiež webové a voľne dostupné.

7.4.1 Porovnanie s AR webovou aplikáciou

AR aplikácia, s ktorou bude riešenie porovnané, je jednoduchá webová aplikácia postavená na WebXR API, ktorá pre renderovanie scény používa knižnicu WebGL [38]. Aplikácia nemá pomenovanie a je jedným z príkladov stránky venujúcej sa WebXR API a zobrazuje možnosti využitia tejto technológie v AR sfére. Cieľom tejto aplikácie je vykreslenie vesmírnych objektov ako planéty, hviezdy či asteroidy do scény zachytenej prostredníctvom fotoaparátu zariadenia. Pre správnu funkcionalitu musí mať používateľ taktiež podporovaný smartfón alebo správne nastavené flagy v prehliadači, inak sa scéna nespustí. Keďže aj táto aplikácia využíva WebXR API, tak na začiatku obsahuje tlačidlo vstupu do AR scény. Je to tak z dôvodu spomínaného v predchádzajúcich kapitolách, kedy je sledovanie okolitého priestoru podmienené používateľským gestom ako je napríklad stlačenie tlačidla. Po jeho stlačení trvá niekoľko sekúnd, kým sa zobrazí výstup z kamery a následne sa doňho vykreslí vesmírna scéna, ako je to znázornené na obrázku 7.5. Táto scéna obsahuje tiež virtuálny objekt, ktorý zobrazuje FPS scény.



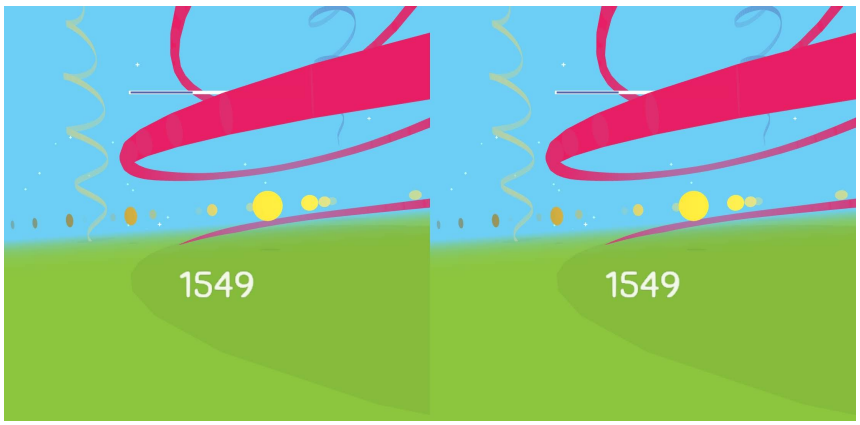
Obr. 7.5: AR aplikácia vykresľujúca vesmírne telesá do okolia používateľa

Po testovaní je možné zhodnotiť, že sa po kvalitatívnej stránke táto webová aplikácia veľmi podobá na vyvinutú aplikáciu LIRKIS G-CVE. Z hľadiska plynulosti mala aplikácia relatívne konštantný frame rate, medzi 25 až 30 FPS. Tento výsledok je veľmi podobný, ako bol dosiahnutý v LIRKIS G-CVE pri použití sledovania polohy telefónu. Z obrázku vyššie je vidieť, že objekty v scéne sú dosť jednoduché a málo detailné. Finálna scéna LIRKIS G-CVE je značne detailnejšia, z čoho opäť vyplýva, že pokles FPS v scéne pri zapnutom sledovaní oproti vypnutému je výsledkom výpočtovej náročnosti sledovania polohy zariadenia v priestore. Táto AR aplikácia taktiež ponúkala veľmi podobnú, až nepostrehnuteľne odlišnú kvalitu sledovania polohy. Taktiež bola testovaná vo viacerých svetelných podmienkach s veľmi podobnými výsledkami ako vyvinutá aplikácia, ktoré boli opísané v predchádzajúcej kapitole. Teda väčšinou bola kvalita sledovania veľmi dobrá až výborná a to aj pri relatívne zhoršených podmienkach. V zlých svetelných podmienkach už virtuálne objekty mizli, alebo sa premiestňovali aj keď bol smartfón statický. Po všetkých testovaných aspektoch sa tak táto aplikácia veľmi podobala na vyvinutý LIRKIS G-CVE z pohľadu použitia telefónu so sledovaním polohy.

