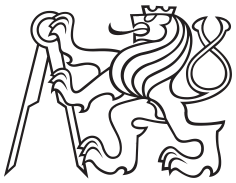
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Aplikace pro zobrazování pozic členů týmu v chytrých brýlích

Matěj Mužátko

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

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Mužátko** Jméno: **Matěj** Osobní číslo: **483809**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra měření**

Studijní program: **Otevřená informatika**

Specializace: **Počítačové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Aplikace pro zobrazování pozic členů týmu v chytrých brýlích

Název diplomové práce anglicky:

Application for display of team members' position in smart glasses

Pokyny pro vypracování:

Navrhněte a implementujte aplikaci pro zobrazování pozic členů týmu v chytrých brýlích nReal. Aplikace bude přijímat ze serveru informace o pozici členů týmu včetně dalších údajů a zobrazovat je v zorném poli uživatele. Aplikace bude umožňovat zobrazení různých módů s informacemi o členech týmu. Aplikace bude ovládána standardním ovládacím zařízením dodávaným k brýlím nReal. Aplikaci otestujte sadou vhodných testů.

Seznam doporučené literatury:

[1] Agrawal, A., & Cleland-Huang, J. (2021). RescueAR: Augmented Reality Supported Collaboration for UAV Driven Emergency Response Systems. arXiv preprint arXiv:2110.00180.
[2] 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.
[3] 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.

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: **16.02.2024** Termín odevzdání diplomové práce: _____

Platnost zadání diplomové práce:
do konce letního semestru 2024/2025

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

Děkuji svým rodičům za podporu nejen při tvorbě práce, ale i během celého studia.
Děkuji své partnerce za bezmeznou podporu v těžkých chvílích při tvorbě práce.
Konečně děkuji vedoucímu práce, panu Ing. Miroslavu Burešovi, za možnost pracovat na zajímavém projektu a získat cenné zkušenosti se zcela novou platformou.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a v souladu s Metodickým pokynem o dodržování etických principů pro vypracování závěrečných prací, a že jsem uvedl všechny použité informační zdroje.

V Praze, 24. května 2024

Matěj Mužátko

Abstrakt

Tato práce se zaměřuje na tvorbu softwarové architektury pro aplikaci určenou k zobrazování členů týmu v chytrých brýlích a implementaci funkčního prototypu za použití této architektury na platformě chytrých brýlí NReal a jejich vývojové sady NRSDK. Je navržena architektura s několika možnostmi její implementace ve skutečném prototypu. Architektura je připravena na změny v budoucnu, například na vývoj speciálního hardwaru vytvořeného pro tuto aplikaci. Tři varianty nasazení této aplikace byly implementovány a byly zmíněny silné a slabé stránky vytvořeného prototypu.

Klíčová slova: chytré brýle, smíšená realita, NReal, NRSDK, Unity, rozšířená realita, virtuální realita, sdílení lokací

Vedoucí práce: doc. Ing. Miroslav Bureš, Ph.D.
Praha, Resslova 307/9 (vstup z Karlovo náměstí 13), místnost: E-221

Abstract

This thesis focuses on creating a software architecture for displaying positions of team members in smart glasses, and implementing a functional prototype of such an architecture on NReal glasses platform using its NRSDK development kit. It proposes the architecture with several options how the components can be implemented in real prototype. The architecture is ready for changes in the future, if the specific hardware will be designed for it. Three deployment variants of the architecture have been implemented, the strengths and weaknesses have been pointed out.

Keywords: smart glasses, mixed reality, NReal, NRDSK, Unity, extended reality, virtual reality, location sharing

Title translation: Application for display of team members' position in smart glasses

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

1.1 Motivace

Rozšířená realita (XR) vzniká již od šedesátých let dvacátého století a jedná se o nestále se vyvíjející oblast, zejména od doby rozšíření chytrých telefonů [1]. Jednou z podskupin XR je MR, neboli takzvaná smíšená realita, která umožňuje zobrazit uživateli virtuální informace (např. ve speciálních *chytrých brýlích* [2]), zatímco uživatel může stále interagovat s reálným světem [3]. Smíšená realita nabízí unikátní způsob vizualizace dat v prostoru.

Existují různé systémy pro sdílení geografických pozic (viz 1.1.1, [4]). Zobrazení pozic členů týmu přímo v chytrých brýlích může zajistit lepší přehled při využití těchto systémů a zajistit rychlejší získání dalších potřebných informací – např. místo sledování pozic značek na mapě na mobilním telefonu se může uživatel rozhlédnout kolem sebe a vidět pozice členů a další informace, přičemž může reagovat na podněty v okolním světě lépe, než kdyby se musel věnovat práci s dalším zařízením.

1.1.1 Projekt DTA

Jednou z motivací pro vznik této aplikace je možnost ji napojit na již existující systém DTA, na kterém se podílí i ČVUT v Praze. Projekt DTA (Digital Triage Assistant) vznikl na Johns Hopkins University, jako studentský projekt, a zabývá se vývojem zařízení pro sledování vitálních funkcí vojáků. Systém DTA např Na projektu v současné době spolupracuje i ČVUT v Praze, včetně laboratoře STILL na Katedře počítačů Fakulty elektrotechnické. Zde v současné době vzniká technické řešení, zatímco tým na Johns Hopkins University se zabývá medicinskou stránkou věci. Informace v této sekci byly převzaty ze zdroje [5].

1.2 Cíle práce

Cílem práce je navrhnout softwarovou architekturu pro aplikaci pro zobrazování pozic členů týmu v chytrých brýlích a vyvinout funkční prototyp této aplikace dle vytvořeného návrhu. Prototyp by měl být vyvinut na platformě

brýlí NReal dle zadání, ale architektura by měla být dostatečně flexibilní, připravena na případnou implementaci na jiném zařízení – např. dedikovaném zařízení vytvořeném přímo pro tuto aplikaci.

