F3

Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Systém rozšířené reality pro vizualizaci pozice a stavu členů týmů IZS

Bc. Zdeněk Havelka

Leden 2024
Vedoucí práce: doc. Ing. Miroslav Bureš, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Havelka** Jméno: **Zdeněk** Osobní číslo: **475392**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Systém rozšířené reality pro vizualizaci pozice a stavu členů týmů IZS

Název diplomové práce anglicky:

Augmented Reality System for Visualization of Position and Status of a First Responders Team Members

Pokyny pro vypracování:

Navrhněte a implementujte systém rozšířené reality pro vizualizaci pozice a stavu členů týmu IZS. Systém bude řešený jako aplikace do vybraných brýlí pro rozšířenou realitu a stavové informace o členech týmu bude přebírat ze serverové části systému (bude dodáno školitelem). Aplikace bude zobrazovat směr, vzdálenost a vybrané stavové informace jednotlivých členů týmu konkrétnímu členovi. V práci se zaměřte na problematiku detekce natočení obličeje člena týmu používajícího vyvíjený systém. Navržený a implementovaný systém otestujte sadou vhodných uživatelských testů v reálném terénu.

Seznam doporučené literatury:

Merino, L., Schwarzl, M., Kraus, M., Sedlmair, M., Schmalstieg, D., & Weiskopf, D. (2020, November). Evaluating mixed and augmented reality: A systematic literature review (2009-2019). In 2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) (pp. 438-451). IEEE.
Van den Oever, F., Fjeld, M., & Sætrevik, B. (2023). A Systematic Literature Review of Augmented Reality for Maritime Collaboration. International Journal of Human–Computer Interaction, 1-16.
Agrawal, A., & Cleland-Huang, J. (2021). RescueAR: Augmented Reality Supported Collaboration for UAV Driven Emergency Response Systems. arXiv preprint arXiv:2110.00180.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Miroslav Bureš, Ph.D. laboratoř inteligentního testování systémů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **22.09.2023** Termín odevzdání diplomové práce: **09.01.2024**

Platnost zadání diplomové práce: **16.02.2025**

doc. Ing. Miroslav Bureš, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Poděkování / Prohlášení

Chtěl bych poděkovat vedoucímu doc. Ing. Miroslavu Burešovi, Ph.D. za cenné rady a vedení projektu, své rodině a přítelkyni za podporu během studia.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 4. 1. 2024

.....

Abstrakt / Abstract

Tato diplomová práce se zaměřuje na využití technologií rozšířené reality za účelem zvýšení operačních schopností pracovníků integrovaného záchranného systému v kritických situacích. Hlavním cílem práce je navrhnout a implementovat prototyp systému rozšířené reality (AR) pro zobrazování pozic a stavů členů integrovaného záchranného systému. Informace o pozici a stavu budou vizualizovány pomocí směrových ukazatelů, objektů rozšířené reality zasazených do reálného světa a mapy. Součástí práce je představení technologie rozšířené reality a nástrojů pro práci s AR, návrh systému, včetně lo-fi prototypu uživatelského rozhraní, popis implementace jednotlivých částí a závěrečného testování.

Klíčová slova: rozšířená realita; krizové situace; záchranné složky; projekce souřadnic

This thesis focuses on the use of augmented reality technologies to enhance operational capabilities of first responders in emergency situations. The main objective of the thesis is to design and implement a prototype of augmented reality system for displaying positions and statuses of first responders. Information about position and status will be visualized using directional indicators, augmented reality objects embedded in the real world and a map. The thesis includes an introduction of augmented reality technology and tools for working with AR, system design, including a lo-fi prototype of user interface, description of the implementation of individual parts and final testing.

Keywords: augmented reality; emergency situations; first responders; coordinate projection

Title translation: Augmented reality system for visualization of position and status of a first responders team members

Obsah /

1 Úvod **1**
1.1 Motivace . . . 1
1.2 Cíl projektu . . . 1
1.3 Struktura práce . . . 2

2 Rozšířená realita **3**
2.1 Sledování a rozpoznávání . . . 3
2.2 Využití . . . 3
2.2.1 Využití v krizových situacích . . . 3
2.3 Zařízení . . . 4
2.3.1 Displeje . . . 4
2.3.2 Vstupní zařízení . . . 5
2.4 Technologie a nástroje . . . 6
2.4.1 Unity . . . 6
2.4.2 ARCore . . . 7
2.4.3 ARKit . . . 8
2.4.4 Vuforia . . . 8

3 Podobné projekty **10**
3.1 RESPOND-A . . . 10
3.2 FASTER . . . 11

4 Analýza **12**
4.1 FlexiGuard . . . 12
4.1.1 Senzorická jednotka . . . 13
4.1.2 Snímací jednotka . . . 13
4.1.3 Vizualizační jednotka . . . 13
4.2 Server . . . 13
4.2.1 RESTové HTTP rozhraní . . . 13
4.2.2 WebSocket rozhraní . . . 14
4.3 Doménový model . . . 15
4.4 Požadavky . . . 15
4.4.1 Funkční požadavky . . . 15
4.4.2 Nefunkční požadavky . . . 16
4.5 Uživatelé systému . . . 16
4.6 GPS . . . 16
4.6.1 WGS 84 . . . 16
4.7 Projekce geografických souřadnic . . . 16
4.7.1 Web Merkator zobrazení . . . 17
4.7.2 Bowringova rovnice . . . 17
4.8 Harvesinova rovnice . . . 18
4.9 Zvolené technologie . . . 18
4.9.1 Unity . . . 18
4.9.2 ARCore . . . 18
4.9.3 Rozhraní statických map . . . 18
4.9.4 Verzovací nástroje . . . 19

5 Návrh **20**
5.1 Návrh uživatelského rozhraní . . . 20
5.1.1 Směrové ukazatele . . . 20
5.1.2 Ovládací rozhraní . . . 21
5.1.3 3D objekt . . . 23
5.2 Komponenty . . . 24
5.3 Architektura . . . 25
5.3.1 Model View Controller . . . 26
5.3.2 Model View Presenter . . . 26
5.3.3 Porovnání . . . 26

6 Implementace **27**
6.1 Podpora Android . . . 27
6.2 Zpracování dat . . . 27
6.3 Lokace . . . 28
6.4 Zobrazení . . . 28
6.4.1 Směrové ukazatele . . . 28
6.4.2 3D objekt . . . 30
6.4.3 Zobrazení jednotek . . . 31
6.5 Okluze . . . 31
6.6 Mapa . . . 33
6.7 Uživatelské rozhraní . . . 33
6.7.1 Responzivní design . . . 34
6.8 Splnění požadavků . . . 34

7 Testování **36**
7.1 Scénáře . . . 36
7.2 Testování lo-fi prototypu . . . 36
7.2.1 Výběr metody . . . 37
7.2.2 Výsledky . . . 37
7.3 Jednotkové testy . . . 38
7.3.1 Projekce . . . 38
7.3.2 Konverze dat . . . 39
7.4 Uživatelské testování . . . 40
7.4.1 Účastníci . . . 40
7.4.2 Průběh . . . 41
7.4.3 Výsledky . . . 43
7.5 Finální úpravy . . . 44
7.5.1 Výkon . . . 44
7.5.2 Nastavení . . . 45
7.5.3 Mapa . . . 46
7.5.4 Uživatelské rozhraní . . . 47

8 Závěr **49**

Literatura **50**

A Zkratky **53**

/ Obrázky

2.1 Brýle Microsoft HoloLens 2 5
2.2 XReal Light Ovladač 6
2.3 Unity Editor 7
3.1 RESPOND-A rozhraní 11
4.1 Flexiguard systém 12
4.5 Doménový model 15
4.6 Popularita verzovacích systémů 19
5.1 Směrové ukazatele 21
5.2 List misí 22
5.3 Nastavení a mapa 23
5.4 3D značka 24
5.5 Komponentový diagram 25
6.1 Rozhraní ConnectionHandler .. 27
6.2 Barevný gradient pro zobrazení stavu 28
6.3 Objekt směrového ukazatele ... 28
6.4 Projekce směrových ukazatelů . 29
6.5 Testování viditelnosti 3D objektu 30
6.6 Transformace souřadnic 31
6.7 3D Ukazatel v Unity 31
6.8 Ilustrace okluze 32
6.9 Implementované rozhraní - hlavní obrazovka, mapa 34
6.10 Implementované rozhraní - list misí, nastavení 34
7.3 Snímky z testování 43
7.5 Upravené možnosti nastavení .. 46
7.6 Upravené zobrazení mapy 47
7.7 Dialog pro chybové hlášky 48

Kapitola 1
Úvod

Diplomová práce se zabývá využitím systémů rozšířené reality členy integrovaných záchranných složek. Dále pak popisuje návrh a implementaci systému pro zobrazování pozic a stavů jednotlivých členů.

1.1 Motivace

V době, která se nejen v Evropě vyznačuje stále častějšími přírodními i lidmi způsobenými katastrofami [1], nebyla nikdy předtím nutnost pokročilých technologických řešení v oblasti reakce na mimořádné události tak naléhavá. Evropská agentura pro životní prostředí zdůrazňuje kritický trend. Tím je výrazný nárůst počtu událostí, které vedou k rozsáhlým ztrátám na lidských životech, hospodářství a životním prostředí - od ničivých následků přírodních katastrof, jako jsou povodně a hurikány, až po chaos způsobený městskými katastrofami - dopravními nehodami, nebo prostorovým kolapsem budov [1]. Tento trend v oblasti mimořádných událostí představuje bezprecedentní výzvu pro pracovníky integrovaných záchranných složek, kteří jsou v těchto krizových situacích často první linií.

V chaotickém prostředí událostí hromadného neštěstí a katastrof je schopnost těchto pracovníků první pomoci efektivně komunikovat a činit rychlá a informovaná rozhodnutí zásadní. Složitost zvládání mimořádných událostí v těchto scénářích je umocněna potřebou přesně třídit a rozdělovat omezené zdroje pod časovým tlakem. Hlavním cílem v těchto rizikových situacích je primárně zajistit účinnou a rychlou evakuaci, ošetření zraněných a minimalizovat škody na zdraví, či majetku [2].

Kromě toho je nesmírně důležitá bezpečnost a efektivita samotných záchranářů, kteří při plnění svých povinností běžně čelí nebezpečným podmínkám. Někteří mohou, ve snaze pomoci druhým, ignorovat příznaky únavy, vyčerpání a nevědomky vystavit své zdraví ještě většímu riziku. Právě nasazením různorodých technologií, které poskytují zprostředkované informace jednotlivým členům okamžitě, můžeme minimalizovat toto riziko újmy na zdraví, a zlepšit efektivitu jednotlivců, či celých skupin [3].

1.2 Cíl projektu

Cílem projektu je seznámit se s danou problematikou, rozšířenou realitou, navrhnout a implementovat prototyp aplikace rozšířené reality. Aplikace zpracuje a zobrazí data o poloze a stavu jednotlivých členů mise integrovaných záchranných složek. Zobrazovaná data jsou poskytována již vybudovanou serverovou aplikací, která není součástí této práce.

Původním záměrem bylo aplikaci vyvinout na zařízení XReal, chytré brýle pro rozšířenou realitu. Ty představují komerční, cenově dostupnou variantu možného použitého zařízení. XReal brýle se bohužel nepodařilo obstarat, proto došlo k úpravě zadání a prototyp je implementovaný pro platformu Android. Implementovaná aplikace bude sloužit

jako *proof of concept* pro uživatelské testování se záměrem využití podobného systému nasazeného do brýlí rozšířené reality, bez nutnosti využití drahého dedikovaného hardware.

1.3 Struktura práce

Práce je rozdělena do osmi kapitol. V druhé kapitole bude popsána technologie rozšířené reality a vybrané nástroje. V třetí kapitole budou představeny podobné projekty, zabývající se využitím rozšířené reality pro zlepšení práce záchranářů v krizových situacích. V následující čtvrté kapitole bude představena analytická část práce. Budou popsány požadavky implementovaného systému, serverová část a vybrané technologie. Následující kapitola se bude věnovat návrhu uživatelského rozhraní, architektuře a komponentám systému. Šestá kapitola popisuje implementaci a řešení jednotlivých částí systému. V předposlední kapitole bude popsáno testování výsledné aplikace pomocí metody kognitivního průchodu, jednotkových testů a průběh uživatelského testování.

Kapitola 2
Rozšířená realita

Technologie rozšířené reality, také označována jako AR, z anglického Augmented Reality, popisuje vnímání reálného světa, obohaceného o generované digitální prvky a často nabízí jistou míru interakce. Přítomnost reality a prvků reálného světa tak odlišuje rozšířenou realitu od reality virtuální (VR, Virtual reality), která uživatele vtáhne do světa plně syntetického [4]. Byť většina AR technologií cílí především na zrakový vjem, který obohacuje o generované 2D nebo 3D informace, nemusí tomu být tak vždy a technologie může cílit i na jiné smysly [5].

2.1 Sledování a rozpoznávání

Sledování (angl. tracking) a rozpoznávání (angl. recognition) jsou klíčové aspekty AR systémů [4], jež cílí na co nejpřesnější prolnutí digitálních prvků do reálného světa za účelem co nejuvěřitelnějšího uživatelského prožitku.

Sledování je prováděno za použití nejrůznějších senzorů jako např. digitální kamery, nebo jiných optických senzorů, kompasu, GPS, akcelerometru atp. Ty slouží především k detekci pohybu a polohy uživatele v reálném prostředí [6]. Strojové učení a algoritmy počítačového vidění jsou hojně využívány k rozpoznávání objektů, značek i obrazů. Toto může být využito jako forma interakce v podobě rozpoznávání gest rukou, ale také jako možnost využití externích značek umístěných například v interiérech objektů, jež AR systém rozpozná a určí, kam může digitální prvky ukotvit[7].

2.2 Využití

V poslední letech nabývá technologie rozšířené reality na popularitě a zájmu o její využití v nejrůznějších odvětvích [8]. Pro širokou veřejnost je pravděpodobně nejznámějším AR titulem mobilní hra PokemonGo. Ta do reálného světa snímaného mobilní kamerou zasazuje nejrůznější charaktery světa Pokemon a umožňuje uživateli s danými virtuálními objekty interagovat [9]. AR je také známá aplikací nejrůznějších obličejových filtrů do videa, či fotografie na sociálních sítích jako je například Instagram [7].

Kromě zábavního průmyslu můžeme AR systémy nalézt také jako pomůcku ve školství, v oblastech reklamy a průmyslu, nebo v medicíně [5].

2.2.1 Využití v krizových situacích

V krizových situacích a při hromadných neštěstích je potřeba urychleně přijmout opatření k zajištění bezpečnosti postižených osob. AR systémy by zde mohly představovat nástroj pro zrychlení komunikace mezi jednotlivými záchranáři, zobrazovat polohy bodů zájmu, či například analyzovat prostředí [7]. V situacích, kdy dojde k požáru, můžou zobrazené body zájmu představovat například hydranty v okolí a analýza prostředí může zahrnovat výstup termální kamery.

Jako řešení se nabízejí chytré brýle v kombinaci s nositelnou elektronikou pro nasazené záchranáře, nebo také pro vedoucí osoby v kontrolním středisku, kde je možné všechny sbírané hodnoty vyhodnocovat a na jejich základě vydávat informovaná rozhodnutí.

Návrhem právě takového systému se zabývá tato diplomová práce. Zobrazení pozic a stavů členů integrovaného záchranného systému při zásahu v krizové situaci může pomoci zlepšit situační povědomí jednotlivých členů. Zobrazení stavu pak indikuje vyčerpanost, či fyzické zranění a urychluje navigaci k takovému jednotlivci.

2.3 Zařízení

Zařízení pro rozšířenou realitu typicky rozdělujeme do tří kategorií - displeje, vstupní zařízení a výpočetní jednotky [5]. Nejrozšířenějším zařízením je chytrý telefon. Chytrý mobilní telefon je schopen obsáhnout všechny kategorie. Obsahuje displej, který zároveň reaguje na uživatelův dotyk a slouží jako vstupní zařízení, procesor jako výpočetní jednotku a řadu dalších senzorů. Téměř všechny mobilní aplikace sociálních sítí využívají jistou formu rozšířené reality v podobě různých efektů a pro pořízení a nahrání fotografie.

2.3.1 Displeje

Displej slouží k zobrazování vykreslených objektů. Většinou rozlišujeme 3 druhy - HMD (Head mounted display), ruční displeje či prostorové. HMD jsou zařízení, která je možné nosit na hlavě, nebo je možné je připevnit k přilbě. Do této skupiny patří například chytré brýle. Ve skupině ručních displejů nalezneme mobilní telefony a tablety. Prostorové displeje jsou pak projektory či zařízení schopná projekce přímo na fyzické objekty. Toho je využíváno především v muzeích [5].

HoloLens 2

HoloLens 2 jsou chytré brýle od společnosti Microsoft viz obrázek 2.1. Brýle částečně připomínají větší VR headsety. Místo klasických nožiček obepínají celou hlavu a váží zhruba 560g. Jsou vybaveny 2K displeji, až 6 kamerami, gyroskopem, magnetometrem, akcelerometrem. Nabízeny jsou ve třech variantách - standardní, industriální edice, určená do přísněji regulovaných prostředí, či edice integrovatelná na helmu, určená na stavbu a podobná prostředí [10]. Tyto brýle jsou pravděpodobně pro použití v terénu při krizových situacích nevhodné. Zejména kvůli své velikosti. Podobné zařízení by však mohlo být v kombinaci s různými displeji ideální v kontrolním středisku.

Obrázek 2.1. Microsoft HoloLens 2 - převzato z [10]

XREAL brýle

XREAL brýle jsou zástupcem AR nositelné elektroniky. Narozdíl od VR zástupců připomínají obyčejné plastové sluneční brýle. Aktuálně jsou prodávány 2 modely - XREAL Light a XREAL Air. Jsou vybaveny dvěma OLED displeji o rozlišení 1920x1080 pixel, mikrofony, reproduktory, RGB i Grayscale kamerami. Modely se liší především kamerou a váhou, která u modelu Light činí 106g a u modelu Air zhruba 77g. Brýle jsou podporovány operačním systémem Android a mohou být doplněny o výpočetní jednotku nebo XREAL Light ovladač, zařízení spárovatelné s telefonem pomocí bluetooth umožňující interagovat s prostředím. Výdrž baterie je uváděna na zhruba dvě a půl hodiny [11].