7.4.2 Porovnanie s VR webovou aplikáciou

Ďalšou porovnávanou aplikáciou je VR webová hra od spoločnosti Google určená pre telefóny s prehliadačom Chrome a pre platformu Cardboard [39]. Je postavená na knižnici WebGL a Three.js. Ani táto aplikácia nemá meno a je to experiment postavený pre Cardboard, ktorý je možné spustiť v prehliadači na rozdiel od takmer všetkých ostatných samostatných aplikácií pre túto platformu. Po otvorení aplikácia hneď žiada používateľa, aby otočil svoj smartfón do landscape módu. Ak tak spraví, tak sa za pár sekúnd načítajú dáta a používateľ si bude môcť už vo VR móde vybrať zo šiestich miniaplikácií. Aplikáciu LIRKIS G-CVE je možné spustiť aj v bežnom móde, a to v landscape aj portrait režime, čo je výhodou oproti tejto aplikácii. Prostredie obsahuje ukazovateľa nachádzajúceho sa staticky v strede kamery a prostredníctvom neho je možné si vybrať miniaplikáciu. Tie sú tvorené dvoma VR 360-stupňovými videami, 3 hrami a jednou nefunkčnou hrou simulujúcou horskú dráhu. Táto hra nefungovala na žiadnom z troch testovaných prehliadačov. Najzaujímavejšou zo zvyšných je hra, v ktorej sú do priestoru generované mince a úlohou je ich nazbierať čo najviac. Používateľ sa automaticky pohybuje smerom, ktorým sa pozerá. Pohyb je riešený prostredníctvom orientá-

cie telefónu v troch osiach, rovnako ako v aplikácii LIRKIS G-CVE. Tu však používateľ nemôže reagovať so scénou prostredníctvom displeja, ako je to vo finálnej scéne keď je použité zariadenie nepodporujúce sledovanie. Samozrejme tu nie je podporované ani sledovanie zmeny polohy prostredníctvom fotoaparátu, keďže sa jedná o VR aplikáciu. Na obrázku 7.6 je znázornená spomenutá miniaplikácia pre nazbieranie čo najväčšieho počtu mincí.



Obr. 7.6: VR webová aplikácia pre zber mincí

Táto aplikácia neponúkala žiadne výhody oproti finálnej LIRKIS G-CVE scéne okrem vyššej obnovovacej frekvencie, približne 40 FPS, v porovnaní s finálnou scénou pri zapnutom sledovaní polohy. Na druhej strane, finálna scéna ponúka pri vypnutom sledovaní značne vyššie FPS a obraz je viditeľne plynulejší. Testovaná aplikácia takisto neponúka žiaden multiplayer mód a používateľ sa vo všetkých miniaplikáciách nachádza sám. Taktiež tu boli pozorované bugy, ako zobrazenie čiernej stránky stlačením tlačidla späť, keď chce používateľ odísť z prostredia. Túto situáciu je nutné riešiť obnovením stránky. Aplikácia LIRKIS G-CVE tak pre mobilné telefóny ponúka oproti testovanej aplikácií podporu klasického aj VR módu, landscape aj portrét režimu, možnosť pripojenia viacerých používateľov, možnosť sledovania polohy alebo výrazne vyššie FPS pri vypnutom sledovaní.

Tieto dve aplikácie boli vybraté ako tie s najpodobnejšou funkcionalitou finálnej scéne z hľadiska VR a AR funkcií. Na internete bolo nájdených extrémne málo takýchto aplikácií, keďže väčšinu titulov bolo nutné samostatne nainštalovať alebo ich nebolo možné spustiť kvôli zastaralým použitým technológiám. Avšak s príchodom WebXR API by mali webové stránky s podporou AR a VR obsahu pribúdať a časom bude možné porovnať podobnejšie tituly.