1.3 Přehled kapitol

V kapitole 2 je nejprve zadání přetvořeno do požadavků, podle kterých je následně navržena architektura aplikace. Popsáno je více úrovní návrhu – jak obecný návrh softwarových komponent, tak i konkrétnější popis různých možných variant těchto komponent. Také je v této kapitole popsán princip převodu souřadnic (v sekci 2.5), včetně zvoleného způsobu převodu úhlových vzdáleností na vzdálenosti v metrech. Zdůrazněna je i nutnost synchronizovat natočení hlavy a GPS souřadnic uživatele chytrých brýlí (sekce 2.8). Tato kapitola popisuje i použitý hardware (sekce 2.6) a použité softwarové nástroje (sekce 2.7), s důrazem na možnosti jejich využití, zejména ty podstatné pro tuto práci. Na závěr kapitoly 2 jsou popsány 4 konkrétní varianty nasazení, je zde již určena konkrétní vazba možných implementovaných komponent na konkrétní hardware použitý v této práci.

Kapitola 3 popisuje implementovaný prototyp. Jedná se především o specifikaci konkrétních aspektů vývoje a popis projektu v Unity, popsaného z hlediska vytvořených scén, objektů a skriptů, které je obhospodařovávají.

Kapitola 4 popisuje způsob testování implementovaného prototypu. Jsou zde definovány jednotlivé testovací scénáře pro manuální testování (sekce 4).

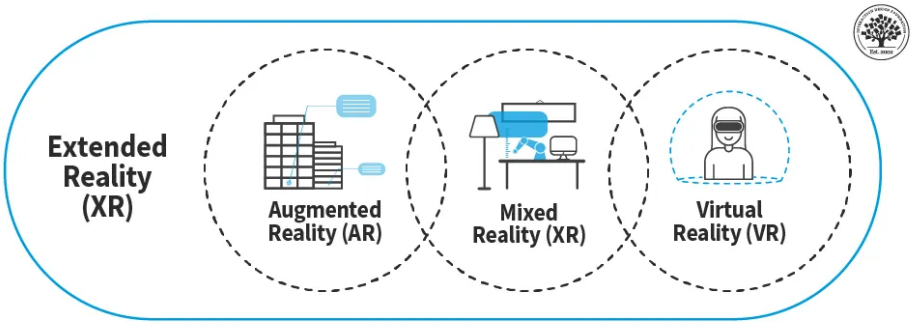
Na závěr, v kapitole 5 je práce shrnuta, jsou popsány jednotlivé výstupy z projektu a je zde vzneseno zamyšlení nad možnými rozšířeními, zlepšeními a dalším postupem týkajícím se tohoto tématu.

1.4 Terminologie

V této sekci je popsána terminologie, která by mohla být v některých případech nejasná či zavádějící.

1.4.1 Oddělení reálného a virtuálního světa

V textu často nastává potřeba popsat prostor, souřadnice v něm, zejména v sekci týkající se převodu souřadnic (2.5). Jelikož může být popisován prostor, ve kterém se nacházejí objekty vykreslované v chytrých brýlích, ale i reálný svět, je zapotřebí tyto termíny oddělit a popsat konkrétní terminologii. V tomto textu je zaveden termín *virtuální svět*, nebo *virtuální prostor*, který popisuje 3D virtuální prostor, do kterého jsou umisťovány objekty a následně v aplikaci zobrazovány v chytrých brýlích. Pro realitu je využíván termín *skutečný svět,* případně *skutečný prostor.* Pro určení souřadnic ve skutečném světě je vždy explicitně zdůrazněno, že se jedná o *geografické souřadnice,* u souřadnic ve virtuálním světě je psáno *virtuální souřadnice.*



Obrázek 1.1: Rozlišení VR, AR, MR a XR [6]

1.4.2 VR, AR, MR a XR

Vzhledem k tomu, že vytvářená aplikace je určena pro chytré brýle, je zapotřebí vysvětlit termíny VR, AR, MR a XR. Kolem nich panují v literatuře nepřesnosti, zejména u rozlišení AR a MR.

Termínem *Virtual reality (VR)* se rozumí technologie, při které je veškerá realita vnímaná uživatelem vytvářena plně digitálně [6]. Přestože se jedná o virtuálně vytvářené prostředí, vyvolává ale u lidí skutečné emocionální reakce [6]. Toho je docíleno např. pomocí speciálních VR brýlí, které nepropouští žádné vizuální podněty z vnějšího prostředí dovnitř, a místo nich na displeji (či displejích) uvnitř zobrazují virtuální svět [7].

Oproti tomu termín *Augmented reality (AR)* popisuje technologii, kdy je již realita vnímaná uživatelem částečně tvořená obrazem okolního světa, který je zpracován a doplněn o digitální objekty [6]. Zde se může jednat např. i o aplikace pro mobilní telefony, které umožňují svým uživatelům prohlížet si na telefonu objekty v kontextu jejich okolního prostředí – to může být užitečné např. pro vzdělávání [8].

Termín *Mixed reality (MR)* je od AR rozdílný v tom, že kromě zobrazení virtuálních objektů v reálném světě umožňuje i interakci reálných objektů s reálnými [6]. Interakcí může být např. přemisťování virtuálních objektů pomocí reálných vlastních rukou, pomocí tzv. *Hand trackingu,* umožněného i v chytrých brýlích NReal použitých v této práci [9]. Jedná se tedy o MR brýle, a proto bude tento termín v textu využíván, ačkoliv nejsou využívány MR funkcionality jako je *Hand tracking*. Obecně bývají v literatuře tyto termíny používány záměnně [10].

Termín *Extended reality (XR)* je termín zaštiťující všechny tři zmíněné termíny, tedy VR, AR i MR, jedná se tedy o obecné označení jakkoliv rozšířené, změněné reality digitálními prostředky [6]. Obrázek 1.1 slouží pro lepší představu rozlišení těchto čtyř termínů.