2.3.2 Vstupní zařízení

Vstupní zařízení jsou určena primárně k interakci. Toho může být dosaženo například pomocí speciálních rukavic, ukazovátek nebo dotykových displejů [5]. Interakce se systémem rozšířené reality nemusí být dosaženo pouze pomocí fyzického zařízení. Některé moderní platformy jsou schopné rozpoznat gesta uživatelových rukou. NRSDK, softwarový balíček, pro výše zmíněné XREAL brýle umí rozpoznat gesta sledováním 22 klíčových bodů na lidské ruce [11]. Příkladem vstupního zařízení je například ovladač XReal Light viz obrázek 2.2.

Obrázek 2.2. XReal Light ovladač - převzato z [11]

2.4 Technologie a nástroje

Technologií, či knihoven pro vývoj AR systémů existuje celá řada. Podle průzkumu ACM Computing Surveys jsou platformy iOS a Android konzistentně podporovány [7]. V této sekci se zaměřím na nástroj Unity a některá SDK pro rozšířenou realitu na mobilních zařízeních a jejich práci s geografickými daty.

Podle počtu implementované funkcionality těmto platformám dominuje ARKit (iOS) od společnosti Apple a ARCore (Android) vyvíjený společností Google [7]. Tyto balíčky pro softwarový vývoj (SDK) a mnoho dalších také podporují herní engine Unity.

2.4.1 Unity

Unity je multiplatformní herní engine určený pro tvorbu 2D a 3D her od Unity Technologies. PocketGamer uvádí, že až 71 ze 100 největších mobilních herních titulů je vyvinuto za pomoci alespoň jednoho Unity SDK [1]. Svoji popularitu si však získal i na poli rozšířené a virtuální reality.

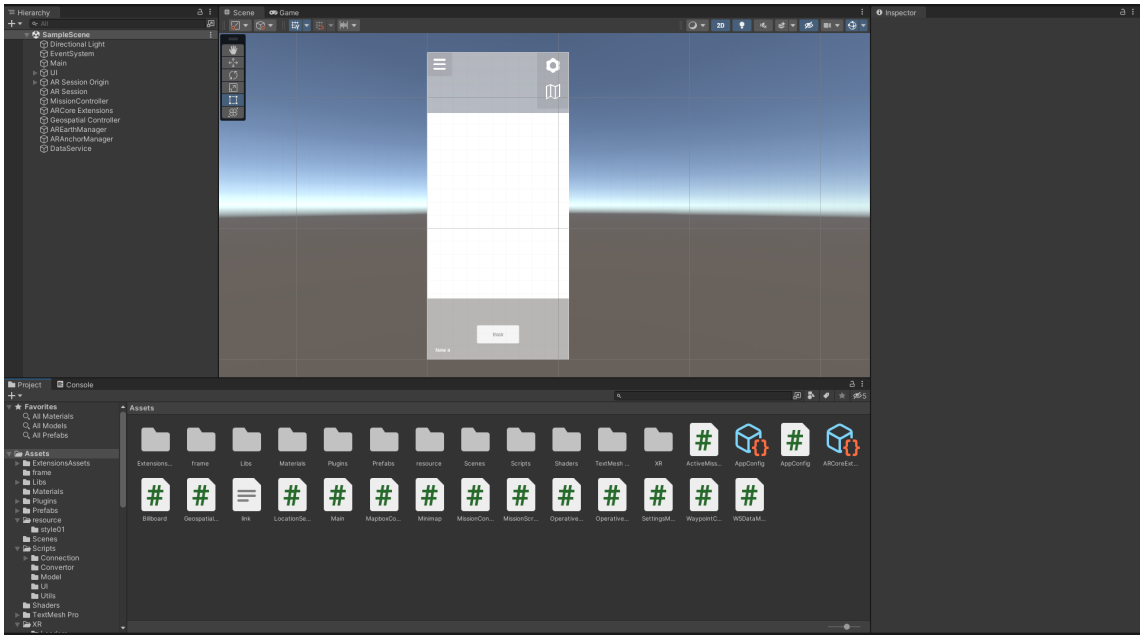
Unity přichází se svým editorem viz obrázek 2.3 a velkým množstvím již zabudovaných nástrojů, nebo vlastním obchodem, kde je možné získat nejrůznější nástroje, grafické modely, či softwarové balíčky.

Základními stavebními kameny v Unity jsou scéna a herní objekty. Scéna, podobně jako scéna divadelní, obsahuje vše, co uživatel aktuálně vidí. Herní objekt je poté prakticky cokoli, co se ve scéně nachází. Na herní objekt jsou napojené komponenty, které mohou měnit jeho funkcionalitu, podobu, nebo chování. Skripty, které upravují chování jednotlivých objektů, jsou typicky psané v programovacím jazyce C#.

AR Foundation

AR Foundation je framework Unity, jenž pomáhá vytvářet multiplatformní AR aplikace s podporou mnoha funkcionalit jako je rozpoznávání ploch, objektů, obličejů, raycasting a další. Výše zmíněný ARKit a ARCore rozšiřují tyto funkcionality s podporou konkrétní platformy.

[1] https://www.pocketgamer.biz/news/82514/71-of-the-biggest-mobile-games-use-a-unity-sdk/



Obrázek 2.3. Unity editor

2.4.2 ARCore

ARCore je softwarový balíček (SDK) od společnosti Google pro vývoj aplikací rozšířené reality na platformě Android. Umožňuje mobilním zařízením vnímat fyzický svět a komunikovat s ním prostřednictvím pokročilých funkcí rozšířené reality. V jádru ARCore plní dvě základní funkce: sleduje polohu mobilního zařízení při jeho pohybu a vytváří představu o reálném světě kolem něj [12]. Tato technika se nazývá SLAM, tedy simultánní lokalizace a mapování (angl. simultaneous localization and mapping). Mezi klíčové funkcionality patří:

- **Sledování pohybu:**
  Sledování pohybu je docíleno algoritmem VIO (visual-inertial odometry), tedy kombinací identifikací obrazu z kamery telefonu a pohybovými senzory
- **Porozumění prostředí:**
  Tato funkcionalita představuje takzvaný meshing jako formu rozpoznávání povrchu, jeho tvarů a velikosti. Takovým povrchem může být například deska psacího stolu, podlaha nebo zeď.
- **Odhad světelných podmínek:**
  Odhad světelných podmínek pomáhá, aby digitální prvky lépe zapadly do scény rozšířené reality, a to díky jejich možnému nasvícení, či vrhání stínů.

ARCore disponuje zajímavým systémem VPS pro zpřesnění uživatelovy polohy a rozhraním pro umísťování objektů v závislosti na jejich geografických souřadnicích.

- **Geoprostorové rozhraní (Geospatial API):**
  Geoprostorové rozhraní je novým doplňkem ARCore, které primárně umožňuje přesné připojení digitálního obsahu k místům popsaným jejich geografickými souřadnicemi. K tomu slouží takzvané geoprostorové kotvy, které můžeme využívat ve třech variantách [13]. Kotvy jsou jakýsi fixní bod v prostoru, ke kterému je možné uchytit digitální prvky. Toto rozhraní využívá systém VPS.
- **VPS:**

VPS je systém (angl. Visual Positioning System, vizuální polohovací systém) pro zpřesnění polohy získané pomocí GPS zejména v městkých prostředích. VPS využívá hluboké učení, konkrétně hluboké neuronové sítě, k identifikaci a popisu charakteristických rysů na snímcích Street View. Tyto části jsou poté zkombinovány napříč desítkami miliónů obrazů k vytvoření 3D prostředí. Celý VPS systém se skládá z trilionů takových bodů. Obraz z uživatelovy kamery je poté zpracován neuronovou sítí, která se jej snaží namapovat do vytvořeného prostředí a pomocí algoritmů počítačového vidění je určena uživatelova poloha a otočení[14]. Limitujícím faktorem je předpoklad, že aktuální poloha je dobře zachycená pomocí Google Street View.

- **WGS84 kotvy:**
  Tyto kotvy je možné vytvořit na místě poskytnutých geografických souřadnic - zeměpisné šířky, výšky a nadmořské výšky podle WGS84 souřadnicového systému. Ten bude popsán v následující kapitole.
- **Terénní kotvy (Terrain anchors):**
  Na rozdíl od WGS84 kotev, terénní kotvy můžeme vytvořit na místě popsaném pouze zeměpisnou šířku a výškou. Informace o nadmořské výšce je poté získána pomocí systému VPS. Specifikovat můžeme pouze odsazení, pokud nechceme aby se kotva nacházela na úrovni terénu.
- **Střešní kotvy (Rooftop anchors):**
  Fungují podobně jako terénové kotvy s rozdílem, že nadmořská výška vytvořené kotvy odpovídá úrovni střechy na daných souřadnicích. Opět, jako u terénových kotev, můžeme odsadit o danou výšku nad úrovní střechy.

ARCore je opensource projekt a je zdarma k využití [7].

2.4.3 ARKit

ARKit je SDK od společnosti Apple pro platformu iOS. I zde je implementována funkcionalita pro sledování pohybu a porozumění prostředí. V některých ohledech předčí svůj protějšek ARCore, a to zejména v možnosti sledování rukou, rozpoznávání 2D a 3D objektů (mimo plochy) a možnosti využití LiDAR senzoru [7].

- **ARGeoAnchor:**
  ARKit geografické kotvy jsou velice podobné ARCore geoprostorovým kotvám. I zde je možné označit geografický bod pomocí jeho souřadnic a ke zpřesnění aktuální polohy uživatele je použit podobný systém jako VPS. ARKit stáhne sadu obrazů zachycujících prostředí na daných souřadnicích a porovná je se snímky z kamery. Tato obrazová geolokalizace nemusí být dostupná kdekoli [15].
  ARKit je k dispozici jako balíček pro engine Unity a je zdarma.

2.4.4 Vuforia

Softwarový balíček pro tvorbu aplikací rozšířené reality od společnosti PTC Inc. s názvem Vuforia Engine patří mezi nejpopulárnější řešení na trhu [16] a podporuje jak Android tak iOS. Využívání Vuforia Engine je možné v limitované podobě zdarma, nebo případně se všemi dostupnými funkcionalitami za měsíční poplatek.

Toto SDK se pyšní především svým cloudovým řešením rozpoznávání obrazu, schopného sledovat a rozpoznávat miliony různých obrazů. Kromě obrazu umožňuje také rozpoznávání vlastních 3D objektů na základě CAD modelu, či skenu, rozpoznávání cylindrických nebo kuželovitých objektů a takzvaných VuMarks. VuMarks jsou speciální čárové kódy, do kterých je možné zakódovat informace jako identifikátor objektu, obrázek, logo či webovou adresu.

Podle oficiální dokumentace Vuforia neumožňuje práci s geografickými souřadnicemi jako ARCore, nebo ARKit.

Kapitola 3
Podobné projekty

V této kapitole budou představeny podobné projekty, které využívají technologie rozšířené reality za účelem zlepšení a zefektivnění práce záchranných týmů.

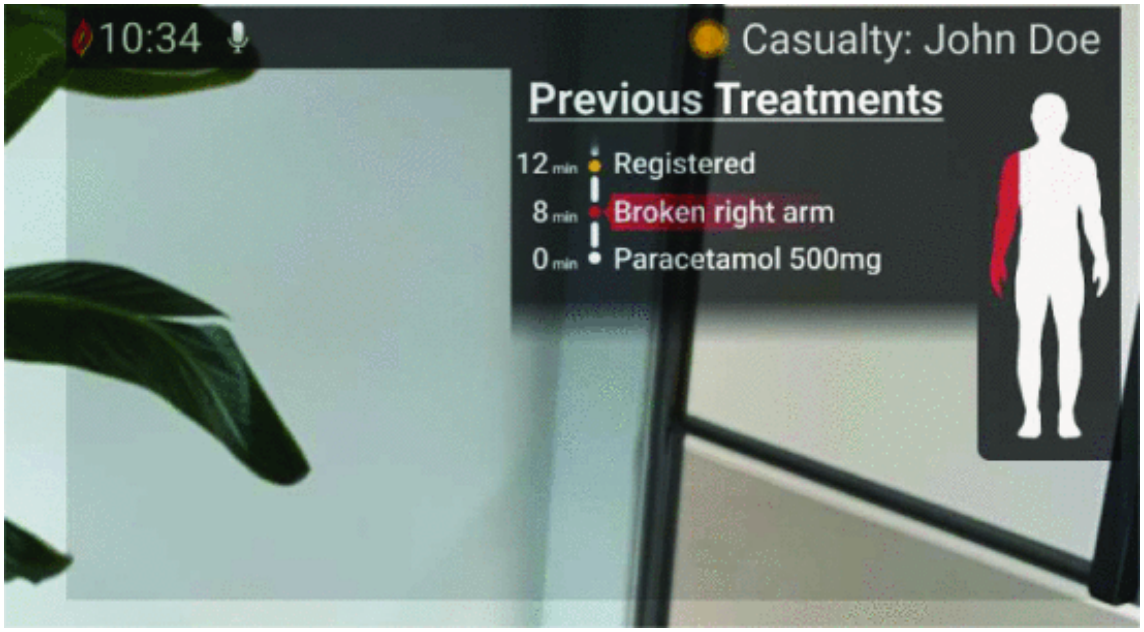
3.1 RESPOND-A

RESPOND-A je evropský projekt zaměřený na zlepšení situačního povědomí a rychlejší, či efektivnější rozhodování. Navržená platforma využívá rozšířené reality, mixované reality a sady senzorů v nositelné elektronice. RESPOND-A také zamýšlí využití dronů a robotů.

Svých cílů projekt dosahuje v několika rovinách. První je použití v terénu. Zde dochází k monitorování stavu záchranářů pomocí nositelné elektroniky a k následné vizualizaci v brýlích Vuzix M5000. Kromě monitorování stavu záchranářů nabízí projekt také sledování lékařské historie raněných pomocí digitálního identifikačního tagu. Rozhraní zobrazující lékařskou historii raněného je zobrazeno na obrázku 3.1. Ten je na raněného umístěn v podobě náramku s NFC čipem. Pro zlepšení vizuálních možností vyžívá RESPOND-A termální kamery.

Druhou rovinou je použití v řídících centrech. Zde jsou využity brýle HoloLens 2 od společnosti Microsoft. Systém zobrazuje podobné informace jako v terénu. Ty jsou však doplněné například o zobrazení pozic na mapách. Systém v HoloLens brýlích dovoluje multimodální interakci, tedy ovládat systém pomocí hlasu a gest rukou, či očí.

Poslední rovinou je komunikace mezi záchranáři a řídícím centrem. Ta umožní přenést záběry z terénu přímo do zařízení člena v řídícím centru a v obráceném případě umožní nasdílet například živé záběry z dronu do brýlí člena v terénu [3].



Obrázek 3.1. Rozhraní s lékařskou historii ošetřovaného [3]

3.2 FASTER

Evropský projekt FASTER, představený v roce 2020, je téměř ve všech bodech totožný s předchozím projektem RESPOND-A. FASTER představuje vývoj prototypu chytré textilie pro monitoring tělesných funkcí jako je teplota, či tepová frekvence. Tyto informace vizualizuje v AR brýlích HoloLens 2 s podporou multimodálních vstupů. I zde se počítá s nasazením dronů nebo robotů. Zajímavostí oproti předchozímu projektu je využití K9 jednotek - psovodů, jejichž cílem je adresovat zprávy lidem postiženým krizovou situací[17].

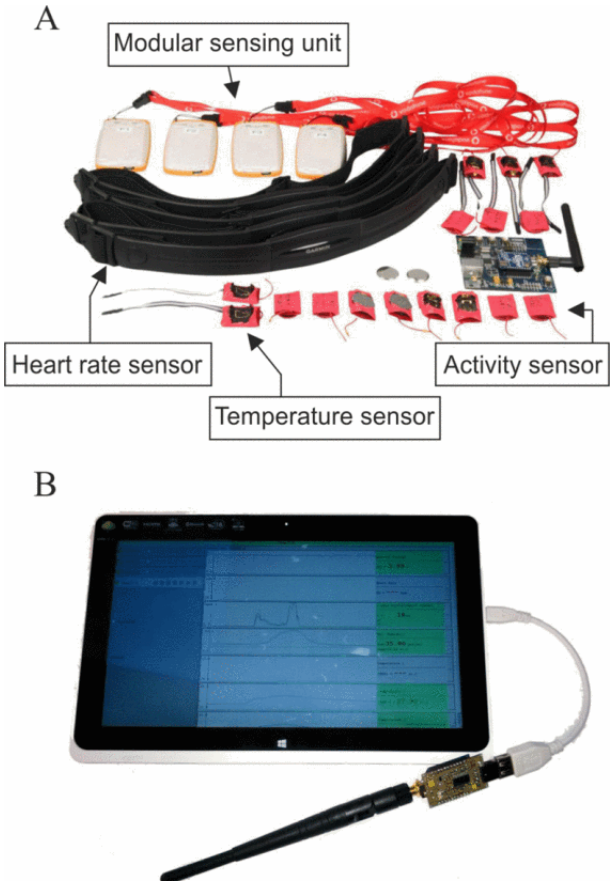
Kapitola 4
Analýza

V této kapitole bude představen systém FlexiGuard pro sběr dat, serverová část projektu,požadavky projektu a jeho uživatelé. V neposlední řadě bude představena GPS a projekce geografických souřadnic do kartézského sytému.

4.1 FlexiGuard

Flexiguard je biotelemetrický systém sloužící k monitorování fyzického stavu záchranářů, hasičů a vojáků v reálném čase [18]. Systém FlexiGuard (viz obrázek 4.1), vyvíjený na Fakultě biomedicínského inženýrství na ČVUT se zaměřením na monitorování jednotlivých členů týmu, je navržen tak, aby zvyšoval efektivitu a bezpečnost operací, či tréninků, poskytováním kritických fyziologických údajů a údajů o prostředí. Systém je schopen zachytit důležité stavy jako je vyčerpání, psychický stres nebo přehřátí, rozlišit intenzitu pohybu a celkový energetický výdej [19].

Systém se skládá ze tří částí popsaných v následujících podsekcích.