8 Záver

Cieľmi tejto práce bolo získanie teoretických poznatkov o možnostiach dnešných inteligentných telefónov a ich senzorov vo VR systémoch, analýza možných spôsobov získavania údajov o polohe týchto zariadení, experimentálna analýza týchto spôsobov, ich ohodnotenie a výber najvhodnejšej technológie, návrh a implementácia výsledného riešenia v podobe klientskej časti viac-používateľského webového VR prostredia pre mobilné telefóny na základe zozbieraných poznatkov, spolupráca s ďalšími dvoma riešiteľmi zvyšných častí prostredia a nakoniec otestovanie a ohodnotenie výsledného riešenia a jeho porovnanie s existujúcimi podobnými produktami. Výsledkom splnenia týchto cieľov je za spoločnej kolaborácie rozšírenie multi-platformového VR prostredia LIRKIS G-CVE určeného pre viacero používateľov pripojených prostredníctvom internetu, ktoré sa pokúša priniesť stabilitu, jednoduchosť, plynulosť či najmodernejšie technológie do veľkého množstva inteligentných zariadení. Tieto a iné vlastnosti prostredia na ňom boli viackrát analyzované, testované a prípadne optimalizované pre poskytnutie čo najlepšieho zážitku a najväčšieho pohltenia do VR sveta. Aplikácia tak spĺňa všetky ciele a kritériá, ktoré sa od nej čakali a je pripravená na nasadenie do LIRKIS laboratória a následné používanie. Avšak aj napriek tomu existujú určité aspekty nad rámec rozsahu tejto práce, ktoré by sa dali odporučiť na pokračovanie vývoja časti určenej pre smartfóny. Prvým aspektom je funkčnosť sledovania len pri použití prehliadača Chrome vo verzii Canary a frameworku A-Frame vo verzii 0.9.2, ktorý vo VR-móde mení orientáciu kamery sám namiesto skriptu sledovania polohy. Preto je nutné na začiatku orientovať telefón správne, aby nevznikol nesúlad medzi jeho orientáciou v skutočnom svete a orientáciou kamery v scéne. Je možné, že nasledujúce verzie frameworku prinesú možnosť kompatibility AR a VR módu, kde bude možné natívne sledovanie polohy cez kameru vo VR prostredí, alebo tieto verzie len umožnia sledovať polohu prostredníctvom vlastných riešení, ako

vytvorený komponent *ar-movement*. Ďalším aspektom potencionálneho zlepšenia zážitku z prostredia je jeho testovanie použitím najnovších high-end mobilných zariadení, ktoré okrem citeľne vyššieho výpočtového a grafického výkonu v porovnaní s testovacím telefónom disponujú aj displejmi s vyššiu obnovovacou frekvenciou. Tie sú teoreticky schopné vykresľovať scénu s obnovovacou frekvenciou 90 Hertzov a vyššou, čím by bol poskytnutý plynulejší pohyb a lepší zážitok minimálne pri vypnutom sledovaní polohy.

Rozšírenie aplikácie LIRKIS G-CVE pre mobilné telefóny ponúka zaujímavý a neštandardný zážitok z virtuálneho sveta kombináciou hlavných vlastností z VR a AR prostredia. Jej vývoj snáď ešte nie je na konci a časom možno ponúkne širšiu podporu, väčšie možnosti alebo ďalšie neobyčajné vlastnosti za pomoci najmodernejších technológií a prejaví tak svoj úplný potenciál.

Zoznam použitej literatúry

[1] "What is VR and how it works?" In: *Thinkmobiles* [*online*]. (máj 2017). URL: https://thinkmobiles.com/blog/what-is-vr/. [cit. 15.4.2019].

[2] Peter Rubin. "The Inside Story of Oculus Rift and How Virtual Reality Became Reality". In: *Wired* [*online*]. (júl 2018). URL: https://www.wired.com/2014/05/oculus-rift-4/. [cit. 18.4.2019].

[3] Ben Lawson. "Motion Sickness Symptomatology and Origins". In: *Human Factors and Ergonomics* [*online*]. (2014), s. 531–599. URL: https://www.academia.edu/23575448/Motion_Sickness_Symptomatology_and_Origins. [cit. 19.4.2019].

[4] David Liu. *Monitoring with head-mounted displays in general anesthesia: a clinical evaluation in the operating room*. Zv. 110. 4. LWW, 2010, s. 1032–1038.

[5] Takashi Shibata. *Head mounted display*. Zv. 23. 1. Apr. 2002, s. 57–64. DOI: https://doi.org/10.1016/S0141-9382(02)00010-0. URL: http://www.sciencedirect.com/science/article/pii/%09%09S0141938202000100.

[6] Ivan E. Sutherland. *A Head-mounted Three Dimensional Display*. AFIPS '68 (Fall, part I). San Francisco, California: ACM, 1968, s. 757–764. DOI: 10.1145/1476589.1476686. URL: http://doi.acm.org/10.1145/1476589.1476686.

[7] *Oculus Go: Features* [*online*]. URL: https://www.youtube.com/watch?v=dcWMuu5j72o. [cit. 5.6.2019].