Kapitola 2
Návrh řešení

2.1 Požadavky

2.1.1 Funkční požadavky

Konkrétní funkční požadavky jsou motivovány několika faktory. Některé požadavky vycházejí přímo ze zadání, jiné ze zadání vycházejí pouze částečně, ostatní jsou přidány navíc oproti zadání, např. pro zpříjemnění práce s aplikací nebo pro demonstrační účely. Přehled funkčních požadavků se nachází v tabulce 2.2.

ID	Popis	Typ požadavku	Priorita
FR01	Aplikace umožní uživateli zobrazit pozice členů týmu v zorném poli uživatele, na pozicích odpovídajících skutečným pozicím.	ZADÁNÍ	Vysoká
FR02	Aplikace umožní uživateli zobrazit přehled pozic členů týmu.	ZADÁNÍ	Vysoká
FR03	Aplikace umožní uživateli určit, kteří členové se budou zobrazovat.	KOMFORT	Střední
FR04	Aplikace umožní uživateli pořídit videozáznam jejího chodu.	DEMONSTRACE	Střední

Tabulka 2.2: Přehled funkčních požadavků

2.1.2 Nefunkční požadavky

Nefunkční požadavky na prototyp jsou specifikovány v tabulce 2.4.

ID	Popis	Důvod	Priorita
NFR01	Využití chytrých brýlí s výdrží baterie výpočetní jednotky alespoň 4 hodiny	Domluveno při zadávání projektu s ohledem na očekávaný případ užití	Vysoká
NFR02	Zobrazení členů týmu v radiusu do 2 km	Počítá se s využitím týmy v danou chvíli soustředěnými na vzdálenostech nižších než 2 km, aplikace neslouží pro navigaci na delší vzdálenosti.	Vysoká
NFR03	Využití průhledných brýlí (smíšená realita)	Pro zachování výhod při použití aplikace je vhodné, aby měl uživatel přímý kontakt s realitou, nikoliv pouze pomocí kamer a displejů	Nízká

Tabulka 2.4: Přehled funkčních požadavků

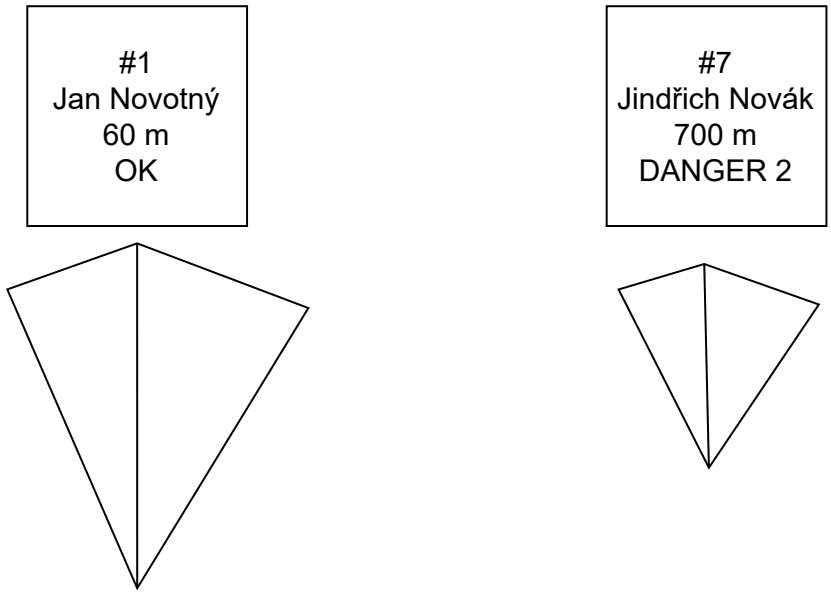
2.2 Návrh uživatelského rozhraní

Součástí aplikačního uživatelského rozhraní jsou 4 režimy, tj. 4 oddělené obrazovky, mezi kterými lze přepínat. Těmito režimy jsou:

- Lokalizační režim (LocalizationMode)
- Režim přehledu (SummaryMode)
- Režim hlášení stavu (StateReportMode)
- Režim videozáznamu (VideoCaptureMode)

2.2.1 Lokalizační režim

Lokalizační režim zobrazuje členy týmu ve smíšené realitě. Členy týmů vyobrazují jehlany rozšiřující se směrem nahoru. U každého jehlanu je zobrazeno id člena, jméno člena, vzdálenost, hlášený stav. S ohledem na co možná nejlepší viditelnost v chytrých brýlích nebyly pro odlišení jehlanů využity různé barvy, byla využita pouze bílá, doporučená jako dobře viditelná barva



Obrázek 2.1: Návrh obrazovky lokalizačního režimu

v dokumentaci NRSDK [11]. Návrh obrazovky lokalizačního režimu se nachází na obrázku 2.1.

2.2.2 Režim přehledu

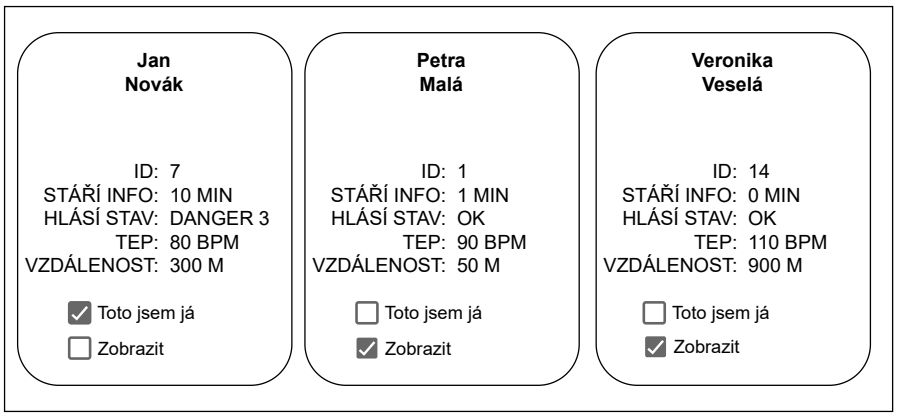
Režim přehledu zobrazuje komplexní přehled přijímaných dat o členech týmu. Zobrazovaná data:

- Jméno
- Lokální id
- Stáří informací
- Hlášený stav
- Tep v BPM
- Vzdálenost od aktuálního uživatele

V tomto režimu je také možné určit, který člen je ve skutečnosti aktuální uživatel pomocí zaškrtávacího boxu *Toto jsem já.*

Původně byla pro zobrazení přehledu zamýšlena tabulka, ale pro konzistenci s *Lokalizačním režimem* a větší přehlednost byl nakonec zvolen návrh pomocí karet, kde každá karta odpovídá jednomu členovi. Manipulace s tabulkou by navíc mohla být obtížná.

Karty jsou uspořádány do horizontální řady a uživatel nimi může procházet pomocí rolování (např. trackpad, závislé na platformě). Návrh obrazovky režimu přehledu se nachází na obrázku 2.2.



Obrázek 2.2: Návrh obrazovky režimu přehledu

2.2.3 Režim hlášení stavu

Režim hlášení stavu umožňuje pomocí čtyř tlačítek nahlásit stav. Z hlediska této aplikace částečně motivované systémem DTA se jedná o zdravotní stav, zjednodušeně nutnost příchodu jiného člena týmu a jeho pomoci. V případě potřeby je ale možné tento stav, tj. z principu pouze číslo 0-3, interpretovat jakkoliv jinak.

Návrh obrazovky režimu hlášení stavu se nachází na obrázku 2.3.

2.2.4 Režim videozáznamu

Pro demonstrační účely je zaveden režim videozáznamu. Ovládání probíhá pomocí malého ovládacího panelu.

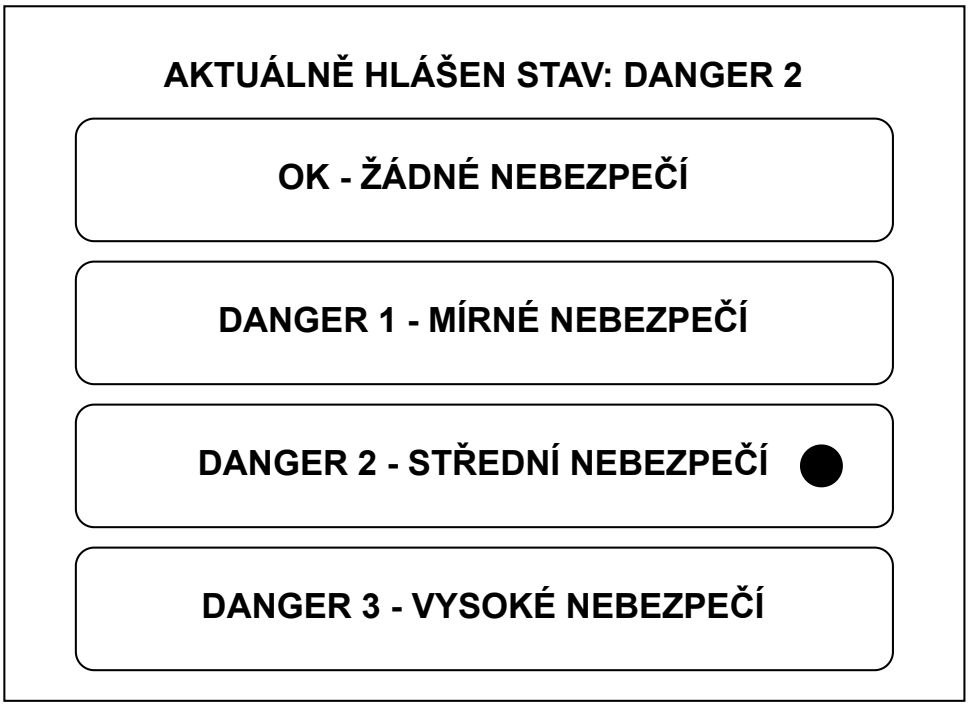
- Kruhové tlačítko slouží k zapnutí, nebo uknočení videozáznamu
- Zaškrtávací políčko *Zahrnout MR objekty* určuje, zda se do nahrávání propíší i virtuální objekty.

Pokud je zaškřtávací políčko *Zahrnout MR objekty* zaškrtnuto, ve videozáznamu se objeví i virtuální objekty. V opačném případě bude snímán pouze obraz z kamery. Ve výchozím stavu je toto políčko zaškrtnuto, jelikož hlavním důvodem zavedení funkcionality nahrávání je zaznamenat chování objektů v MR pro účely demonstrace. Návrh ovládacího panelu režimu videozáznamu se nachází na obrázku 2.4.

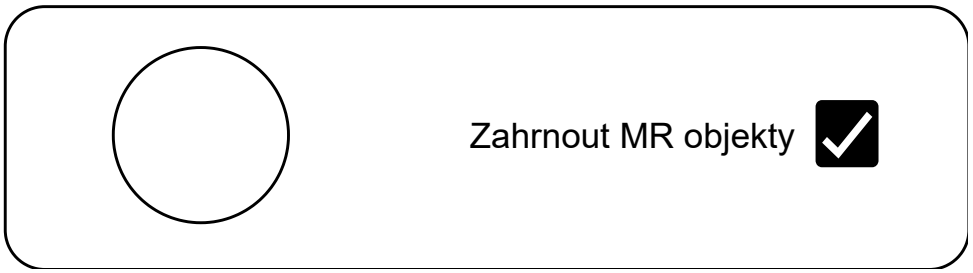
2.2.5 Ovládání aplikace

Ovládání jednotlivých režimů je řešené pomocí prvků grafického rozhraní a je popsáno v samotných sekcích o režimech dříve.