Obrázek 4.1. Flexiguard systém - převzato z [20]

4.1.1 Senzorická jednotka

Senzorická jednotka se skládá ze tří modulů - senzorový, hlavní a napájecí. Zde jsou využita komerční zařízení pro sledování srdečního tepu, teploty, relativní vlhkosti a aktivity [20].

4.1.2 Snímací jednotka

Druhou vrstvu systému FlexiGuard tvoří sofistikovaná sestava elektronických součástek a komunikačních modulů integrovaných do čtyřvrstvé desky s plošnými spoji (PCB). Tato vrstva má zásadní význam pro zpracování, ukládání a komunikační schopnosti systému [20].

4.1.3 Vizualizační jednotka

Poslední vrstva systému FlexiGuard, známá jako vizualizační vrstva, hraje klíčovou roli při prezentaci a analýze shromážděných dat. Tato vrstva je navržena tak, aby poskytovala velitelům, nadřízeným nebo jakémukoli odpovědnému personálu komplexní přehled o psychofyziologickém stavu osob a fyzických parametrech prostředí [20].

4.2 Server

Již hotová serverová část projektu není součástí této práce. Data nasbíraná pomocí FlexiGuard zařízení jsou uložena v PostgreSQL databázi a poskytována serverovou aplikací. Serverová aplikace, vytvořená v jazyce Java, vystavuje REST HTTP rozhraní pro listování jednotlivých uložených misí a WebSocket rozhraní pro konzumaci dat o poloze a stavu členů mise. Pro serializaci dat je použit JSON formát. Serverová aplikace obsahuje dokumentaci svých rozhraní ve specifikaci OpenAPI a AsyncApi a jejich yaml soubory.

4.2.1 RESTové HTTP rozhraní

REST (Representational State Transfer) je architektonický styl pro návrh aplikací. REST vytvořil Roy Fielding ve své doktorské disertační práci v roce 2000 a jedná se o soubor principů, které definují, jak by měly být webové standardy, jako je HTTP, používány k vytváření škálovatelných a flexibilních webových služeb. REST není protokol ani norma, ale soubor pokynů a osvědčených postupů pro vytváření efektivních síťových aplikací [21].

HTTP rozhraní je zdokumentováno pomocí OpenAPI specifikace. Server poskytuje dva URI identifikátory, jeden pro získání seznamu misí a druhý pro čtení detailu jednotlivých misí.

- [GET, POST] **/active-mission** pro čtení a vytvoření nové mise.
- [GET, DELETE] **/active-mission/mission_id** pro získání detailu mise.

```
[
  {
    "mission_id": 3025,
    "name": "Active mission Charlie",
    "start_datetime": "2021-04-10T17:00:00Z"
  },
  {
    "mission_id": 4256,
```

```
    "name": "Active mission Delta",
    "start_datetime": "2021-04-09T17:00:00Z"
  }
]
```

Obrázek 4.2. Příklad odpovědi obsahující seznam misí

4.2.2 WebSocket rozhraní

Protokol WebSocket, standardizovaný jako RFC 6455 [22], představuje významný pokrok v technologii webové komunikace. Poskytuje plně duplexní komunikační kanál přes jediné dlouhodobé spojení TCP, které umožňuje nepřetržitou výměnu dat mezi klientem (například webovým prohlížečem) a serverem. Nabízí tak alternativu k HTTP protokolu s jeho tradičním request-response přístupem a server pollingu, tedy konstantnímu dotazování serveru pomocí HTTP požadavků.

WebSocketová komunikace pro přenos dat konkrétní mise je zahájena otevřením Websocketového kanálu na **/ws/mission-data** a následným odeslání zprávy **SubscribeMissionMessage**. V té specifikujeme identifikátor mise a zpoždění oproti reálnému času.

```
{
  "request_id": "string",
  "code": 100,
  "mission_id": 0,
  "delay": 0
}
```

Obrázek 4.3. Zpráva pro spuštění streamu dat mise

Server poté začne streamovat data o pozicích a stavu jednotlivých členů mise. Každý člen je označen identifikátorem a jeho stav je reprezentován atributem **mls** (Predicted mortality likelihood), tedy číslem v intervalu od nuly do sta. Nula značí nulovou pravděpodobnost, že nastane kritický stav. Hodnota sto pak znamená kritický, či fatální stav.

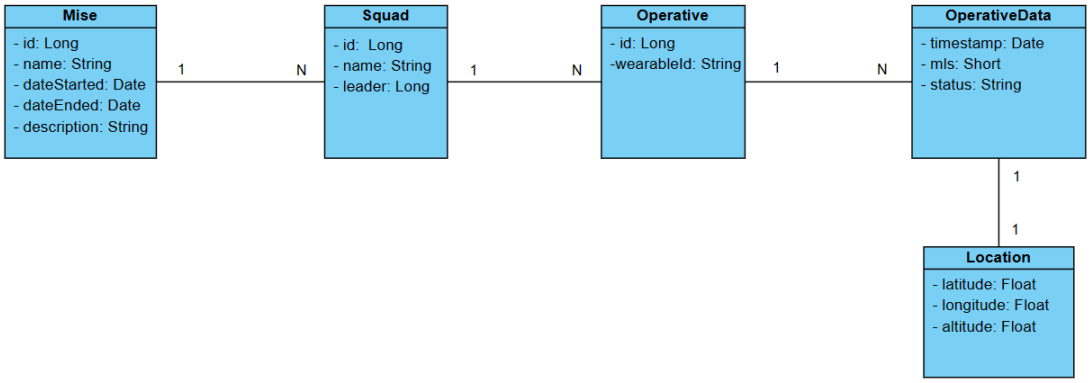
```
{
  "code": 4,
  "soldier": {
    "id": "string",
    "location": {
      "lat": 0,
      "lon": 0,
      "alt": 0
    },
    "mls": 100,
    "endangerment": "dead",
    "lastTimestamp": "2019-08-24T14:15:22Z",
    "status": "ok"
  }
}
```

Obrázek 4.4. Příklad zprávy obsahující data o členovi mise

4.3 Doménový model

Do domény systému patří čtyři entity - mise, jednotka, člen a členova data. Diagram doménového modelu je na obrázku 4.5.

- **Mise:**
  Mise seskupuje vizualizovaná data o jednotlivých členech. Je popsána unikátním identifikátorem, datem začátku a konce, jménem a textovým popisem.
- **Jednotka:**
  Do mise je zapojena hierarchická struktura jednotek tvořená jednotlivými členy. Každá jednotka má svého velitele, jméno a identifikátor.
- **Člen:**
  Jednotliví členové jsou označeni unikátním identifikátorem a identifikátorem nositelné elektroniky.
- **Data z nositelné elektroniky:**
  Data z nositelné elektroniky představují geografické souřadnice a identifikátor **mls**. Tento identifikátor je klíčový pro zobrazení stavu jednotlivce. Každý záznam z nositelné elektroniky je označen přesným časem pořízení.



Obrázek 4.5. Diagram doménového modelu

4.4 Požadavky

V této sekci jsou představeny funkční a nefunkční požadavky projektu.

4.4.1 Funkční požadavky

- **FR1** Zobrazení polohy uživatele:
  Systém zobrazí geografickou polohu uživatele v reálném čase.
- **FR2** Zobrazení polohy členů mise:
  Systém zobrazí polohu, směr a vzdálenost jednotlivých členů mise v reálném čase. Zobrazení bude provedeno jak ve 2D v uživatelském rozhraní, tak 3D pomocí nástrojů rozšířené reality.
- **FR3** Zobrazení stavu členů mise:
  Systém zobrazí zdravotní stav jednotlivých členů v závislosti na jejich **mls** ukazateli. Zobrazení stavu bude opět provedeno jak ve 2D, tak 3D.
- **FR4** Zobrazení mapy:
  Systém zobrazí mapu okolí uživatele a vyznačí jeho polohu a polohu ostatních členů.

- **FR5** Nastavitelnost zobrazení:
  Uživatel si zvolí vzdálenost k cíli pro škálování objektů.

4.4.2 Nefunkční požadavky

- **NFR1** Usability:
  Uživatelské rozhraní je jednoduché, intuitivní a nerušivé při práci se stresovou zátěží.
- **NFR2** Podpora platformy Android:
  Systém musí podporovat operační systém Android 8.0 (API 26) a pozdější verze.

4.5 Uživatelé systému

Uživatelem představeného systému jsou primárně členové integrovaných záchranných složek. Systém je navržen pro použití jak pro jednotlivce v terénu, tak osoby zodpovědné za řízení mise z přilehlých kontrolních stanic.

4.6 GPS

Globální polohový systém (GPS) je satelitní navigační systém, který poskytuje geolokační a časové informace přijímači GPS kdekoli na Zemi nebo v její blízkosti. Skládá se ze soustavy družic obíhajících kolem Země na šesti oběžných dráhách rozmístěných tak, aby uživatel byl v dosahu alespoň čtyř satelitů, které vysílají signály umožňující přijímačům GPS určit jejich aktuální polohu, čas a rychlost [23]. Systém používá k určení přesné polohy uživatele techniku zvanou triangulace. Měřením vzdálenosti mezi přijímačem a několika satelity může přijímač určit svou polohu.

4.6.1 WGS 84

Světový geodetický systém 1984 (WGS 84) představuje vývoj a zpřesnění globálního geodetického referenčního systému. Vznikl jako první verze WGS 60 a prošel několika aktualizacemi až do současné podoby WGS 84. Tento systém byl vyvinut s dvojím cílem: sladit se s nejlepšími dostupnými vědeckými referenčními systémy a udržet úroveň stability nezbytnou pro praktické aplikace [24]. Byl navržen jako praktický standardní globální geocentrický horizontální referenční bod pro mapování, geodézii a navigaci [24]. Jako referenční elipsoid volí WGS 84 geocentrický ekvipotenciální elipsoid a definuje následující důležité konstanty.

- Délka hlavní poloosy [24].
$$a = 6378137m$$
- Zploštění [25]
$$f = 1/298.257223563$$
- Délka vedlejší poloosy [25]
$$b = a(1 - f)$$

4.7 Projekce geografických souřadnic

Geografické souřadnice (zeměpisná šířka ϕ, délka λ a nadmořská výška h) jsou hodnoty v radiánech o poloze členů mise poskytované serverem. Pro splnění funkčního požadavku FR2, tedy zobrazení polohy jednotlivců, je nutné nejdříve nalézt způsob, jak tyto souřadnice transformovat do kartézské soustavy (X, Y, Z). V této sekci bude představeno Web Merkator zobrazení a Bowringova rovnice.

4.7.1 Web Merkator zobrazení

Merkatorovo zobrazení zavedl v roce 1569 Gerardus Mercator. Jedná se o projekci na válcovou mapovou projekci, která se stala standardem pro námořní navigaci. Merkatorovo zobrazení je konformní, což znamená, že zachovává úhly, takže je užitečné pro relativně přesné znázornění tvarů kontinentů a zemí [26].

Merkatorova projekce sice zachovává úhly a tvary, ale výrazně zkresluje velikost a vzdálenost, zejména v blízkosti pólů. Toto zkreslení má za následek, že oblasti ve vysokých zeměpisných šířkách, jako je Grónsko a Antarktida, vypadají ve srovnání s rovníkovými oblastmi mnohem větší, než je jejich skutečná velikost [27].

Merkátorova projekce sférické šířky ϕ a délky λ do X a Y (východní a severní šířky) je popsána rovnicemi 1, 2, kde R je poloměr sféry [28].

$$X = R(\lambda) \tag{1}$$

$$Y = R \ln \tan(\frac{\phi}{4} + \frac{\lambda}{2}) \tag{2}$$

Projekce šířky ϕ a délky λ z elipsoidu rovnicemi 3, 4, kde a je délka hlavní poloosy a e je excentricita elipsoidu. Tu můžeme spočítat pomocí rovnice 5 [25].

$$X = a\lambda \tag{3}$$

$$Y = a \ln(\tan(\frac{\pi}{4} + \frac{\phi}{2})(\frac{1 - e \sin \phi}{1 + e \sin \phi})^2) \tag{4}$$

$$e^2 = f(2 - f) \tag{5}$$

Web Merkator je označení pro variantu Merkatorové projekce používanou především pro 2D rastrové online mapy. Tato projekce se stala standardem po uvedení Google Maps v roce 2005 [26] a pro výpočet využívá sférické rovnice se souřadnicemi z WGS84 elipsoidu [28].

Tím se oproti původnímu Merkatorovu zobrazení stává nekonformní (nezachovává úhly). Byť je výsledný rozdíl na globálním měřítku nepatrný, v téměř krajních hodnotách u pólů mohou rozdíly dosáhnout až 50 km [26].

4.7.2 Bowringova rovnice

Bowringova rovnice, kterou ve 20. století formuloval Brian Bowring, je matematický vzorec používaný při geodetickém měření a mapování. Poskytuje jednoduchou a efektivní metodu pro převod mezi geodetickými a kartézskými souřadnicemi [29]. Transformace souřadnic elipsoidu (ϕ, λ, h) do kartézské soustavy (X, Y, Z) dosáhneme pomocí následujících rovnic 6, 7, 8,

$$X = (N + h) \cos \phi \cos \lambda \tag{6}$$

$$Y = (N + h) \cos \phi \sin \lambda \tag{7}$$

$$Z = (N(1 - e^2 + h)) \sin \phi \tag{8}$$

kde

$$N = \frac{a}{\sqrt{1 - e^2 \sin \phi^2}} \tag{9}$$

$$e^2 = \frac{a^2 - b^2}{a^2} \tag{10}$$

4.8 Harvesinova rovnice

Haversinova rovnice je základním nástrojem v navigaci a geografii, který se používá k určení nejkratší vzdálenosti mezi dvěma body na povrchu koule, jako je Země. Tento výpočet je zvláště užitečný při určování vzdáleností mezi dvěma zeměpisnými místy vzhledem k jejich zeměpisným délkám a šířkám [30]. Obecnou Harvesinovu rovnici můžeme vidět v rovnici 11, kde θ je středový úhel mezi dvěma sférickými body (ϕ_1, λ_1) a (ϕ_2, λ_2) [31].

$$Haversine(\theta) = \sin^2 \frac{\theta}{2} \tag{11}$$

Výpočet vzdálenosti mezi těmito body pak určíme jako:

$$d = 2R \arcsin(\sqrt{\frac{(\sin \phi_2 - \phi_1}{2}^2) + \cos \phi_2 \cos \phi_1 (\frac{\sin \lambda_2 - \lambda_1}{2}^2)}) \tag{12}$$

Byť Harvesinova rovnice uvažuje nad Zemí jako nad dokonalou koulí a GPS souřadnice v WGS 84 formátu uvažují s elipsou, dostáváme stále velice přesné výsledky. O něco přesnější je Vincentyho iterativní přístup, který již Zemi uvažuje jako elipsu [31].

4.9 Zvolené technologie

V této sekci a následujících podsekcích budou představeny zvolené technologie pro tvorbu prototypu. Unity a ARCore jsem již představil v předchozí kapitole. Zde zmíním důvody pro jejich výběr.

4.9.1 Unity

Jelikož byl původní cíl vytvořit prototyp pro brýle XReal, byl zvolen nástroj Unity. SDK pro brýle XReal je vytvořeno pouze pro Unity. U Unity jsem zůstal pro možnost multiplatformního vývoje, neboť je jedním z nefunkčních požadavků podpora platformy Android. Dále proto, že je podporována velkou řadou SDK pro jednotlivá zařízení, jako jsou chytré brýle. Pro případné navázání na tuto práci s podporou konkrétního hardwaru se Unity jeví jako ideální kandidát. Posledním faktorem je, že je práce s 3D a herním enginem pro mě novou výzvou.

4.9.2 ARCore

Ačkoliv by nebylo nutné do práce zapojit další nástroj pro práci s AR, kromě výchozího frameworku ARFoundation v Unity, rozhodl jsem se využít knihovnu ARCore. ARCore jsem zvolil, jelikož je to softwarové řešení od jedné z největších firem na trhu. Nativně podporuje Android a nabízí zajímavé geoprostorové rozhraní s možností využití geografických kotev. Dalším z faktorů je technologie VPS, představená v předchozí kapitole, pro zpřesnění lokalizace zařízení.

4.9.3 Rozhraní statických map

Pro splnění požadavku **FR4** je nutné najít rozhraní, které poskytne statický obrázek mapy. Všichni představení poskytovatelé těchto služeb nabízejí téměř totožnou funkcionalitu. Po registraci uživatel obdrží unikátní autentizační klíč. Pomocí klíče je možné získat obrázek zasláním HTTP požadavku. Požadavek je rozšířen o geografický střed mapy nebo geografické hranice zobrazené mapy, velikost, přiblížení a podobně.

- **Google Maps** [1]
  Google Maps jsou pravděpodobně nejznámější službou v oblasti digitálních map navigace. Jejich API bohužel není dostupné zdarma ani v omezené podobě. Maximální velikost může být pouze 640x640. K využívání rozhraní je nutné nejdříve povolit účtování za služby. Cena činí dva dolary za každých tisíc požadavků, pokud celkový počet nepřesáhne sto tisíc.
- **MapTiler** [2]
  MapTiler nabízí mapy v rozlišení 1024x1024 se zajímavou možnost zobrazení vlastních značek, dle zadaných souřadnic, případně s vlastní ikonou, či barvou. Další možností je automatické upravení velikosti mapy v závislosti na zobrazených značkách. Pozice jednotlivých značek je předána jako parametr do url. Statické mapy jsou zdarma s omezeným počtem požadavků.
- **Mapbox** [3]
  Mapbox poskytuje, kromě statických map, velkou řadu dalších mapových a navigačních služeb. Mapbox poskytuje statické mapy do velikosti 1024x1024 s podobnými možnostmi jako MapTiler, rozšířenými o další možnosti. Značky jsou zde specifikovány ve formátu GeoJson. Mapy podporují pouze WebMerkator projekci. I tato služba je zdarma v limitované podobě - do padesáti tisíc požadavků.

  Pro implementaci systému jsem se rozhodl pro službu Mapbox. Jednak díky bezplatné registraci a dobrému limitu v bezplatné verzi, tak také pro její rozsáhlejší, přehlednou dokumentaci.