[8] Stefaan Ternier et al. "ARLearn: Augmented Reality Meets Augmented Virtuality". In: *J. UCS* [*online*]. 18 (2012), s. 2143–2164. URL: https://www.semanticscholar.org/paper/ARLearn%3A-Augmented-Reality-Meets-Augmented-Ternier-Klemke/55a343cec8e289b13a66ff6b5e7887e4940019ab. [cit. 8.5.2019].

[9] Eddie Makuch. "Xbox One, PS4 "too limited" for Oculus Rift, says creator". In: *GameSpot* (nov. 2013). URL: https://www.gamespot.com/articles/xbox-one-ps4-too-limited-for-oculus-rift-says-creator/1100-6416153/. [cit. 15.5.2019].

[10] Anjul Patney et al. *Perceptually-based Foveated Virtual Reality*. SIGGRAPH '16. Anaheim, California: ACM, 2016. ISBN: 978-1-4503-4372-5. DOI: 10.1145/2929464.2929472. URL: http://doi.acm.org/10.1145/2929464.2929472.

[11] Pierre-Yves Laffont et al. *Rectifeye: A Vision-correcting System for Virtual Reality*. SA '16. Macau: ACM, 2016. ISBN: 978-1-4503-4542-2. DOI: 10.1145/2996376.2996382. URL: http://doi.acm.org/10.1145/2996376.2996382.

[12] Yariv Levski. "A Brief Guide to VR Motion Tracking Technology". In: *AppReal RSS* [*online*]. (nov. 2017). URL: https://appreal-vr.com/blog/virtual-reality-motion-tracking-how-it-works/. [cit. 29.4.2019].

[13] Michael Mehling. "Implementation of a Low Cost Marker Based Infrared Light Optical Tracking System". Diz. pr. Institute for Software Technology & Interactive Systems, 2006. URL: http://publik.tuwien.ac.at/files/PubDat_210294.pdf. [cit. 9.5.2019].

[14] "Augmented Reality Demo application (Vuforia, marker tracking)". In: *Innowise Group* [*online*]. (júl 2018). URL: https://www.youtube.com/watch?v=dcWMuu5j72o. [cit. 5.4.2020].

[15] Wei Fang et al. "Real-time motion tracking for mobile augmented/virtual reality using adaptive visual-inertial fusion". In: *Sensors* [*online*]. 17.5 (2017), s. 1037. URL: https://www.mdpi.com/1424-8220/17/5/1037/pdf. [cit. 19.5.2019].

[16] Elisa Ziegler. "Real-time markerless tracking of objects on mobile devices". Diz. pr. Citeseer, 2009. URL: http://www.mobilelifecentre.org/sites/default/files/BA_Ziegler-1_final.pdf. [cit. 28.5.2019].

[17] Aditya Rizki Yudiantika, Selo Sulistyo a Bimo Sunarfri Hantono. "The development of mobile augmented reality quiz visualization methods based on markerless tracking for museum learning application". In: *The International Forum on Strategic Technology* (*IFOST*) [*online*]. 2015. URL: https://www.academia.edu/15515253/The_Development_of_Mobile_Augmented_

Reality_Quiz_Visualization_Methods_Based_on_Markerless_Tracking_for_Museum_Learning_Application. [cit. 29.5.2019].

[18] "Virtual Reality Motion Tracking Technology Has All the Moves". In: *Virtual Reality Society* [*online*]. (jún 2017). URL: https://www.vrs.org.uk/virtual-reality-gear/motion-tracking/. [cit. 5.6.2019].

[19] Weijun Tao et al. "Gait analysis using wearable sensors". In: *Sensors* [*online*]. 12.2 (2012), s. 2255–2283. URL: https://www.mdpi.com/1424-8220/12/2/2255/pdf. [cit. 8.6.2019].

[20] Cenk Acar a Andrei Shkel. *MEMS vibratory gyroscopes: structural approaches to improve robustness*. Springer Science & Business Media, 2008, s. 8. ISBN: 978-0-387-09535-6.

[21] Peter Corke. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. Zv. 118. Springer, 2017, s. 83. ISBN: 978-3-319-54413-7.

[22] "Accelerometer Basics". In: *SparkFun Electronics* [*online*]. (mar. 2017). URL: https://learn.sparkfun.com/tutorials/accelerometer-basics/all. [cit. 6.10.2019].