Přepínání mezi režimy není řešeno pomocí prvků grafického rozhraní. Je možné jej implementovat např. pomocí tlačítek samostatného ovládacího zařízení. Dostačující je mít vyhrazené jedno tlačítko pro cyklické přechody mezi režimy.



Obrázek 2.3: Návrh obrazovky režimu hlášení stavu



Obrázek 2.4: Návrh ovládacího panelu režimu videozáznamu

2.3 Model komponent

Systém je navržen tak, aby byl v budoucnu připraven na změny.
Změnou může být např.:

- Změna HW platformy
- Změna SW knihoven
- Napojení na nový systém sdílení pozic
- Napojení na jiný senzor, senzorový systém - kompas apod.

Hardware platforma může být změněna např. na specializovaný hardware navržený přímo pro tuto aplikaci, tj. integrace více komponent přímo do zařízení chytrých brýlí.

Software knihovny mohou být změněny s ohledem na posun ve vývoji MR a chytrých brýlí. Mohou být integrovány nové funkcionality knihoven, výrobci brýlí mohou časem se svými SDK přejít na jinou softwarovou vrstvu, aby umožnili větší unifikaci napříč platformami (přechod na vyšší vrstvu), nebo naopak větší svobodu při práci s konkrétním zařízením (přechod na nižší vrstvu). Např. výrobce brýlí NReal, které jsou použity v této práci se připravuje na podporu vývoje nativních Android aplikací a podporu pro herní engine *Unreal engine,* čímž plánují vytvořit alternativní cestu k vývoji, bez nutnosti využít platformu Unity (více v sekci 2.7.3, [12]).

Aplikaci by pro větší užitečnost mělo být možné snadno přeportovat i na jiné systémy sdílení pozic, než jsou implementovány v této práci. Přestože jednou z motivací pro vznik této aplikace bylo její napojení na systém DTA (viz 1.1.1), jedná se o obecné zadání na modul, který může být využit v širokém množství aplikací – kdekoliv, kde různí členové týmu potřebují navzájem znát své pozice v reálném čase a je jim umožněno využití chytrých brýlí.

Poslední bod může, ale nemusí souviset s celkovou změnou HW platformy. Zdůrazňuje možnost výměny jednotlivých senzorů, nebo změny rozhraní, pomocí kterého jsou připojeny – softwarová a/nebo hardwarová změna.

Ze zmíněných bodů vyplývá, že při návrhu aplikace musí být funkcionalita rozdělena do komponent s dostatečně jemnou granularitou, tak aby:

- běžné změny (např. napojení na jiný systém pozic) mohly být provedeny snadno
- zásadní změny (např. změna HW, SW platforem) mohly být provedeny po částech, odděleně, s možností rozdělení úkolů do týmu více vývojařů

Na základě těchto několika objektiv byly navrženy komponenty znázorněné v tabulkách 2.5 a 2.6 a na obrázku 2.5.

2.3.1 MpsgComponentsConfigurator

Tato komponenta je určena k inicializaci ostatních komponent ve správných variantách dle konfigurace. Jejím úkolem je v současné době především určení správných variant komponent *MemberLocationProvider* a *NorthProvider.*

2.3.2 MemberInfoProvider

MemberInfoProvider je komponenta, která poskytuje pozice členů týmu včetně dalších informací.

2.3.3 ActiveUserInfoProvider

ActiveUserInfoProvider je komponenta odvozená od komponenty MemberInfoProvider, získává pozici a další informace aktuálního uživatele, kterému je

z aktuální instance aplikace posílán obraz do brýlí. To, kdo je *aktuální uživatel* může být určeno např. konfigurací v komponentě *MemberDataManager* (viz sekci 2.3.4), tj. v Režimu přehledu (viz sekci 2.2.2).

Tato komponenta může být implementačně totožná s komponentou *MemberInfoProvider*, ale je návrhem oddělená, aby bylo možné ji propojit jinou, nezávislou cestou – např. pozice ostatních se mohou stahovat méně často než pozice aktuálního uživatele, ale pozice aktuálního uživatele častěji, aby mohlo častěji docházet k resynchronizaci absolutních GPS souřadnic s relativními souřadnicemi chytrých brýlí vůči renderovanému prostoru (viz sekci 2.8).

2.3.4 MemberDataManager

Komponenta *MemberDataManager* využívá komponentu MemberInfoProvider. Periodicky od ní získává informace o členech týmu a spravuje je. Tyto informace pak dává k dispozici ostatním komponentám, tj. jedná se o mezivrstvu mezi komponentou *MemberInfoProvider* a ostatními komponentami. Mimo jiné zajišťuje integritu těchto dat, např. uchování a poskytování informace o tom, který člen je aktuálním uživatelem.

2.3.5 NorthProvider

Komponenta *NorthProvider* poskytuje vektor severu. Orientace virtuálního světa totiž nemusí nutně odpovídat orientaci reálného světa. Ačkoliv brýle mohou mít i magnetometr a všechny potřebné senzory k určení severu, nemusí nutně data z těchto senzorů poskytovat ve svém SDK (viz 2.7.3). Proto může být nutné vytvořit samostatný modul pro určení severu (viz 2.8). Z tohoto důvodu byla v návrhu vytvořena samostatná komponenta *NorthProvider*.

2.3.6 GeolocationCalculationsService

Komponenta *GeolocationCalculationsService* slouží k přepočtům geografických souřadnic na souřadnice ve virtuálním světě. Tato komponenta nutně potřebuje mít znalost geografických souřadnic aktuálního uživatele aplikace, vzhledem ke kterým jsou počítány virtuální relativní souřadnice. Dále je zapotřebí, aby tato komponenta znala aktuální vektor severu, který je možný získat z komponenty *NorthProvider*.