4.9.4 Verzovací nástroje

Verzovací nástroje jsou systémy, které umožňují zaznamenání provedených změn do souboru v průběhu času tak, aby bylo možné se ke konkrétním verzím kdykoli vrátit zpět. Verzovací nástroje tak umožňují udržovat integritu jednotlivých souborů, sledovat jejich vývoj a jednotlivé změny i s uživatelem, který změny provedl [32].

Obrázek 4.6. Popularita verzovacích systémů - převzato z [33]

- **Git**
  Git vytvořil Linus Thorvalds, autor Linuxu, v roce 2005. Jedná se o distribuovaný systém, kdy klient si stahuje celý repositář včetně jeho kompletní historie. Pokud dojde k selhání a ztrátě dat na serveru, se kterým systém komunikuje, je klient schopen veškerá data obnovit [32]. Git dnes patří mezi nejpopulárnější verzovací nástroje. Podle průzkumu od Stack Overflow, až téměř 94% vývojářů používá právě Git [33], viz obrázek 4.6. Na technologii Git je postaveno několik služeb. Mezi ně patří například GitLab, GitHub nebo BitBucket. Tyto služby kromě verzování kódu poskytují řadu dalších možností jako je DevOps.

  Git jsem zvolil pro jeho popularitu a také jako preferenci dle mé osobní zkušenosti.

[1] https://developers.google.com/maps/documentation/maps-static/usage-and-billing
[2] https://documentation.maptiler.com/hc/en-us/articles/360020805897-Static-maps-for-your-web
[3] https://docs.mapbox.com/api/maps/static-images/

Kapitola 5
Návrh

V této kapitole bude představen návrh implementované aplikace. V první části je to lo-fi prototyp uživatelského rozhraní. Dále pak jednotlivé komponenty a architektura aplikace.

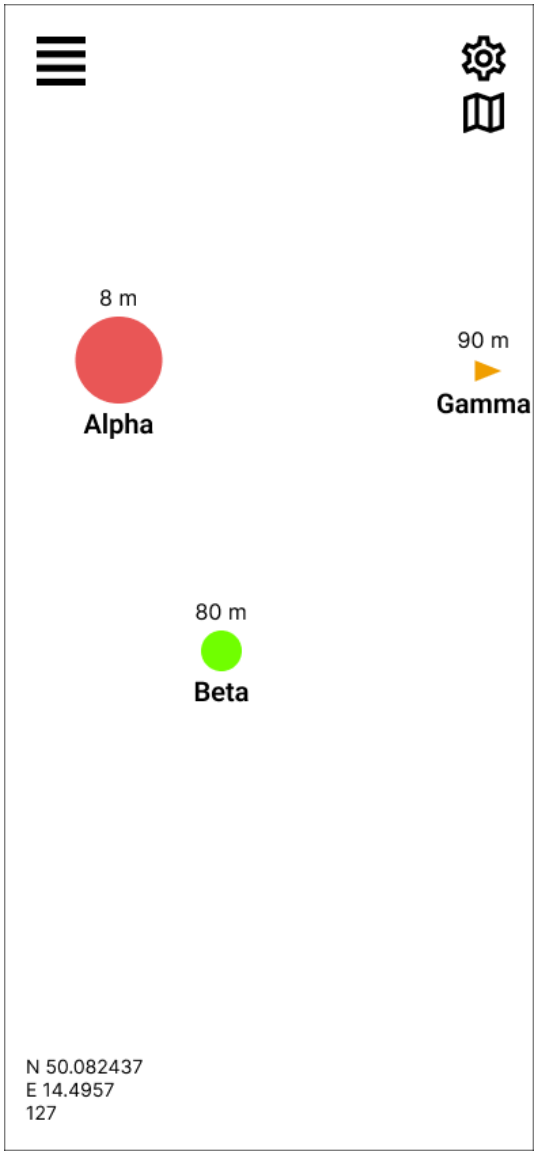
5.1 Návrh uživatelského rozhraní

Pro vytvoření prvotních návrhů uživatelského rozhraní implementovaného systému jsem zvolil lo-fi prototyp a návrhářský nástroj Figma s pluginem Material Design Icons pro snadné přidání již připravených ikon. Lo-Fi prototyp jsem zvolil pro jeho rychlou tvorbu a možnou rychlou zpětnou vazbu. Prototyp jsem rozdělil na tři hlavní části - směrové ukazatele, ovládací rozhraní a 3D objekty. Jednotlivé části jsou popsané v následujících podsekcích.

5.1.1 Směrové ukazatele

Směrové ukazatele řeší požadavek na zobrazení stavu a pozic členů mise ve 2D. Navrženým řešením je zobrazení značky pro každého člena mise na displeji tak, aby pozice na displeji reflektovala směr, kterým se člen nachází. Směrový ukazatel je navržený jako obyčejný barevný kruh, pod nímž se nachází text s identifikátorem člena mise a nad ním přesná vzdálenost. Barva značí stav a přechází ze zelené po červenou. Zelená znamená stav dobrý, sytě červená pak ten nejhorší. Velikost škáluje podle vzdálenosti mezi souřadnicemi uživatele a zobrazeného člena. Pokud se člen nenachází v zorném poli kamery, kruh se změní na šipku, která přilne ke straně obrazovky tak, aby po natočení kamery ve směru šipky se ukazatel do zorného pole dostal.

Barva značky přecházející ze zelené do červené není vybrána náhodně. Tento barevný gradient je typicky zvolen jako ukazatel zdraví v počítačových hrách, indikátor stavu baterie v telefonu, či na semaforech. Podle Plutchikova *kola emocí*, kde rozděluje základní barvy do osmi kategorií a přiřazuje jim osm emocí, je červená barva spojena s negativními emocemi a agresivitou. Světlejší odstíny zelené pak představují důvěru, nebo úžas, či obdiv [34].



Obrázek 5.1. Směrové ukazatele

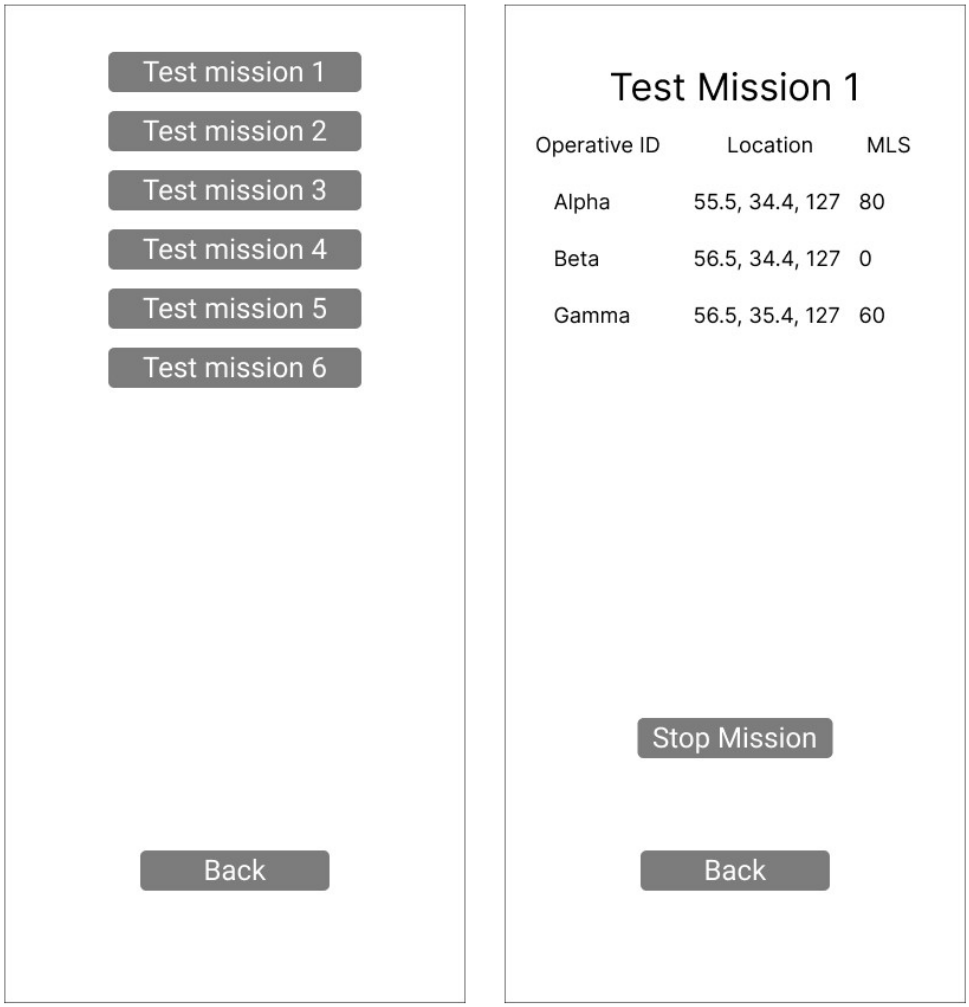
Na obrázku 5.1 je možné vidět tři značky pro členy s identifikátory Alpha, Beta, Gamma. Alpha je k uživateli ve vzdálenosti osmi metrů, a proto je jeho značka největší. Jeho stav je značen červenou barvou a mělo by se mu dostat okamžité pomoci. Gamma je nejvzdálenější a nenachází se v zorném poli. Trojúhelník směřující do pravé strany značí směr otočení kamery.

5.1.2 Ovládací rozhraní

Ovládací rozhraní slouží k manipulaci systému, zejména pak ke spuštění a zastavení přehrávání mise, zobrazení nastavení a mapy. Jelikož se jedná o aplikaci rozšířené reality, je na pozadí vždy zobrazen obraz uživatelovy kamery. Jednotlivé obrazovky ovládacího rozhraní pouze transparentně překrývají obraz kamery.

Hlavní obrazovka viz. obrázek 5.1 obsahuje tři tlačítka. V levém horním rohu tlačítko pro zobrazení seznamu misí a v pravém horním rohu dvě tlačítka pro zobrazení nastavení a zobrazení mapy. V levém spodním rohu je zobrazena aktuální poloha uživatele.

Dalšími obrazovkami je list misí a detail mise 5.2. Zde dojde k jednoduchému zobrazení všech dostupných misí. Po zvolení konkrétní mise se zobrazí její detail - název, účastníci, jejich aktuální pozice, stav a **mls** identifikátor. Pokud uživatel již misi nechce sledovat, může ji pomocí tlačítka *Stop Mission* zastavit.

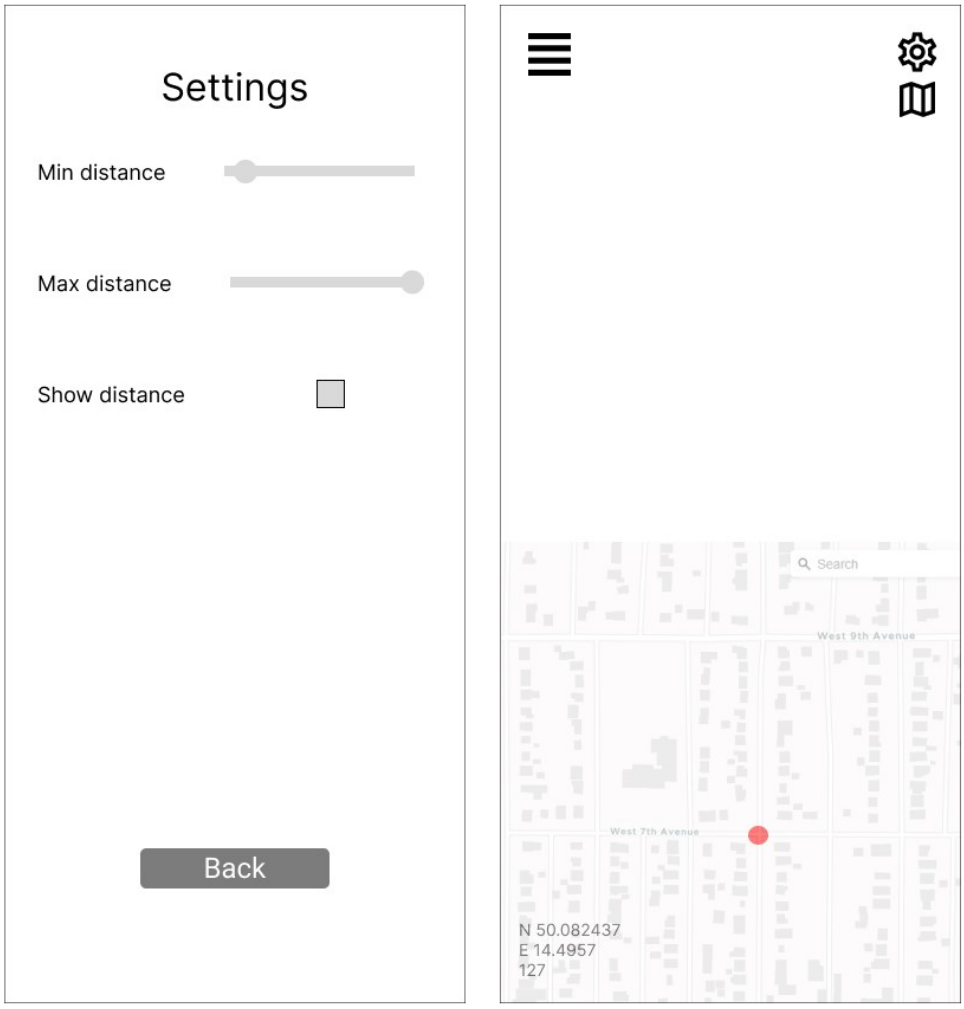


Obrázek 5.2. List misí a detail

Posledními obrazovkami prototypu je obrazovka nastavení a mapa 5.3. Zde může uživatel měnit jednotlivé parametry pro zobrazení směrových indikátorů. Navržené možnosti v nastavení jsou následující.

- **Min distance:** Nastaví spodní hranici vzdálenosti, kdy bude směrový indikátor největší.
- **Max distance:** Nastaví horní hranici vzdálenosti, kdy bude směrový indikátor nejmenší.
- **Show distance:** Je přepínač označující, zda-li má být zobrazena textová vzdálenost u směrových indikátorů.

Mapa je statický rastrový obrázek překrývající část hlavní obrazovky. Na mapě je vyznačena poloha uživatele a případně polohy dalších členů mise. Aby textura mapy byla co nejméně rušivá například v chytrých brýlích, je nastavena její průhlednost.



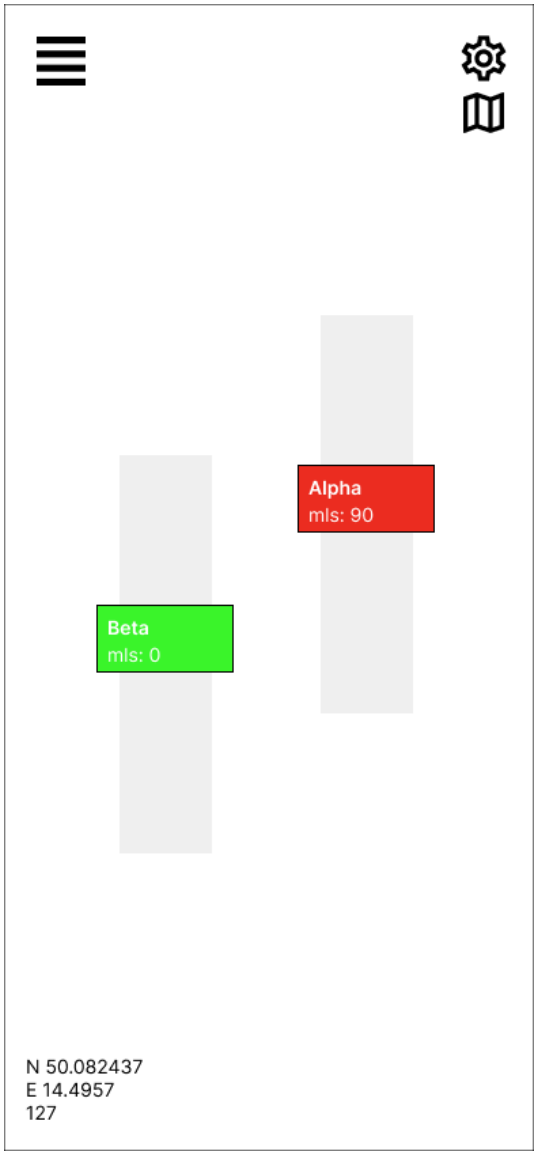
Obrázek 5.3. Obrazovka nastavení a mapa

V případě, že jakákoliv akce selže (připojení k serveru, vytvoření kotev), bude uživateli zobrazen jednoduchý, uzavíratelný dialog s chybovou hláškou.

5.1.3 3D objekt

Zatímco směrové ukazatele cílí na zobrazení pozice a stavu členů, kteří jsou od uživatele vzdálenější a umístění značky na displeji znázorňuje směr a případnou jednoduchou navigaci k cíli, 3D objekt je určen pro znázornění stavu jednotlivců, je-li uživatel v jejich relativní blízkosti. V ten moment jsou směrové ukazatele spíše přítěží a je příhodné plně využít možností rozšířené reality.

Člen mise je reprezentován jako neviditelný válec o výšce 170 cm 5.4. Tento válec slouží pro správné zasazení dat do reálného světa. K válci je připnuta karta s detailem. Barva karty evokuje členův stav jako v předchozím případě. Na kartě je textový identifikátor člena a jeho přesné **mls**.



Obrázek 5.4. 3D značka

5.2 Komponenty

Komponenty reprezentují modulární části systému, které komunikují pomocí předepsaných rozhraní. Komponenty nám dávají ucelenou představu o jednotlivých částech systému, jejich odpovědnosti a komunikaci. Návrh implementovaného systému jsem rozdělil na několik komponent znázorněných na komponentovém diagramu 5.5 vytvořeném pomocí webového nástroje *Visual Paradigm Online*. Jednotlivé komponenty jsou popsány níže.