[23] Hamed Ketabdar, Kamer Ali Yuksel a Mehran Roshandel. "MagiTact: Interaction with Mobile Devices Based on Compass (Magnetic) Sensor". In: *Proceedings of the 15th International Conference on Intelligent User Interfaces*. IUI '10. Hong Kong, China: ACM, 2010, s. 413–414. ISBN: 978-1-60558-515-4. DOI: 10.1145/1719970.1720048. URL: http://doi.acm.org/10.1145/1719970.1720048. [cit. 10.10.2019].

[24] "SparkFun 9DoF IMU (ICM-20948) Breakout Hookup Guide". In: *SparkFun* [*online*]. (jún 2019). URL: https://learn.sparkfun.com/tutorials/sparkfun-9dof-imu-icm-20948-breakout-hookup-guide/all. [cit. 31.3.2020].

[25] Dom Barnard. "Degrees of Freedom (DoF): 3-DoF vs 6-DoF for VR Headset Selection". In: *VirtualSpeech* [*online*]. (máj 2019). URL: https://virtualspeech.com/blog/degrees-of-freedom-vr. [cit. 9.9.2019].

[26] Lisa Eadicicco. "Google's Virtual Reality Headset is About to Get Better". In: *Time* [*online*]. (jan. 2016). URL: https://time.com/4180621/google-cardboard-headset-audio/. [cit. 25.10.2019].

[27] Jeremy Clyde. "On the road for VR: Microsoft HoloLens at Build 2015, San Francisco". In: *Doc* [*online*]. (jún 2015). URL: http://doc-ok.org/?p=1223. [cit. 9.4.2019].

[28] Raymond Wong. "Microsoft Hololens 2 ushers in the next generation of augmented reality". In: *Mashable* [*online*]. (feb. 2019). URL: https://mashable.com/article/microsoft-hololens-2-mwc-2019/?europe=true. [cit. 9.4.2019].

[29] Bc. Kristína Matiková. "Kolaborácia vo virtuálnej realite: vstupno-výstupná časť pre MS HoloLens". (V tlači). Dipl. pr. Technická univerzita v Košiciach, máj 2020.

[30] Jessica Conditt. "Mozilla makes it easy to create VR websites with 'A-Frame'". In: *Engadget* [*online*]. (dec. 2015). URL: https://www.engadget.com/2015-12-17-mozilla-makes-it-easy-to-create-vr-websites-with-a-frame.html. [cit. 23.11.2019].

[31] "What's an Entity System?" In: *Entity Systems Wiki* [*online*]. (nov. 2014). URL: http://entity-systems.wikidot.com/. [cit. 23.11.2019].

[32] Anil Dash. "What is Glitch?" In: *Medium* [*online*]. (nov. 2019). URL: https://medium.com/glitch/what-is-glitch-90cd75e40277. [cit. 15.1.2020].

[33] "Motion Sensors Explainer". In: *The World Wide Web Consortium* [*online*]. (aug. 2017). URL: https://www.w3.org/TR/motion-sensors/. [cit. 28.10.2019].

[34] "Building an augmented reality (AR) application using the WebXR Device API". In: *Google* [*online*]. (2019). URL: https://codelabs.developers.google.com/codelabs/ar-with-webxr/#0. [cit. 8.1.2020].

[35] Marián Hudák, Štefan Korečko a Branislav Sobota. "Enhancing Team Interaction and Cross-platform Access in Web-based Collaborative Virtual Environments". In: *Proceedings of 2019 IEEE 15th International Scientific Conference on Informatics* (2019), s. 160–164. [cit. 9.2.2020].

[36] Bc. Michal Ivan. "Kolaborácia vo virtuálnej realite: komunikačná a riadiaca časť". (V tlači). Dipl. pr. Technická univerzita v Košiciach, máj 2020.

[37] "Async function". In: *MDN Web Docs* [*online*]. (apr. 2020). URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function. [cit. 10.4.2020].

[38] *WebXR Samples*. 2018. URL: https://immersive-web.github.io/webxr-samples/.

[39] *Chrome Experiments for Virtual Reality*. Máj 2015. URL: https://vr.chromeexperiments.com/.

Zoznam príloh

Príloha A CD médium — obsahuje Diplomovú prácu, Používateľskú a Systémovú príručku v digitálnej forme a zdrojový kód aplikácie.

Príloha B Používateľská príručka

Príloha C Systémová príručka