2.3.7 ApplicationStateController

ApplicationStateController je komponenta k uchování a manipulaci stavu aplikace. Je zodpovědná za přechody mezi jednotlivými režimy aplikace (viz 2.2), zprostředkovává komponentě *UserInputHandler* možnost vyvolat přechod na jiný režim.

2.3.8 SettingsController

SettingsController shromažďuje nastavení parametrů. Může být využita například pro uživatelské nastavení některých parametrů, nebo nastavení parametrů před sestavením aplikace. Zároveň mohou ostatní komponenty z této komponenty uložená nastavení získat a přizpůsobit jim své chování.

2.3.9 MemberStateReportManager

Komponenta *MemberStateReportManager* zajišťuje nahlášení stavu uživatele do externího systému. To může být vyvoláno zejména změnou stavu v režimu hlášení stavu, tj. komponentou *StateReportModeController.*

2.3.10 UserInputHandler

Komponenta *UserInputHandler* reaguje na uživatelské vstupy, např. vyvoláním přechodu na jiný režim pomocí komponenty *ApplicationStateController*, nebo změnou nastavení v komponentě *SettingsController*.

2.3.11 LocalizationModeController

Komponenta *LocalizationModeController* zajišťuje správnou činnost aplikace, pokud se nachází v lokalizačním režimu (viz. 2.2.1). Od komponenty *MemberDataManager* získává informace o členech týmů a zobrazuje jejich pozice v rozšířené realitě. Ke zpracování dat využívá komponentu *GeolocationCalculationsService.*

2.3.12 SummaryModeController

Komponenta *SummaryModeController* zajišťuje správnou činnost aplikace, pokud se nachází v režimu přehledu (viz. 2.2.2). Od komponenty *MemberDataManager* získává informace o členech týmů a zobrazuje jejich přehledový výpis. Ke zpracování dat (získání vzdálenosti, azimutu) využívá komponentu *GeolocationCalculationsService.*

2.3.13 StateReportModeController

Komponenta *StateReportModeController* zajišťuje správnou činnost aplikace, pokud se nachází v režimu hlášení stavu (viz. 2.2.3). Využívá komponentu *MemberStateReportManager* (viz sekci 2.3.9), aby byla provedená změna viditelná globálně u všech případných členů.

2.3.14 VideoCaptureModeController

Komponenta *VideoCaptureModeController* zajišťuje správnou činnost aplikace, pokud se nachází v režimu videozáznamu (viz. 2.2.4).

2.3.15 RenderingEngine

Externí komponenta *RenderingEngine* zaštiťuje vykreslení objektů ve 3D, manipulaci s nimi. Zároveň se stará o správu a přepínání scén na cílové platformě.

Může se jednat např. o nějaký herní engine (Unity, Unreal), nebo nativní knihovnu pro cílovou platformu (Android, vlastní platforma).

2.3.16 Display

Komponenta *Display* se stará o zobrazení výsledného obrazu v chytrých brýlích. Z hlediska této práce se o externí komponentu, v budoucnu se může stát interní, pokud bude vyvinut specializovaný hardware.

2.3.17 UserInputListener

UserInputListener je další externí komponenta. Monitoruje uživatelské vstupy, zajišťuje vyvolání odpovídajících reakcí v komponentě *UserInputHandler*. V této práci se jedná o součást vývojové sady od výrobce chytrých brýlí, ale může se jednat i o externí vstupní zařízení.

2.4 Varianty komponent

V této sekci jsou popsány konkrétní varianty jednotlivých komponent, které jsou součástí prototypu implementovaného v rámci této práce (sekce 2.3 popisuje pouze obecný návrh komponent).

2.4.1 MemberInfoProvider

Komponenta *MemberInfoProvider* je navržena v následujících variantách:

- FakeStaticMemberInfoProvider
- FakeDynamicMemberInfoProvider
- MpsgWebMemberInfoProvider
- DtaMemberInfoProvider

FakeStaticMemberInfoProvider

Tato komponenta dodává falešné pozice nehybných členů týmu. Je vhodná pro základní testy spojené se správným vykreslováním pozic.

FakeDynamicMemberInfoProvider

Tato komponenta dodává falešné pozice členů týmu, které se ale mění. Je vhodná pro pokročilejší testy, např. s využitím předem nahraných tras (více v sekci 2.9.2). Tuto komponentu je možné řešit tak, aby brala v potaz skutečnou polohu aktuálního uživatele z komponenty *ActiveUserInfoProvider*, nebo aby se pozice kamery na začátku přizpůsobila polohám ostatních členů týmu a poté zůstala nehybná.

MpsgWebMemberInfoProvider

Tato komponenta již získává data z testovací aplikace *MpsgWeb*. Testovací aplikace je jednoduchý php server, data z něj jsou získávány pomocí protokolu HTTP, více implementačních detailů v sekci 3.4.

DtaMemberInfoProvider

Tato komponenta získává pozice členů týmu ze systému DTA pomocí protokolu WebSocket. Jedná se o pasivní komponentu, která pouze naváže websocketové spojení a vyčkává na příjem dat. Přijatá data uchovává a poskytuje *MemberDataManageru*.

2.4.2 ActiveUserInfoProvider

Komponenta *ActiveUserInfoProvider* je navržena v následujících variantách:

- FakeStaticActiveUserInfoProvider
- MpsgWebActiveUserInfoProvider
- DtaActiveUserInfoProvider

FakeStaticActiveUserInfoProvider

Tato komponenta poskytuje falešnou neměnnou pozici aktuálního uživatele.

MpsgWebActiveUserInfoProvider

Tato komponenta poskytuje pozici aktuálního uživatele získanou z testovací aplikace *MpsgWeb* pomocí protokolu HTTP. K určení aktuálního člena je využita komponenta *SettingsController*.