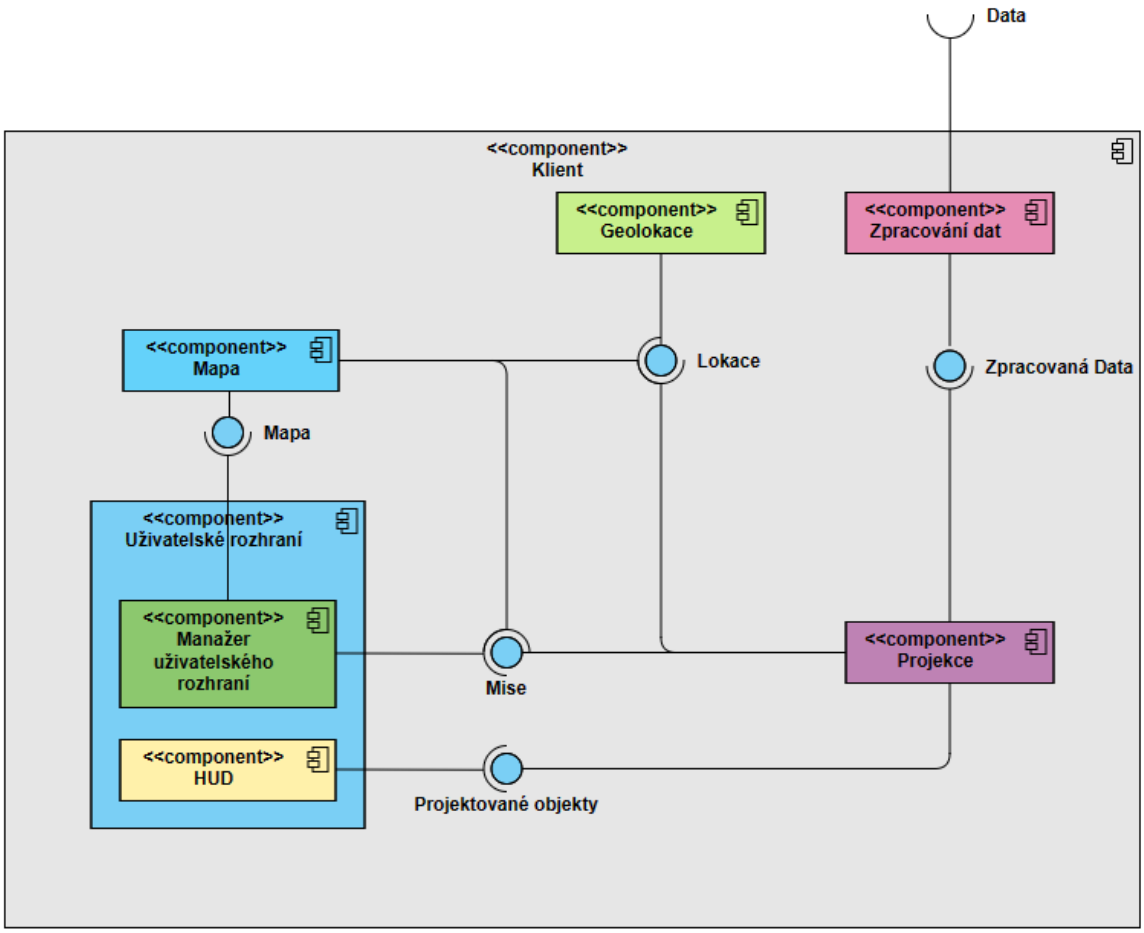
- **Zpracování dat**

  Komponenta zpracování dat řeší připojení k serveru jak pomocí HTTP protokolu, tak protokolu WebSocket. Dále řeší serizalizaci a deserializaci dat, tedy převedení do formátu JSON a zpět.

- **Geolokace**

  Geolokační komponenta se stará o aktualizaci a poskytování uživatelovy GPS polohy.

- **Projekce**
  Komponenta projekce přijímá de-serializované zprávy ze serveru. Její hlavní funkcionalitou je vytvoření vnitřní datové reprezentace mise, vytvoření vizuální reprezentace členů a transformace jejich GPS souřadnic do Kartézského systému.
- **Uživatelské rozhraní**
  Uživatelské rozhraní má dvě dílčí komponenty. První komponenta, manažer uživatelského rozhraní, se stará o logiku přepínání jednotlivých obrazovek a jejich obsah. Druhá komponenta, komponenta HUD, o logiku přiřazení správných pozic a viditelnost směrových ukazatelů.
- **Mapa**
  Komponenta mapy komunikuje se službou třetí strany pro získání rastrového obrázku, použitého jako textura zobrazené mapy na základě aktuální polohy a vyznačí na mapě polohu uživatele a členů mise.



Obrázek 5.5. Komponentový diagram

5.3 Architektura

Návrhové vzory jsou jakýmsi souborem zásad nebo doporučení, které nabízí čisté, snadné, časem otestované řešení často se opakujících softwarových návrhových pro-

blémů [35]. Model View Controller (MVC), Model View Presenter (MVP) a Model View View-Model (MVVM), často označované jako architektonické návrhové vzory, jsou jedny z nejznámější vzorů pro návrh systémů s uživatelským rozhraním zejména při vývoji webových, či velkých podnikových aplikací. Jejich hlavním cílem je oddělit byznys logiku a data od jejich grafické reprezentace [36]. Unity ve své dokumentaci zmiňuje použití právě MVC nebo MVP, ale také fakt, že ne vše může v Unity naprosto odpovídat učebnicovým definicím jednotlivých vzorů. Dokumentace klade důraz na zásadu jediné odpovědnosti, tedy že každá třída kódu dělá pouze jednu věc a dělá ji správně.

5.3.1 Model View Controller

Model View Controller, jak název napovídá, dělí aplikaci na tři části a definuje jak jednotlivé části společně komunikují. Model reprezentuje data bez jakékoli logiky. *View* se stará o to, co uživatel vidí. Vykresluje obrazovky, komponenty uživatelského rozhraní v závislosti na *Modelu* a jeho změnách. *Controller* reaguje na uživatelovy vstupy, řídí logiku a aktualizuje hodnoty *Modelu*. MVC populární především pro webové frameworky.

5.3.2 Model View Presenter

Model View Presenter opět dělí aplikaci na tři části. *Model* představuje data, *View* aktualizuje uživatelské rozhraní s daty a informuje *Presenter* o uživatelských akcích. Ten plní úlohu prostředníka mezi *View* a *Modelem*, řídí logiku, upravuje model v závislosti na uživatelských akcích. *Presenter* naprosto odděluje závislost *View* na *Modelu*, což usnadňuje testování [37]. Pro menší projekty může MVP přinést nežádoucí komplexitu při striktním oddělení jednotlivých částí a značnou řadu šablonovitého kódu.

5.3.3 Porovnání

MVC i MVP dělí aplikaci na části tak, aby se oddělila logika, prezentace a data, byť trochu odlišnými přístupy. Podle komparativní studie od M. Rizwan Jameel Qureshi a Fatima Sabir je MVC lepší variantou pro menší aplikace s jednoduchou navigací, zatímco MVP je správnou volbou při testově orientovaném vývoji díky separaci závislosti *View* a *Modelu* [37]. Právě proto jsem pro implementaci prototypu zvolil vzor MVC.

Kapitola 6
Implementace

V této kapitole bude popsána implementace jednotlivých částí systému.

6.1 Podpora Android

Podpora platformy Android je zařízena čistě pomocí Unity a jejího nastavení. Implementovaná aplikace podporuje ARMv7 i ARM64 architektury. Minimální verze Android je nastavena na verzi 8 (API 26). Za grafické API je zvoleno OpenGLES3 a za skriptovací backend IL2CPP. Výsledný instalační soubor (.apk) je podepsaný privátním klíčem.

6.2 Zpracování dat

Část aplikace pro zpracování dat zprostředkovává komunikaci se serverem pomocí protokolu WebSocket a HTTP. Komponenta pro zpracování dat je implementována jako komponenta herního objektu v Unity. Komponenta implementuje rozhraní *ConnectionHandler* s předepsanými metodami pro serverovou komunikaci. Metody *StartMission*, *EndMission* a *ListenToMission* data pro WebSocket komunikaci, *ListMissions* a *GetMissionDetail* pro komunikaci pomocí HTTP.

```
public interface ConnectionHandler
{
    public void StartMission(long missionId);

    public void StopMission(long missionId);

    public void ListenToMissionData(UnityAction<Message> call);

    public IEnumerator ListMissions(
        UnityAction<List<Mission>> onResult,
        UnityAction<string> onError);

    IEnumerator GetMissionDetail(
        long missionId,
        UnityAction onResult,
        UnityAction<string> onError);
}
```

Obrázek 6.1. Rozhraní ConnectionHandler

Pro komunikaci prostřednictvím protokolu WebSocket jsem využil C# knihovnu *WebSocketSharp*. HTTP komunikace je realizována pomocí knihovny *UnityEngine.Networking*.

Serializace a deserializace zpráv do a z formátu JSON je implementována pomocí knihovny *NewtonSoftJson*. Zprávy jsou převedeny do C# objektů na základě jejich uvedeného atributu *code*.

Pro účely testování a demonstrace na fyzickém zařízení je vytvořen druhý objekt implementující rozhraní *ConnectionHandler*, který obsahuje statická testovací data.

6.3 Lokace

Jelikož je prototyp implementován na platformu Android, je nutné systém požádat o povolení k používání lokace a kompasu. O získání povolení se stará lokační servisa, implementovaná jako singleton, která data o lokaci průběžně aktualizuje a poskytuje ostatním komponentám. Pro získání přesné lokace je nutné si vyžádat práva na *Fine Location* a *Coarse Location*.

6.4 Zobrazení

Zobrazení, nebo projekce, je rozděleno na dvě části. První část je zobrazení směrových ukazatelů na displeji uživatele. Druhá část je vytvoření 3D objektů na transformovaných souřadnicích. Zvolený barevný gradient pro zobrazení stavu je k vidění na obrázku 6.2.

Obrázek 6.2. Barevný gradient pro zobrazení stavu

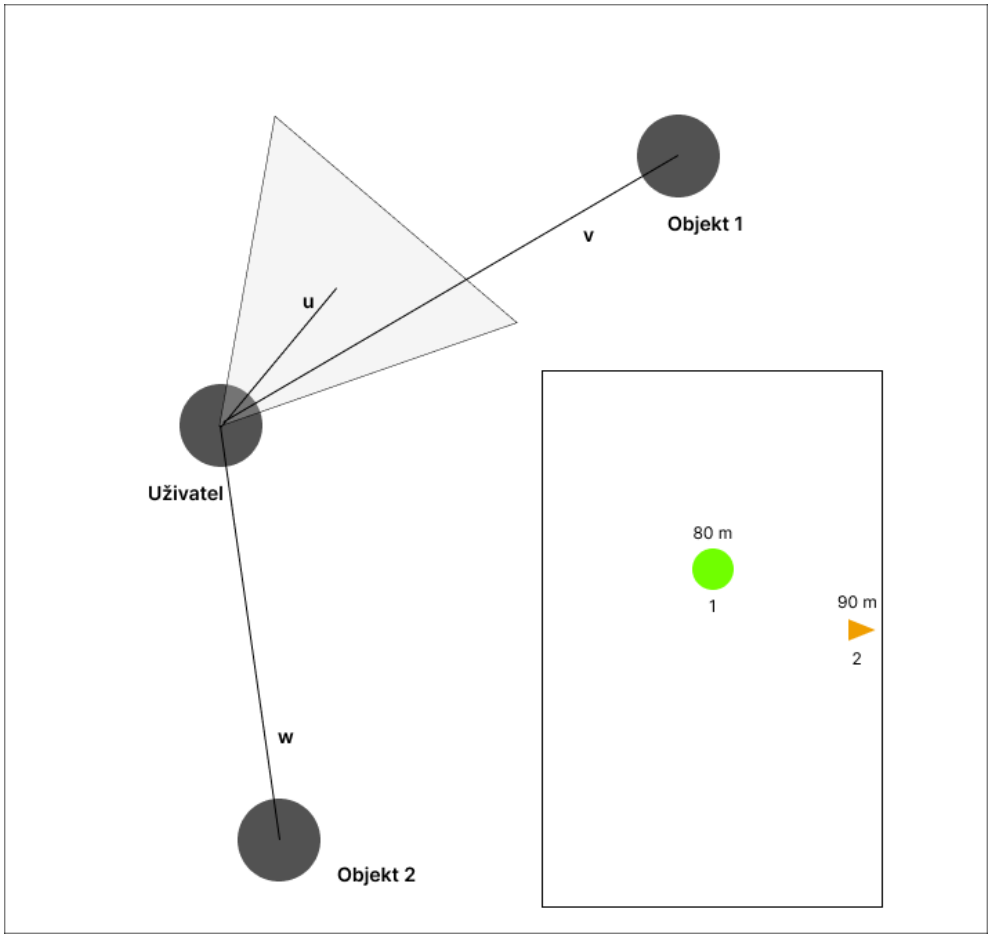
6.4.1 Směrové ukazatele

Směrové ukazatele jsou implementovány jako 2D herní objekty uživatelského rozhraní viz obrázek 6.3. Značka se skládá z obrázku a dvou textových polí. Textura obrázku je proměnná v závislosti na jeho pozici. Pokud je značka umístěna vně obrazovky, vykreslí se barevný kruh. V opačném případě je vykreslen trojúhelník rotovaný ve směru stěny, ke které je přichycen. Ikony jsou použité z dostupné Unity knihovny *simple icon pastel tone*[1].

Obrázek 6.3. Objekt směrového ukazatele

[1] `https://assetstore.unity.com/packages/2d/gui/icons/simple-icon-pastel-tone-107568`

Cílem směrových ukazatelů je vyznačit směr a vzdálenost k cílovému objektu. Nachází-li se objekt v těsné blízkosti uživatele, není jich již dále potřeba. Pro vykreslení značky je nejdříve nutné zjistit, kde se cílový objekt nachází v závislosti na natočení kamery. K tomu je využit jednoduchý skalární součin dopředného vektoru kamery a normalizovaného vektoru směru uživatele k cílovému objektu. Je-li výsledný součin kladný, cílový objekt se nachází před uživatelem. Objekty, které se nachází za uživatelem jsou připnuty ke straně obrazovky tak, aby bylo jasné, kterým směrem se natočit. Toto jsem se pokusil ilustrovat na obrázku 6.4, kde u je dopředný vektor kamery a v, w jsou směrové vektory k cílovým objektům. V pravém dolním rohu je poté ilustrace výsledného zobrazení 2D značek.



Obrázek 6.4. Projekce směrových ukazatelů

Zjištění viditelnosti 3D objektu pro schování 2D značky je poměrně ošemetné. K tomu zde vyžívám kombinaci testování rovin a *raycasting* 6.5. *Raycasting* označuje techniku vyslání virtuálního paprsku z jednoho bodu do druhého a zaznamená objekty, které paprsek na své dráze protne.

ARFoundation poskytuje vlastní *Raycast Manager*. Ten typicky reaguje na uživatelův vstup v podobě kliknutí na obrazovku. Z tohoto bodu dotyku vysílá paprsek do 3D scény. V ukázce kódu 6.5 je místo bodu dotyku na obrazovce vyžita proměnná *pos*, která značí, kde na obrazovce by se nacházela 2D značka.

```
Bounds bounds = renderer.bounds;
Plane[] planes = GeometryUtility.CalculateFrustumPlanes(arCamera);
```

```
bool isInCameraFrustrum =
    GeometryUtility.TestPlanesAABB(planes, bounds);

if(!isInCameraFrustrum)
{
    return false;
}

List<ARRaycastHit> hits = new();
if (arRaycastManager.Raycast(pos, hits, TrackableType.AllTypes)) {
    if(hits.Count > 0) {
        if (hits[0].trackable.gameObject == target)
        {
            return true;
        }
    }
}

return false;
```

Obrázek 6.5. Testování viditelnosti 3D objektu

6.4.2 3D objekt

3D objekt, podobně jako v návrhu, je 3D herní objekt skládající se z válce o velikosti $1.7m$. Data o stavu člena zobrazená ve 3D scéně jsou umístěna na plátně (*Canvas*), které se běžně používá k vykreslování uživatelského rozhraní. Toto plátno je součástí jeho 3D herního objektu. Aby byly informace na plátně dostupné a čitelné pro uživatele, je nutné jej rotovat vždy ve směru uživatele.

3D objekty jsou umisťovány do scény ve dvou režimech. Prvním je použití knihovny *ARCore*, pokud mobilní zařízení podporuje Geospatial API. V tomto případě jsou vytvořeny, či aktualizovány prostorové kotvy, na které jsou objekty přichyceny.

Druhý režim je aplikován pokud, pokud zařízení Geospatial API nepodporuje. Dochází k vlastní transformaci geografických souřadnic 6.6 do kartézského systému a následnému upravení pozice objektu.

```
public static Vector3 ConvertGeoToCartesian(
            double userLat, double userLon, double userAlt,
            double targetLat, double targetLon, double targetAlt)
    {
        double latRad = DegreesToRadians(userLat);

        // Calculate an approximate radius of earth at given latitude
        double currentRadius = GetRadius(userLat);

        // Get meters per degree of lat / lon
        double latToMeters = DegreesToRadians(currentRadius);
        double lonToMeters =
            DegreesToRadians(currentRadius * Math.Cos(latRad));
```

```
        double x = (userLon - targetLon) * lonToMeters;
        double y = userAlt - targetAlt;
        double z = (userLat - targetLat) * latToMeters;
        return new Vector3((float)x, (float)y, (float)z);
    }
```

Obrázek 6.6. Transformace souřadnic

Obrázek 6.7. 3D Ukazatel Unity

6.4.3 Zobrazení jednotek

Po otestování prototypu, které bude zmíněno v následující kapitole, jsem se rozhodl implementovat funkcionalitu, která místo jednotlivých členů zobrazí celé skupiny, či jednotky. Zobrazení celých skupin pomůže při vizualizaci velkých misí, kdy by na displeji uživatele bylo zobrazeno příliš mnoho indikátorů. Zobrazení je implementováno s předpokladem, že skupina pracuje společně, v krátké vzdálenosti od sebe. Implementovaný systém zprůměruje pozice jednotlivých členů pro zobrazení směrového ukazatele a vezme maximální hodnotu **mls** identifikátoru pro zobrazení stavu jednotky.