DtaActiveUserInfoProvider

Tato komponenta poskytuje pozici aktuálního uživatele získanou ze systému DTA pomocí protokolu WebSocket. Jedná se o pasivní komponentu, která naváže websocketové spojení nebo využije stávající, pokud existuje (již navázáno komponentou *DtaMemberInfoProvider*), a vyčkává na příjem dat. Přijatá data uchovává a poskytuje ostatním komponentám.

2.4.3 NorthProvider

Komponenta *NorthProvider* je navržena v následujících variantách:

- FakeStaticNorthProvider
- IotNorthProvider

FakeStaticNorthProvider

Tato varianta poskytuje falešný statický vektor severu. Vektor je inicializován při prvním načtení, později zůstává stejný. Hodnota je nastavena na vektor směřující dopředu, otočený o hodnotu *InitialHeadingDegrees*, ve stupních, nastavenou pomocí komponenty *SettingsController*.

Varianta *FakeStaticNorthProvider* je vhodná pro krátkodobé testování bez nutnosti použití skutečného zdroje severu. Pro jednoduché otestování tato komponenta stačí, pouze je zapotřebí zajistit správné natočení uživatele při startu aplikace (viz sekci 2.9.2). Může být např. užitečné nastavit hodnotu *InitialHeadingDegrees* na hodnotu často zaujímaného natočení při testování, případně natočení při startu aplikace zkontrolovat pomocí kompasu a kontrolovat, zda jsou objekty umisťovány správně.

IotNorthProvider

Tato varianta poskytuje vektor z externího senzoru, připojeného do sítě WiFi. Data jsou z něj získávána pomocí protokolu HTTP.

2.4.4 Ostatní komponenty

Komponenty, které nebyly zmíněné, mají ze své podstaty pouze jednu variantu, a nepotřebují další komentář.

2.5 Princip převodu souřadnic

Tato sekce popisuje princip převodu geografických souřadnic na virtuální. Tento převod je součástí komponenty *GeolocationCalculationsService*. Úkolem je převést souřadnice bodu zadané zeměpisnou šířkou, zeměpisnou délkou a nadmořskou výškou na vektor, který znázorňuje posun bodu oproti kameře ve virtuálním světě v metrech. K tomu je nutné, aby kromě převáděných geografických souřadnic vstupovaly do algoritmu také geografické souřadnice kamery v reálném světě, aby byl možný převod geografických souřadnic (vztažených ke středu planety Země, viz 2.5.1) na souřadnice vztažené k pozici kamery. Také je zapotřebí mít na vstupu referenční vektor ve virtuálním světě, reprezentující směr severu v reálném světě, jelikož směr dopředu ve virtuálním světě nemusí nutně odpovídat severu.

Pro práci se souřadnicemi ve virtuálním světě byl zvolen levoruký systém souřadnic, tj. hodnoty souřadnice x rostou směrem doprava, hodnoty y rostou směrem nahoru a hodnoty z rostou směrem dopředu [13].

Jelikož výsledný vektor udává hodnoty v metrech, určit souřadnici y je triviální – odpovídá rozdílu převáděné nadmořské výšky a nadmořské výšky kamery. Dále je rozebrán převod souřadnic x a z.

Základní princip spočívá v provedení rozdílu převáděných geografických souřadnic a geografických souřadnic kamery po jednotlivých složkách, čímž vzniknou 2 rozdíly – rozdíl zeměpisných délek a rozdíl zeměpisných šířek, v obou případech ve stupních. Dále na ně bude referováno jako na *úhlové vzdálenosti*. Tyto vzdálenosti jsou poté převedeny na vzdálenosti v metrech, které by již reprezentovaly výsledné souřadnice, pokud by osa z směřovala na sever. Na závěr je tedy provedena korekce virtuálních souřadnic na základě vstupního vektoru severu.

Zásadní faktor ovlivňující kvalitu tohoto převodu je volba metody převodu vypočítaných stupňových rozdílů zeměpisných délek a zeměpisných šířek na vzdálenosti v metrech. Tato problematika je dále rozebrána v následujících podsekcích.

2.5.1 Geodetický systém WGS84

Světový geodetický systém 1984 (angl. *World Geodetic System 1984*, odtud WGS84) je světově uznávaný standard využívaný i v systému GPS [14]. Definuje geocentrický systém souřadnic a model tvaru planety Země, ve kterém je Země reprezentována jako elipsoid s konkrétními parametry. Konkrétní parametry elipsoidu v této práci nejsou uvedeny, vztahy pro převod v sekci 2.5.2 nejsou v této práci odvozeny, jsou převzaty. Je ale důležité využívat metody založené na zvoleném modelu WGS84, kvůli konzistenci se systémem GPS, který je využit pro získání pozic členů týmu [14].

2.5.2 Princip převodu úhlových vzdáleností

Úhlové vzdálenosti, tj. vypočítané rozdíly ve stupních, jsou převedeny po jednotlivých složkách – zeměpisná délka a zeměpisná šířka je převedena zvlášť, přičemž způsob převodu se pro každou složku liší.

Přesnost převodu stupňových rozdílů na vzdálenosti v metrech závisí na zvoleném modelu tvaru Země, zde je využit standard WGS84 (viz sekci 2.5.1, [14]), jelikož je využíván i v systému GPS.

Nejprve je vypočítána délka jednoho stupně (zeměpisné délky, nebo šířky) v metrech, posléze je touto délkou vynásoben stupňový rozdíl dané složky. Vztah 2.1 slouží k výpočtu délky v metrech jednoho stupně zeměpisné délky, pro danou zeměpisnou šířku ϕ:

$$111412.84 \cos\phi - 93.5 \cos 3\phi + 0.118 \cos 5\phi \tag{2.1}$$

Vztah 2.2 slouží k výpočtu délky v metrech jednoho stupně zeměpisné šířky, pro danou zeměpisnou šířku ϕ:

$$111132.92 - 559.82 \cos 2\phi + 1.175 \cos 4\phi - 0.0023 \cos 6\phi \tag{2.2}$$

Tyto vztahy byly převzaty z práce [15], jejich správnost byla ověřena využitím v práci [16].