6.5 Okluze

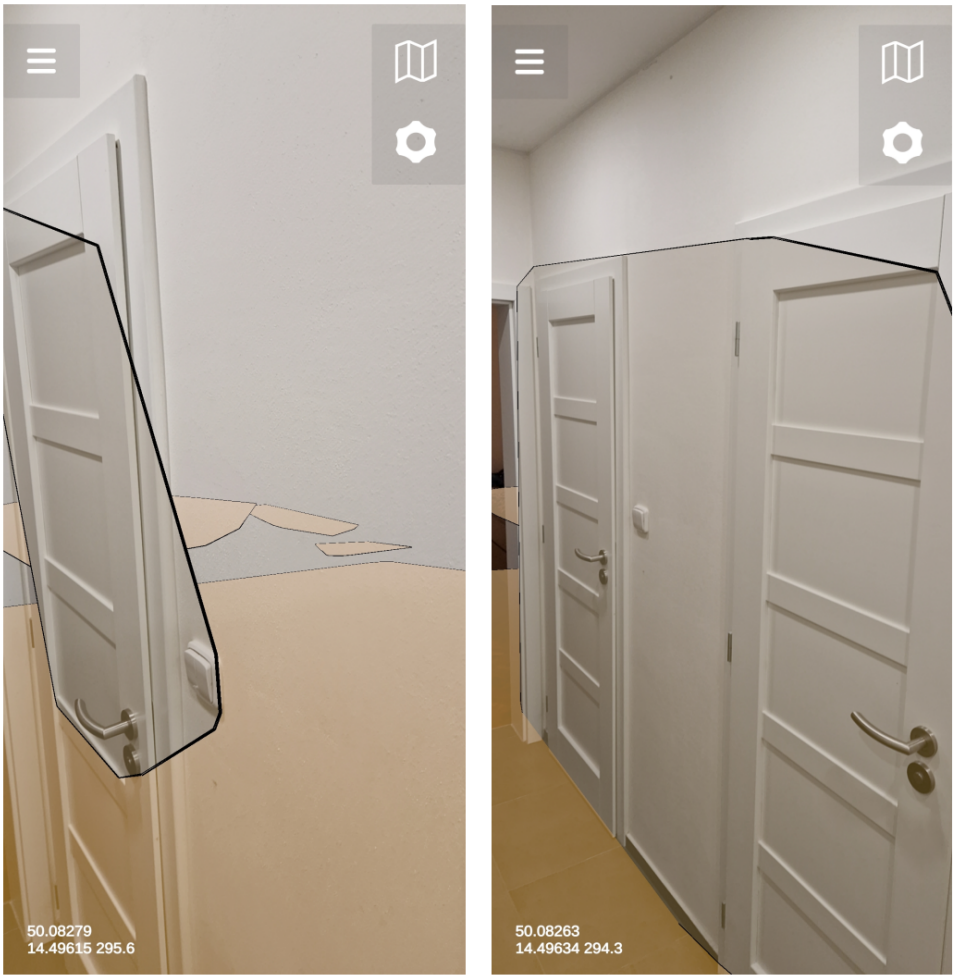
Okluze v systémech rozšířené reality popisuje problematiku zakrývání reálných i virtuálních objektů [38]. Pokud tuto problematiku systém neřeší, může docházet k situacím, kdy například virtuální objekt prostupuje stěnou, i když by neměl být viděn. Toto způsobuje, že se celý systém jeví méně věrohodně a reálně.

Pro řešení tohoto problému je klíčová detekce ploch. Plochy jsou detekovány jak horizontální, tak vertikální. O detekci se stará *AR Plane Manager* komponenta, která je součástí *AR Session Origin* objektu. Na detekované plochy je použit speciální *shader* aplikovaný na jejich komponentu materiálu.

Shader je malý program běžící na GPU, který umožňuje vytvářet nejrůznější grafické efekty. Tento program provádí matematické výpočty a sadu instrukcí pro zpracování barev každého pixelu daného objektu na displeji. Oficiálním jazykem pro tvorbu *shaderů* v Unity od verze 2019 je jazyk *HLSL* [39].

Tento *shader* zapříčiní, že cokoliv za detekovanými plochami z pohledu uživatele nebude vykresleno. Detekovaná plocha ale zároveň zůstane průhledná.

Použitý shader vychází z existujího shaderu ve veřejném repositáři Dana Millera[2], kdy jedinou změnou je nastavení průhlednosti fragmentu.



Obrázek 6.8. Ilustrace okluze

Problematika okluze je ilustrována na záběrech z aplikace na obrázku 6.8. Na levém obrázku je možné vidět detekované plochy označené světle hnědou barvou za použití *DebugPlane* materiálu. Plochy detekované v jedné místnosti je možné vidět skrze stěnu z druhé místnosti. Na pravém obrázku aplikace rozpoznala plochu stěny a dveří, které způsobí zakrytí vykreslených ploch v předchozí místnosti.

[2] `https://gist.github.com/DanMillerDev/6cfa871b2fe1ddbbbe0cb3dea08f7504`

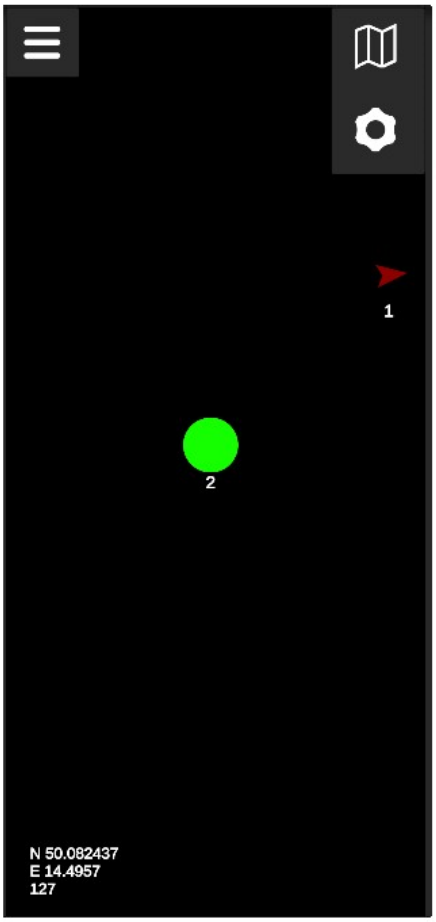
6.6 Mapa

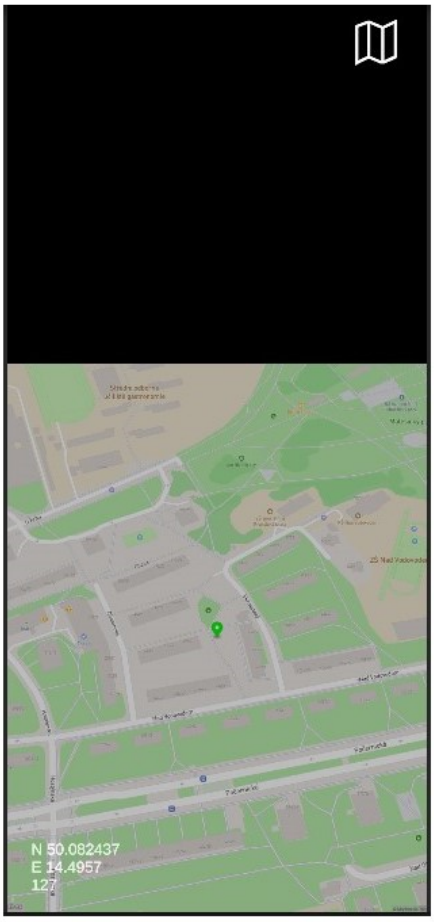
Mapa je implementována pomocí služby *MapBox*. Pro zobrazení mapy je odeslán HTTP požadavek s autentizačním klíčem a parametry specifikujícími polohu, velikost výsledného statického obrázku nebo informace o ukazatelích ve formátu *GeoJson*. Výsledný obrázek je použitý jako průhledná textura panelu uživatelského rozhraní.

Aby se předešlo zbytečnému odesílání požadavků, je o texturu požádáno pouze v případě, že se změnily souřadnice uživatele, či některého ze členů mise.

6.7 Uživatelské rozhraní

Uživatelské rozhraní je implementováno pomocí běžných komponent uživatelského rozhraní v Unity. Obrazovky jsou reprezentovány herními objekty, které jsou součástí plátna. Logiku přepínání jednotlivých obrazovek má na starost třída *UIManager*. O události (eventy) jednotlivých akčních prvků se starají *Controller* třídy daných obrazovek. Použité ikony jsou z volně dostupného Unity balíčku *simple icon pastel tone*[3]. Práce s uživatelským rozhraním mi v Unity přišla poměrně náročná, jelikož jsem s Unity v minulosti nepracoval. Aby jednotlivé prvky jako například tlačítka byla uživatelsky přívětivá, je téměř nutné použít kombinaci vlastních materiálů, shaderů, či obrázků. Chyběla mi zde možnost využití již existujícího řešení jako je knihovna *Material* pro Android.

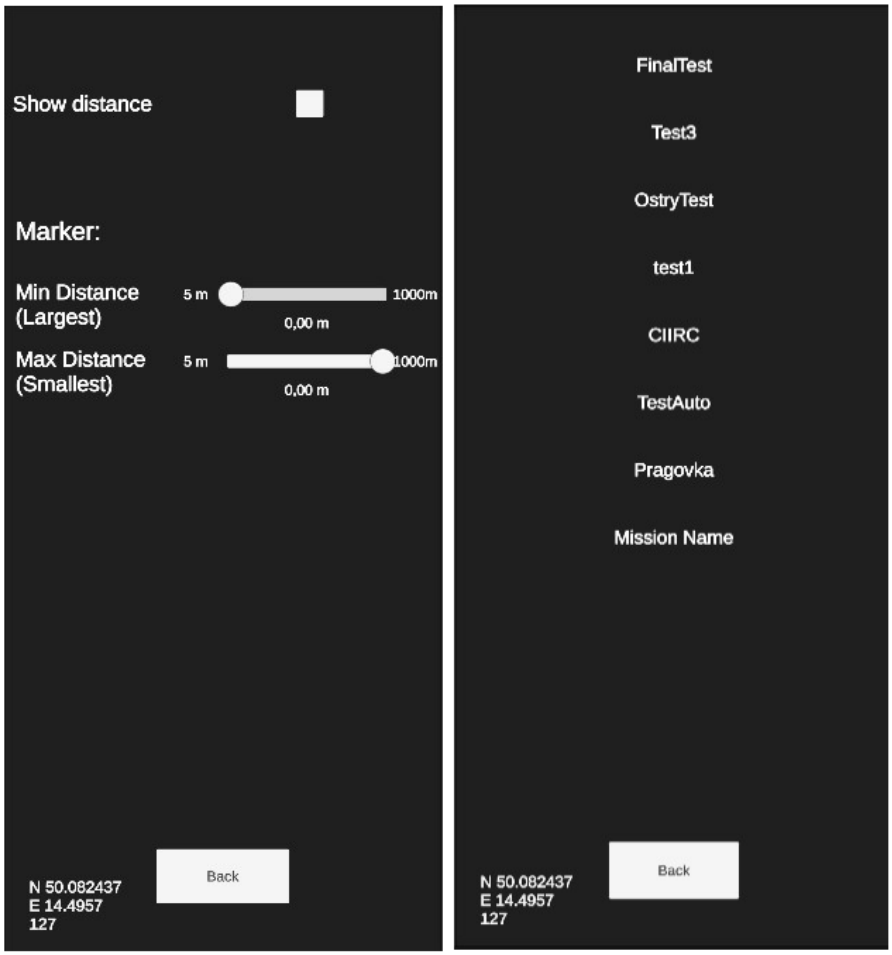




[3] https://assetstore.unity.com/packages/2d/gui/icons/simple-icon-pastel-tone-107568

Obrázek 6.9. Implementované rozhraní - hlavní obrazovka, mapa

Implementované uživatelské rozhraní je možné vidět na obrázku 6.9 a 6.10 pořízených v Unity editoru. Na obrázku 6.9 je možné vidět hlavní obrazovku s právě probíhající misí se dvěma směrovými ukazateli a zobrazení mapy. Na obrázku 6.10 je k vidění obrazovka nastavení a list testovacích misí.



Obrázek 6.10. Implementované rozhraní - list misí, nastavení

6.7.1 Responzivní design

Aplikace je primárně vyvíjena na zařízení Samsung Galaxy S22 s rozlišením 1080 × 2340. Blog Media Genesis[4] uvádí rozlišení jednotlivých vybraných Android i iOS zařízení. V rámci implementace jsem se soustředil nejen na pixelové rozlišení 1080 × 2340, ale také 1440 × 2960, 1440 × 2560 a 1080 × 1920.

6.8 Splnění požadavků

V této sekci bych se rád poohlédl zpět na představené požadavky a zhodnotil jejich splnění.

Funkční požadavky

- **FR1** Zobrazení polohy uživatele:
  Poloha uživatele je zobrazena fixně v levém spodním rohu dle návrhu.

[4] `https://mediag.com/blog/popular-screen-resolutions-designing-for-all/`

- **FR2** Zobrazení polohy členů mise:
  Poloha členů mise je zobrazena trojím způsobem - směrové ukazatele, 3D objekty a vyznačení na mapě.
- **FR3** Zobrazení stavu členů mise:
  Stav jednotlivých členů je převeden do barevného gradientu, který je součástí vizualizace pozic.
- **FR4** Zobrazení mapy:
  Mapa je implementována pomocí služby MapBox a zobrazena po stisknutí příslušného tlačítka. Na mapě jsou zobrazeny pozice uživatele a ostatních členů mise.
- **FR5** Nastavitelnost zobrazení
  Implementované možnosti nastavení zobrazení jsou změna dolní a horní hranice vzdálenosti pro škálování směrových ukazatelů markeru, volitelné zobrazování textové vzdálenosti a hranice pro přichycení značky ke kraji displeje.

Nefunkční požadavky

- **NFR1** Usability
  Zda-li je splněn nefunkční požadavek na intuitivnost ukáže uživatelské testování.
- **NFR2** Podpora platformy Android Nefunkční požadavek na podporu platformy Android je splněn.

Kapitola 7
Testování

V této kapitole bude představeno testování lo-fi prototypu a výsledné aplikace pomocí jednotkových testů a průběh uživatelského testování.

7.1 Scénáře

K otestování systému jsem připravil dva následující jednoduché scénáře. Tyto scénáře sledují reálné použití systému a testují jeho funkčnost a použitelnost.

- **Scénář 1: Lokalizace raněného**

  Uživatel se připojí do probíhající mise. Jeho cílem je najít člena mise, jehož stav není dobrý (může se jednat například o zraněného) a pomocí navrženého systému jej lokalizovat.

  Cílem tohoto jednoduchého scénáře je ověřit, zda je zobrazení stavu uživatelovi zjevné.

  Kroky:

  1. Přejděte do seznamu misí.
  2. Vyberte zadanou misi ze seznamu.
  3. Najděte směr, ve kterém se nachází člen s nejhorším stavem.
  4. Zobrazte si polohu cíle na mapě.
  5. Dojděte k místu, kde se člen nachází a identifikujte jej pomocí 3D objektu.

- **Scénář 2: Lokalizace nejbližšího raněného**

  Uživatel se připojí do probíhající mise. V probíhající misi je více členů, jejichž stav není dobrý. Cílem uživatele je co nejrychleji pomoci členovi, který se nachází v jeho blízkosti.

  Cílem tohoto druhého scénáře je vyzkoušet, zda škálování směrových indikátorů pomáhá uživateli určit vzdálenost k jednotlivým cílům.

  Kroky:

  1. Přejděte do seznamu misí.
  2. Vyberte zadanou misi ze seznamu.
  3. Najděte raněného člena mise s nejbližší polohou.
  4. Zobrazte si polohu cíle na mapě.
  5. Dojděte k místu, kde se člen nachází, a identifikujte jej pomocí 3D objektu.

7.2 Testování lo-fi prototypu

Testování prototypu pomáhá odhalit nedokonalosti, či případné chyby v návrhu v brzkých fázích projektu. Odhalené chyby je v lo-fi prototypu poměrně jednoduché a rychlé odstranit. Kromě snahy odhalit návrhové chyby je cílem mého testování ověřit, zda-li je splněn nefunkční požadavek na *usability*.

7.2.1 Výběr metody

K otestování prototypu a jeho použitelnosti (usability) se typicky používají tři metody - uživatelské testování, heuristická evaluace a kognitivní průchod. Jednou z prvních myšlenek bylo převést obrazovky prototypu do papírové podoby a vyzkoušet uživatelské testování. Po dalším zamyšlení jsem nedokázal najít vhodný způsob jak uživatelovi nasimulovat škálování a změny pozic jednotlivých značek v závislosti na vzdálenosti nebo orientaci v reálném čase.

Testování použitelnosti metodou heuristické evaluace spočívá v testování průchodu aplikací vybranými jednotlivci, kteří hodnotí kroky průchodu podle zadaných heuristik. Často používané heuristiky jsou od Jakoba Nielsena[1]. Aby tato metoda byla co nejefektivnější, je dobré k testování použít vícero hodnotitelů [40].

Třetí metodou je metoda kognitivního průchodu. Zde se snažíme vcítit do role uživatele a jeho přemýšlení. Na vybraných scénářích simulujeme uživatelův průchod aplikací a snažíme se odhalit nedostatky, či návrhové problémy [41]. Tato metoda je často prováděna právě designérem, či skupinou designérů. Ti zkoumají sled akcí zadaného scénáře a snaží se obhájit, proč by jednotlivé akce provedl i sám uživatel [42].

Pro testování prototypu jsem zvolil právě metodu kognitivního průchodu. Zvolil jsem ji z důvodu, že je možné ji provést sám, což usnadní a urychlí případné iterace v návrhu.

7.2.2 Výsledky

Kognitivní průchod byl proveden na krocích uvedených scénářů 7.1.

Jednotlivé akce při průchodu aplikací mi přijdou intuitivní. Celkové ovládání je poměrně jednoduché. Při představě zobrazených směrových indikátorů jsem objevil dva nedostatky, které můj návrh opomněl. Nálezy a jejich popisy jsou uvedeny níže.

- **Poznámka 1**
  - **Závažnost:** střední
  - **Popis:** Pokud operuje více členů na podobné pozici v určité vzdálenosti od uživatele, může dojít k překrytí jejich směrových indikátorů a zhoršení, či znemožnění jejich čitelnosti.
  - **Návrh řešení:** Možným řešením je implementace funkcionality, která vykreslené, překrývající se značky, seskupí do jedné.
- **Poznámka 2**
  - **Závažnost:** střední
  - **Popis:** Účastní-li se mise větší množství členů, může dojít k zahlcení uživatele. Případné velké množství indikátorů může být problém například v chytrých brýlích.
  - **Návrh řešení:** Pokud se mise účastní velké množství lidí, může být vhodnější variantou agregovat pozice jednotlivých členů jako celou jednotku. Toho může být docíleno pouze za předpokladu, že celá jednotka operuje společně.

Jelikož se nálezy týkají směrových ukazatelů, rozhodl jsem se návrh upravit o navrhované řešení druhého problému. Toto řešení částečně řeší i první nalezený problém. Upravený prototyp obsahuje možnost přepínání mezi zobrazením jednotlivců, či celých jednotek v obrazovce nastavení. Směrové značky zůstávají nezměněné.

[1] https://www.nngroup.com/articles/ten-usability-heuristics/

7.3 Jednotkové testy

Implementovaná aplikace je pokryta sadou jednotkových testů. Jednotkové testování v Unity je prováděno pomoci *Unity Test Framework*, který integruje knihovnu *NUnit* pro testování *.Net* jazyků. Podle oficiální dokumentace [2] je nutné testy oddělit do vlastního adresáře s vlastním *assembly* souborem. Ten zapříčiní, že testy nejsou součástí výsledného artefaktu.

7.3.1 Projekce

První částí aplikace, která je pokryta jednotkovými testy je vlastní transformace souřadnic. Zobrazování 3D objektů na správnou pozici je klíčové a tyto testy jsou oporou, že kód pro vkládání 3D objektů do scény 6.6 je vkládá na správné místo. Transformace souřadnic a výpočet vzdálenosti jsou testovány na základě předpokladu, že je zachována vzdálenost. Tím je myšleno, že vzdálenost dvou geografických bodů bude stejná, nebo alespoň velice podobná vzdálenosti projektovaných bodů do kartézského systému.

Souřadnicové trojice (lat_1, lon_1, alt_1) a (lat_2, lon_2, alt_2) označím jako α, β a projektované souřadnice (X, Y, Z) jako α_{proj}, β_{proj}. Chci aby platila nerovnost z rovnice 1, tedy aby rozdíl výpočtu Harvesinovy rovnice a Euklidovské vzdálenosti byl menší než ϵ.

$$|h(\alpha, \beta) - e(\alpha_{proj}, \beta_{proj})| < \epsilon \tag{1}$$

Jelikož jsou výsledné vzdálenosti v metrech, zvolil jsem za ϵ hodnotu 10^{-2}. Rozdíl v řádech několika centimetrů by neměl kazil dojem rozšířené reality nebo znepřesnit navigační indikátory.

```
[Test]
public void TestCoordinatesProjection()
{
    foreach (var c in coordinatesAndDist)
    {
        var dists =
            CompareDistance(c[0], c[1], c[2], c[3]);
        var geoDistM = dists.Item1;
        var cartDistM = dists.Item2;

        Assert.IsTrue(IsSimilar(geoDistM, cartDistM));
        Assert.IsTrue(IsSimilar(cartDistM,(float) coordinates[4]));

    }
}

private Tuple<float,float> CompareDistance(double lat1, double lon1,
    double lat2, double lon2)
{
    Vector3 pos = TransformCoordinates(lat1,lon1,0,lat2,lon2,0);

    float geoDistM = CoordinatesTransformer
```

[2] https://docs.unity3d.com/Packages/com.unity.test-framework@1.3/manual/getting-started.html

```
        .CalculateDistanceM(lat1, lon1, lat2, lon2);
    float cartDistM = Vector3
        .Distance(new() { x = 0, y = 0, z = 0 }, pos);

    return Tuple.Create(geoDistM, cartDistM);
}

private bool IsSimilar(float a, float b)
{
    return Mathf.Abs(a- b) <
    Mathf.Max(
        1E-02f * Mathf.Max(Mathf.Abs(a), Mathf.Abs(b)),
        Mathf.Epsilon * 8f);
}
```

Obrázek 7.1. Jednotkové testy pro projekci souřadnic

Test iteruje přes zadefinované dvourozměrné pole s dvojicemi souřadnic (c[0], c[1]) a (c[2], c[3]) a jejich očekávanou vzdálenost (c[4]). Očekávanou vzdálenost jsem převzal z online nástroje [3]. Test transformuje souřadnice c[2], c[3] do x,y,z a porovná vzdálenost vůči počátku.

7.3.2 Konverze dat

Druhou částí systému, pokrytou jednotkovými testy, je konverze dat. Unit testy zde garantují správnost serializace a deserializace zpráv a jejich případné selhání upozorní na změnu API. Při inicializaci testovací třídy dojde k vytvoření několika instancí zpráv. Ty jsou následně v jednotlivých testech postupně převedeny do formátu JSON a poté zpět. Každé převedení je doplněno několika kontrolami, že provedená akce byla úspěšná.

```
[Test]
public void BatchMessageConversion()
{
    string messageJson = serialize(batch);

    Assert.IsNotNull(messageJson);

    Message message = deserialize(messageJson);
    BatchMessage batchConverted = message as BatchMessage;

    Assert.IsNotNull(message);
    Assert.IsInstanceOf(typeof(BatchMessage), batchConverted);
    Assert.IsNotNull(batch.content);
}

[Test]
public void ListMessageConversion()
{
    foreach (var missionMessage in messageList) {
```

[3] https://www.gpsvisualizer.com/calculators

```
        string missionJson = serialize(missionMessage);

        Assert.IsNotNull(missionJson);

        Message message = deserialize(missionJson);
        SoldierListMessage missionConverted =
            message as SoldierListMessage;

        Assert.IsNotNull(message);
        Assert.IsInstanceOf(
            typeof(SoldierListMessage),
            missionConverted);
        Assert.IsNotNull(missionConverted.payload);
    }

}

[Test]
public void MissingMessageCodeConversion()
{
    Message message = deserialize("{}");
    Assert.IsNull(message);
}
```

Obrázek 7.2. Jednotkové testy pro konverzi dat

7.4 Uživatelské testování

Závěrečnou testovací fází je uživatelské testování. Cílem uživatelského testování bylo zjistit použitelnost aplikace uživateli různorodých skupin.

Testování proběhlo na připravených statických datech. Statická data byla použita z důvodu nenasazení serverové aplikace a nemožnosti využít domácí síť ve venkovním prostředí. Data obsahovala pozice v lokalitě Praha 10, Malešice. Pro každý scénář byla použita trochu odlišná data s odlišnými pozicemi a stavy tak, aby vyhověla předepsaným scénářům, a aby uživatel nemohl spoléhat na předchozí znalost pozic. Tato data byla rozdělena do misí s názvy *Scénář 1* a *Scénář 2*. Pro účely testování byl 3D válec viditelný, aby bylo možné lépe rozpoznat hledanou pozici.

Jako zařízení byl použit telefon Samsung Galaxy S22 s rozlišením 2340 x 1080 (FHD+), osmi jádrovým procesorem a 8 GB operační paměti [4].

Před začátkem testování byla aplikace uživatelům v rychlosti popsána a byli uvedeni do role záchranáře snažícího se lokalizovat raněného kolegu. Po ukončení testování uživatelé vyplnili krátký dotazník.

7.4.1 Účastníci

Testování se zúčastnilo celkem pět lidí. Účastníci byli vybráni tak, aby byly obsaženy různé věkové skupiny s různým vztahem k technice. Byť většinou bývá personál zaškolen k používání moderních technologických nástrojů, je dobré otestovat různorodou uživatelskou skupinu. Jednotliví účastníci jsou představeni níže.

[4] https://www.samsung.com/cz/smartphones/galaxy-s22/specs/

- **Účastník 1**
  Pohlaví: Muž
  Věková skupina: 20-30
  Vztah k technologiím: Kladný
  Zkušenost s AR: Uživatel v minulosti hrál AR hru PokemonGo, jinak žádná.
  První účastník se pohybuje v IT a technologiím se věnuje profesně.
- **Účastník 2**
  Pohlaví: Muž
  Věková skupina: 20-30
  Vztah k technologiím: Kladný
  Zkušenost s AR: Uživatel v minulosti hrál AR hru PokemonGo, jinak žádná.
  Druhý účastník je členem Policie ČR. K technologiím má kladný vztah, ale nemá žádné hlubší technologické znalosti.
- **Účastník 3**
  Pohlaví: Žena
  Věková skupina: 20-30
  Vztah k technologiím: Kladný
  Zkušenost s AR: Uživatelka párkrát využila aplikaci na pozorování souhvězdí na noční obloze.
  Třetím účastníkem je žena, absolventka FSV ČVUT. K technologiím má kladný vztah, opět bez hlubších znalostí.
- **Účastník 4**
  Pohlaví: Muž
  Věková skupina: 50-60
  Vztah k technologiím: Kladný
  Zkušenost s AR: Uživatel občasně využije aplikaci na pozorování souhvězdí na noční obloze.
  Čtvrtý účastník reprezentuje skupinu ve věku 50-60 let. Účastník se pohybuje v IT a technologiím se věnuje profesně.
- **Účastník 5**
  Pohlaví: Žena
  Věková skupina: 50-60
  Vztah k technologiím: Spíše kladný
  Zkušenost s AR: Žádná
  Posledním účastníkem je žena. Technologie využívá pravidelně, hlavně v zaměstnání, ale nemá k nim hlubší vztah.

7.4.2 Průběh

V této sekci bude popsán průběh testování jednotlivými uživateli, pozorované problémy a závěrečné hodnocení. Testování se všemi účastníky proběhlo ve dne, za dobrých světelných podmínek. Směrové ukazatele měly na začátku každého testování vypnutou možnost výpisu textové vzdálenosti, aby se otestovalo jejich škálování pro rozlišení vzdálenosti.

Po ukončení testování účastníci ohodnotily celkový dojem a intuitivnost jednotlivých kroků známkami 1-5.

- **Účastník 1**
  Při testování obou scénářů nedošlo k žádným komplikacím. Po zvolení mise a několika pohybech telefonem uživatel rychle pochopil chování směrových ukazatelů. Během testování prvního scénáře dokonce začal zkoumat nastavení a

jednotlivé režimy chování ukazatelů. Uživatel poměrně hravě zvládl oba testované scénáře, bez jakéhokoliv problému rozpoznat stav i vzdálenost hledaných pozic. Zobrazení pozic na průhledné mapě se mu zdálo poměrně nevýrazné.

Hodnocení

Vzhled a dojem: 3

Intuitivnost: 2

Uživatelské rozhraní aplikace shledal účastník jako jednoduché a nevýrazné. Práce s ním mu přišla přímočará a intuitivní.

- **Účastník 2**

  U druhého účastníka se objevily drobné problémy při spuštění první mise. Po spuštění se směrové ukazatele začaly přemísťovat bez jakékoliv rotace zařízením. Po restartování telefonu a následném pomalém otáčení a uražení malé vzdálenosti s telefonem tyto problémy zmizely. Proč tyto problémy nastaly se mi nepodařilo zjistit. Očekávám, že došlo ke špatnému čtení prostředí AR frameworkem. Průběh obou scénářů proběhl bez dalších komplikací. Účastník správně rozlišil barvy označující stavy a rozpoznal škálování směrových ukazatelů.

  Hodnocení

  Vzhled a dojem: 2

  Intuitivnost: 1

  Aplikace se účastníkovi líbila a přišla mu přínosná. Líbila se mu zejména možnost využití mapy pro rychlejší lokalizaci. Ocenil by ale označení směru, kterým se dívá na mapě, a možnost zobrazit relevantní objekty.

- **Účastník 3**

  U třetího účastníka došlo k lehkému zmatení po vykreslení směrových ukazatelů. Ještě před mojí intervencí do testování a vysvětlením, účastnice zhruba pochopila jejich význam. U zobrazené mapy nemohla účastnice rozlišit svoji polohu a polohu ostatních. Zbytek obou scénářů proběhl bez dalších komplikací. Význam barvy stavu i zvětšující ho ukazatele rozpoznala účastnice správně.

  Hodnocení

  Vzhled a dojem: 3

  Intuitivnost: 3

  Výslednou aplikaci shledala účastnice jako zajímavou. Chybělo jí podrobnější uvedení do funkcí aplikace, aby se jí lépe ovládala a věděla přesně, co má dělat.

- **Účastník 4**

  I zde došlo k lehkému zmatení při vykreslení směrových ukazatelů po spuštění první mise. Pro navigaci a lokalizaci cílových bodů uživatel využíval raději mapu. V mapě a jejích vykreslených bodech neměl problém se orientovat. Označení raněného člena spíše odhadoval. Po dosažení cílové polohy došlo k několika zásekům, které brzy odezněly. Uživatel neregistroval změnu velikosti ukazatelů. Po mé intervenci a zapnutí textové vzdálenosti u jednotlivých ukazatelů bylo uživateli vše jasnější.

  Hodnocení

  Vzhled a dojem: 1

  Intuitivnost: 2

  Výsledně shledal účastník aplikaci poměrně intuitivní. Líbilo se mu minimalistické uživatelské rozhraní a jednoduchý design.

- **Účastník 5**

  Poslední účastnice testování zpočátku nevěděla, jak ovládání aplikace uchopit vzhledem k absenci zkušenosti s technologiemi rozšířené reality. Účastnici jsem

musel poměrně podrobně vysvětlit principy chování indikátorů a jednotlivých funkcionalit. I poté účastnice spíše odhadovala, který objekt lokalizovat. Opět zde pomohlo zapnutí textové vzdálenosti k jednotlivým bodům.

Hodnocení

Vzhled a dojem: 2

Intuitivnost: 4

Účastnice výslednou aplikaci neshledala příliš intuitivní. Avšak celkový vzhled na ní působil dobře.

Obrázek 7.3. Snímky z testování

7.4.3 Výsledky

Většina uživatelů pochopila chování aplikace hned nebo po krátké chvíli. Zásadním faktorem je zde, podle mého soudu, pohyb a sledování chování ukazatelů při různých pohybech zařízením, rotací, nebo uražením několika metrů. Dalšími pozorovanými nedostatky byly drobné chyby zobrazování uživatelského rozhraní při změně rotace zařízení.

Průměrné hodnocení:

- **Vzhled a dojem:** 2.2
- **Intuitivnost:** 2.4

Výsledné hodnocení považuji za dobré s prostorem pro zlepšení.

Pozorované problémy:

- **Výkon**

  Během testování jsem pozoroval zmíněný problém s přemisťováním ukazatelů a drobné zásek, když se na scéně objevili vytvořené 3D objekty.
- **Špatné zobrazení 3D objektů**

  Opakujícím se problémem bohužel byla špatná pozice 3D objektů na určeném místě. Pozoroval jsem, že dochází k rozdílům ve čtených hodnotách nadmořské výšky zařízení. Rozdíl v řádech jednotek metrů způsobí, že objekt není vidět. Pokud je měřená nadmořská výška větší než vizualizovaná hodnota, dojde k vytvoření objektu pod úrovní terénu. Viditelnost objektu pak znemožní detekovaná plocha s okluzním materiálem.

  Jako částečné řešení se nabízí využití terénních kotev z knihovny ARCore. Toto řešení je limitováno použitím vně budov, jelikož nadmořská výška vytvořené kotvy odpovídá výšce terénu.

  Druhým možným řešením pro použití venku je otestovat plochy pomocí raycastingu. Vyslat paprsek směrem k transformovaným souřadnicím a otestovat body dotyku. Jednotlivé body dotyku by mohly být otestovány na vzdálenost k cíli a nejbližší plocha by mohla sloužit jako kotva pro 3D objekt.
- **Směrový ukazatel je vidět zároveň s 3D objektem**

  Pokud je vidět 3D objekt, měl by zůstat směrový ukazatel skrytý. Tak tomu nebylo ve všech případech a ukazatel byl přichycen ke straně obrazovky.
- **Drobné problémy v UI**

  Pozice tlačítek na hlavní obrazovce nebyla správně responzivní. Při překlopení telefonu do horizontální polohy byly tlačítka špatně zobrazena. Po zobrazení mapy a následném přechodu do obrazovky s nastavením, mapa překrývala zobrazenou obrazovku.

7.5 Finální úpravy

Tato sekce popisuje implementační opravu chyb nalezených během uživatelského testování, či testování prototypu a zapracování připomínek uživatelů, nebo z mého pozorování.

7.5.1 Výkon

Problémy s výkonem se objevili při uživatelském testování. Pokud se na scéně objevil 3D objekt začalo docházet k malým zásekům. Problém jsem se pokusil vyřešit následovně. Logiku upravování pozic 2D ukazatelů jsem omezil tak, aby se volala zhruba třicetkrát za snímek a upravil jsem podmínky pro testování zorného pole kamery, případně *raytracingu* ke zjištění viditelnosti 3D objektu.

```
private float timer = 0.0f;
private float updateInterval = 1f / 30f; // 30 FPS
// Update is called once per frame
```

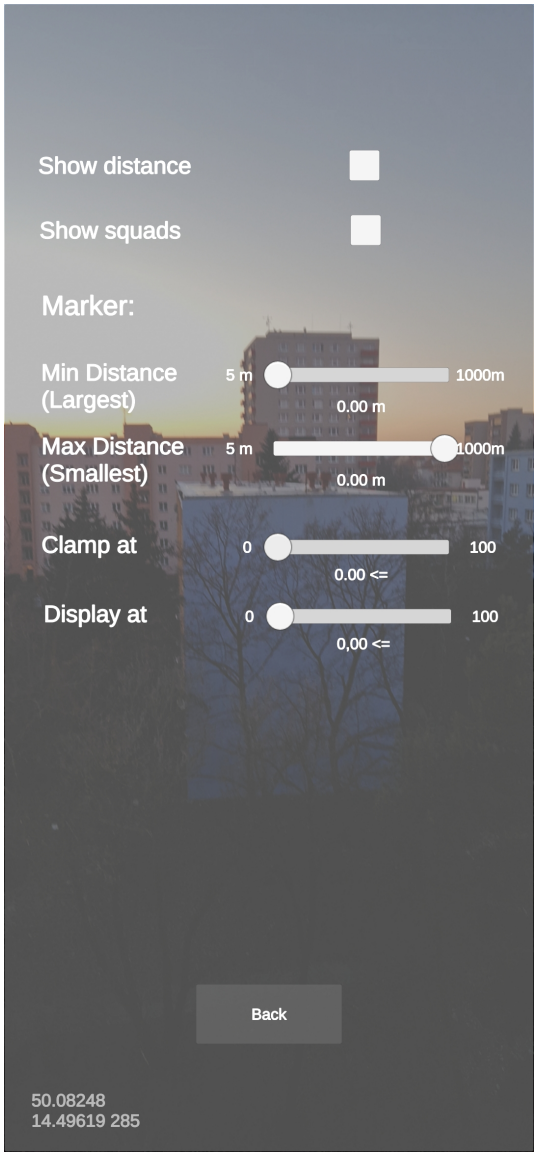
```
void Update()
{
    timer += Time.deltaTime;

    if (timer >= updateInterval)
    {
        operatives.ForEach(operative => {
            UpdateWaypointPosition(operative);
        });
        timer = 0f;
    }
}
```

Obrázek 7.4. Ukázka upraveného kódu

Toto řešení pomohlo zredukovat míru záseků. Dalším možným řešením je směrové ukazatele schovat pokud se cíl nachází v určitém okruhu od uživatele, tedy otestovat pouze vzdálenost. Pak by nebylo nutné provádět výpočetně náročnější operace jako testování rovin a raytracing.

7.5.2 Nastavení

Přidání možnosti sledovat celé skupiny místo jednotlivců byla již zmíněna v kapitole Implementace. Další úpravou je přidání možnosti výběru hodnoty **mls**, při které se budou ukazatele přichytávat ke straně displeje, nebo se nebudou vůbec zobrazovat. Díky tomuto nastavení si uživatele vybere, zda-li chce sledovat všechny členy mise, či pouze členy se zhoršeným stavem pro jejich snadnější lokalizaci. Dále pak displej bude obsahovat méně značek, tedy bude obsahovat méně rušivých elementů při práci v terénu. Výsledné možnosti v nastavení jsou zobrazeny na obrázku 7.5. Jedná se o přepínač pro zobrazení pozic, nebo jednotek a posuvníky pro nastavení hranic škálování ukazatele a nastavení hranice **mls** pro filtrování.

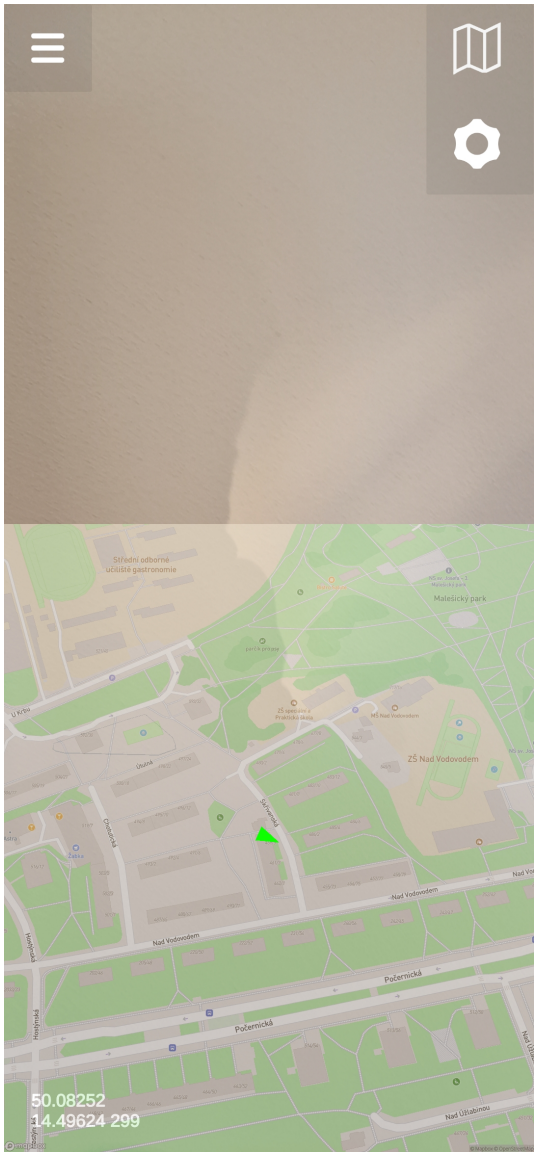


Obrázek 7.5. Upravené možnosti nastavení

7.5.3 Mapa

Označení uživatele a ostatních členů mise pomocí špendlíkových ukazatelů v kombinaci s transparentností mapy se ukázalo poměrně nevýrazné. Dalším bodem mého pozorování je umístění pozice uživatele uprostřed mapy. Pokud nejsou zobrazení účastníci rozprostřeni okolo uživatele, není využita většina plochy mapy.

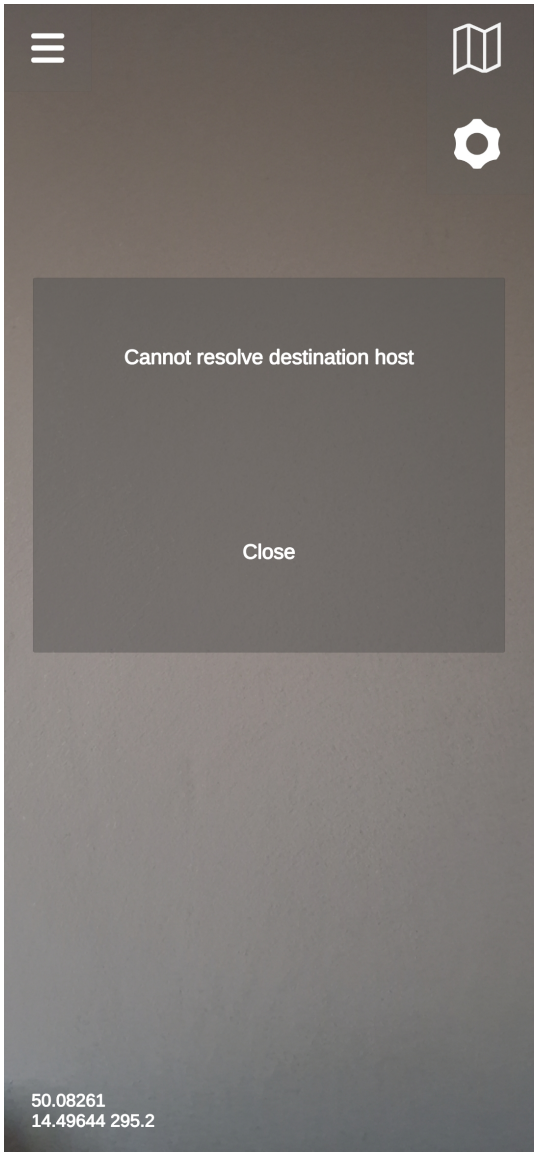
Upravené řešení spočítá geografické hranice mapy tak, aby byly zobrazeny všechny ukazatele. Uživatel je zobrazen ikonou trojúhelníku, který je rotován v závislosti na natočení kompasu. Členové mise jsou reprezentování podobně jako na směrových ukazatelích. Jejich poloha na mapě je spočtena pomocí WebMerkator projekce.



Obrázek 7.6. Upravené zobrazení mapy

7.5.4 Uživatelské rozhraní

V uživatelském rozhraní bylo opraveno pár drobných chyb. Umístění ikon na hlavní obrazovce neodpovídalo při překlopení telefonu do horizontální polohy. Zobrazená mapa překrývala obrazovku nastavení, i když měla být transparentně pod ní. V poslední řadě došlo na drobné vizuální úpravy panelů a tlačítek s cílem zlepšit vizuální stránku aplikace.



Obrázek 7.7. Dialog pro chybové hlášky

Kapitola 8
Závěr

Cílem diplomové práce bylo navrhnout a implementovat prototyp aplikace rozšířené reality, která vizualizuje pozice a stavy členů integrovaného záchranného systému. Prototyp byl původně určen pro brýle rozšířené reality XReal. Jelikož se brýle nepodařilo obstarat, byl prototyp implementován na platformu Android.

Práce popsala technologii rozšířené reality, její základní principy, zařízení a vybrané nástroje pro práci s rozšířenou realitou. Ve třetí kapitole byly představeny dva evropské projekty, které se zabývají podobnou problematikou. Představené projekty jsou projekty RESPOND-A a FASTER. Jejich cílem je zlepšení efektivity a bezpečnosti záchranářů při krizových situacích zejména pomocí technologie rozšířené reality.

V analytické části byly definovány požadavky implementovaného prototypu a představeno komunikační rozhraní již připravené serverové aplikace a představen systém FlexiGuard. Dále byla vysvětlena projekce geografických souřadnic do Kartézského systému a WebMerkator projekce pro transformaci souřadnic do statických map. Závěr analytické části tvoří výběr použitých technologií.

Navazující část představila návrh implementovaného řešení. Byl vytvořen lo-fi prototyp, zachycující klíčové aspekty systému a uživatelského rozhraní. Těmi jsou ovládací rozhraní, směrové indikátory ve formě HUD, a 3D objekty zobrazující stav jednotlivých členů mise. 3D objekty byly zasazeny na obdržených souřadnicích a zasazeny do reálného světa pomocí rozšířené reality. Návrh systému byl ilustrován na komponentovém diagramu, který dělí systém na několik modulárních částí. V neposlední řadě byla představena architektura systému pomocí vzoru MVC.

V implementační části byla představena implementace jednotlivých funkcionalit pro splnění požadavků z analýzy. Zobrazení pozic bylo implementováno trojím způsobem. Prvním bylo HUD řešení s ukazateli pro jednotlivé členy a jednoduchou navigaci. Druhým způsobem bylo umístění virtuálních 3D objektů v rozšířené realitě. Posledním způsobem bylo pak označení členů na statické 2D mapě.

Závěr práce tvoří testování. Prototyp byl testován metodou kognitivního průchodu. Výsledná aplikace má kritické části pokryté sadou jednotkových testů a prošla uživatelským testováním. Pozorované problémy a některé připomínky uživatelů byly opraveny.

Výsledná aplikace splňuje zadané cíle a požadavky. Zobrazuje data, poskytovaná serverem, o pozicích a stavech členů mise záchranného systému a vizualizuje je pomocí technologie rozšířené reality.

Literatura

[1] David Saurı a others. *Mapping the impacts of recent natural disasters and technological accidents in Europe*. Office for official publications of the European Communities, 2003.

[2] Fatih Demir, Salman Ahmad, Prasad Calyam, Duo Jiang, Rui Huang a Isa Jahnke. A Next-Generation Augmented Reality Platform for Mass Casualty Incidents (MCI). *Journal of Usability Studies*. 2017, 12 (4).

[3] Harbil Arregui, Eider Irigoyen, Iñaki Cejudo, Sebastian Simonsen, Drazen Ribar, Michail-Alexandros Kourtis, Yannis Spyridis, Natalia Stathakarou a Michael C. Batistatos. *An Augmented Reality Framework for First Responders: the RESPOND-A project approach*. 2022.

[4] Dhiraj Amin a Sharvari Govilkar. Comparative study of augmented reality SDKs. *International Journal on Computational Science & Applications*. 2015, 5 (1), 11–26.

[5] Julie Carmigniani a Borko Furht. *Augmented Reality: An Overview*. 2011.

[6] Mehdi Mekni a Andre Lemieux. Augmented reality: Applications, challenges and future trends. *Applied computational science*. 2014, 20 205–214.

[7] Jacky Cao, Kit-Yung Lam, Lik-Hang Lee, Xiaoli Liu, Pan Hui a Xiang Su. Mobile augmented reality: User interfaces, frameworks, and intelligence. *ACM Computing Surveys*. 2023, 55 (9), 1–36.

[8] Leonel Merino, Magdalena Schwarzl, Matthias Kraus, Michael Sedlmair, Dieter Schmalstieg a Daniel Weiskopf. *Evaluating Mixed and Augmented Reality: A Systematic Literature Review (2009-2019)*. 2020. `http://dx.doi.org/10.1109/ISMAR50242.2020.00069`.

[9] Marc Riar, Jakob J Korbel, Nannan Xi, Rudiger Zarnekow a Juho Hamari. The use of augmented reality in retail: a review of literature. 2021.

[10] *Overview, Features, and Specs | Microsoft HoloLens*. `https://www.microsoft.com/en-us/hololens/hardware`.

[11] *Xreal Glasses - NRSDK*. `https://xreal.gitbook.io/nrsdk/nrsdk-fundamentals/xreal-devices/readme`.

[12] Micheal Lanham. *Learn ARCore-Fundamentals of Google ARCore: Learn to build augmented reality apps for Android, Unity, and the web with Google ARCore 1.0*. Packt Publishing Ltd, 2018.

[13] *Geospatial Anchors*. `https://developers.google.com/ar/develop/unity-arf/geospatial/anchors`.

[14] *Global Localization with VPS*. `https://developers.google.com/ar/develop/geospatial##global_localization_with_vps`.

[15] *Tracking Geographic Locations in AR.* `https://developer.apple.com/documentation/arkit/arkit_in_ios/content_anchors/tracking_geographic_locations_in_ar`.

[16] *Getting Started | Vuforia Library.* `https://developer.vuforia.com/library/`.

[17] Anastasios Dimou, Dimitrios G Kogias, Panagiotis Trakadas, Fabio Perossini, Maureen Weller, Olivier Balet, Charalampos Z Patrikakis, Theodore Zahariadis a Petros Daras. *FASTER: First Responder Advanced Technologies for Safe and Efficient Emergency Response*. Springer, 2021.

[18] Patrik Kutilek, Petr Volf, Slavka Viteckova, Pavel Smrcka, Vaclav Krivanek, Lenka Lhotska, Karel Hana, Radek Doskocil, Leos Navratil, Zdenek Hon a Alexandr Stefek. *Wearable systems for monitoring the health condition of soldiers: Review and application*. 2017.

[19] Radim Kliment, Pavel Smrcka, Karel Hana, Jakub Schlenker, Vladimir Socha, Lubos Socha, Patrik Kutilek a others. Wearable modular telemetry system for the integrated rescue system operational use. *Journal of Sensors*. 2017, 2017

[20] Jakub Schlenker, Vladimír Socha, Pavel Smrčka, Kareł Hána, Vladimír Begera, Patrik Kutílek, Zdeněk Hon, Jan Kašpar, Lukáš Kučera, Jan Mužík, Tomáš Veselý a Martin Vítězník. FlexiGuard: Modular biotelemetry system for military applications. 2015, 1-6. DOI 10.1109/MILTECHS.2015.7153712.

[21] Andy Neumann, Nuno Laranjeiro a Jorge Bernardino. An Analysis of Public REST Web Service APIs. *IEEE Transactions on Services Computing*. 2021, 14 (4), 957-970. DOI 10.1109/TSC.2018.2847344.

[22] Ian Fette a Alexey Melnikov. *The websocket protocol.* .

[23] Sameer Kumar a Kevin B Moore. The evolution of global positioning system (GPS) technology. *Journal of science Education and Technology*. 2002, 11 59–80.

[24] James A Slater a Stephen Malys. *Wgs 84—past, present and future*. Springer, 1998.

[25] George P Gerdan a Rodney E Deakin. Transforming Cartesian coordinates X, Y, Z to Geographical coordinates φ, λ, h. *The Australian Surveyor*. 1999, 44 (1), 55–63.

[26] Fritz C Kessler, Sarah E Battersby, Michael P Finn a Keith C Clarke. Map projections and the Internet. *Choosing a Map Projection*. 2017, 117–148.

[27] S Lumley a RENEE SIEBER. Web Maps for Global Data Visualization: Does Mercator Matter? *Spatial Knowledge and Information Canada*. 2019, 7 (1), 5.

[28] Emmanuel Stefanakis. Web mercator and raster tile maps: two cornerstones of online map service providers. *Geomatica*. 2017, 71 (2), 100–109.

[29] Sten J Claessens. Efficient transformation from Cartesian to geodetic coordinates. *Computers & Geosciences*. 2019, 133 104307.

[30] Nitin R Chopde a Mangesh Nichat. Landmark based shortest path detection by using A* and Haversine formula. *International Journal of Innovative Research in Computer and Communication Engineering*. 2013, 1 (2), 298–302.

[31] Hagar Mahmoud a Nadine Akkari. *Shortest path calculation: a comparative study for location-based recommender system*. In: *2016 world symposium on computer applications & research (WSCAR)*. 2016. 1–5.

[32] Scott Chacon a Ben Straub. *Pro git*. Springer Nature, 2014.

[33] *Developer Survey 2022*. `https://survey.stackoverflow.co/2022/##technology-version-control`.

[34] Mykhailo Bevz. Exploring the Use of Color as an Informational Layer in Video Game Design. 2023.

[35] James William Cooper. Java design patterns: a tutorial. 2000.

[36] Unity Technologies Technologies. *Build a modular codebase with MVC and MVP Programming Patterns*. `https://unity.com/how-to/build-modular-codebase-mvc-and-mvp-programming-patterns`.

[37] M Qureshi a Fatima Sabir. A comparison of model view controller and model view presenter. *arXiv preprint arXiv:1408.5786*. 2014.

[38] Manisah Mohd Shah, Haslina Arshad a Riza Sulaiman. *Occlusion in augmented reality*. In: *2012 8th International Conference on Information Science and Digital Content Technology (ICIDT2012)*. 2012. 372-378.

[39] Fabrizio Espındola a Pablo Yeber. *The Unity Shader Bible*. Jettelly, 2022.

[40] Jakob Nielsen. How to conduct a heuristic evaluation. *Nielsen Norman Group*. 1995, 1 (1), 8.

[41] Clayton Lewis a Cathleen Wharton. *Cognitive walkthroughs*. Elsevier, 1997.

[42] John Rieman, Marita Franzke a David Redmiles. *Usability evaluation with the cognitive walkthrough*. 1995.

Příloha A
Zkratky

AR	Rozšířená realita
3D	Trojrozměrný
2D	Dvourozměrný
HUD	Head-up display
NFC	Čip pro bezdrátovou, rychlou a zabezpečenou výměnu dat na vzdálenost do 4 cm
Lo-fi	Low-fidelity