Délka jednoho stupně není konstantní, protože vzhledem k tvaru Země se délka jednoho stupně zeměpisné délky i zeměpisné šířky liší pro každou zeměpisnou šířku [17]. Nabízí se tedy obava, že princip výpočtu délky jednoho stupně a následné roznásobení je nedostačující, a měl by být použit přesnější model. Nicméně, u zeměpisných délek se délka jednoho stupně pohybuje v desítkách kilometrů, stejně tak změny u délek jednoho stupně zeměpisné šířky, které jsou navíc zanedbatelné [17].

V této aplikaci se předpokládají vzdálenosti nižší, než je délka jednoho stupně (viz sekci 2.1.2, [17]). Jedná se tedy o dostačující aproximaci pro tuto aplikaci. Není třeba použít přesnější model, ani délku jednoho stupně přepočítávat v rámci jednoho běhu aplikace – nepředpokládá se přesun uživatele o délku větší než jednoho stupně při jednom spuštění, s ohledem na výdrž baterie (viz sekci 2.6.1) a nefunkční požadavky (viz sekci 2.1).

2.6 Použitý hardware

2.6.1 Chytré brýle NReal

Pro implementaci prototypu jsou v této práci použity chytré brýle NReal Light. Původně byla snaha nalézt brýle, které by splňovaly co nejvíce nefunkčních požadavků již v rámci brýlí či jejich ovládací jednotky.

Nakonec ale sehrál největší roli parametr výdrže baterie a cena brýlí. Jelikož byla původně snaha nalézt brýle co by již v sobě či ovládací jednotce měly GPS, kompas k určení orientace hlavy vůči severu, apod., nebyl při první rešerši kladen na brýle NReal Light velký důraz, ale při konzultaci s Ing. Davidem Sedláčkem, Ph.D., na Katedře počítačové grafiky a interakce, byly právě brýle NReal Light doporučeny z hlediska osobní zkušenosti s parametry brýlí a vývojařské přívětivosti těchto brýlí.

NReal Light DevKit a výpočetní jednotka

V této práci byla konkrétně využita varianta NReal Light DevKit. Tato varianta brýlí NReal Light je dodávána ve vývojářské sadě, jejíž hlavní součástí je NReal Light výpočetní jednotka (v dokumentaci *Computing unit* [18]). Na výpočetní jednotku jsou nahrávány Android aplikace vytvořené pro brýle NReal. Brýle se k jednotce připojí pomocí konektoru USB-C.

Kromě toho, že jednotka slouží ke spouštění aplikací, je schopna sloužit jako ovladač aplikací pro chytré brýle NReal [18]. Otáčením jednotky je možné ovládat Na horní kruhové části se nachází 3 tlačítka:

- APP tlačítko (horní část kruhu)
- TRIGGER tlačítko (střed)
- HOME tlačítko

APP a TRIGGER tlačítka může bez problémů definovat každá aplikace pro své účely, HOME tlačítko slouží především k opuštění aplikace (krátkým stisknutím), nebo kalibraci ukazovátka (dlouhým stisknutím). Zároveň tato kruhová část slouží jako trackpad – pohybem prstů po něm je možné např. ovládat kurzor, nebo rolovat obsah komponent uživatelského rozhraní [18].

Z jednotky lze dokonce horní část oddělit. Má vlastní akumulátor a při dostatečném nabití slouží jako ovladač odděleně [18]. Tato jednotka byla dle zadání využita pro vývoj.

2.6.2 Modul magnetometru HMC5883L

V průběhu tvorby bylo nejprve počítáno s jednoduchou webovou aplikací, která by byla načtena na mobilním telefonu a posílala data o lokaci, hlášeném stavu a natočení hlavy na webový server. Tato jednoduchá aplikace byla vytvořena. Webový server byl vytvořen v php, frontend pomocí javascriptu získaval data o natočení a poloze telefonu. Nakonec ale tato testovací aplikace, později nazvaná *MpsgWeb* byla využita pouze jako *NorthProvider* (viz. sekci 3.4).

Pro snazší integraci (oddělený modul připevněný na hlavě, viz sekci 3.5) a vyšší přesnost byl ale nakonec zvolen modul magnetometru HMC5883L. Jedná se o tříosý magnetometr, s nominální přesností 1-2 stupňů [19]. Dle zkušenosti autora byla implementace snazší než pomocí webové aplikace, jelikož nebyla nalezena spolehlivá implementace kompasu v klientském javascriptu.

2.6.3 Bezdrátový modul Wemos D1 Mini (ESP8266)

Pro přenos dat z magnetometru je v aplikaci využita síť WiFi. K připojení k síti je využit modul Wemos D1 Mini, postavený na základu WiFi modulu ESP8266. Tento modul v této práci slouží pro implementaci modulu kompasu, neboli *IotNorthProvideru* (viz sekci 3.5).

2.7 Použité softwarové nástroje

2.7.1 Unity

Unity je herní engine, umožňující tvorbu her na PC, konzole, mobilní telefony a web [20]. Tvorba aplikací v Unity sestává z práce ve dvou prostředích – *Unity Editor* a vývojové prostředí *Microsoft Visual Studio* (výchozí), nebo jiné vývojové prostředí [13].

V *Unity Editoru* je tvořena herní scéna, pomocí jednoduchých nástrojů, podobných pokročilejším prostředím pro tvorbu 3D modelů. Mezi tyto nástroje patří např. tvorba základních 3D objektů, translace, rotace, nastavení textur, materiálů, a jiné [20].

Ve vývojovém prostředí je pak možné vytvářet a upravovat skripty, které poté ovlivňují logiku aplikace a chování objektů. Skripty je možné k objektům přiřadit opět v *Unity Editoru*.

2.7.2 Jazyk C# v Unity

Jazyk C# je v Unity možné použít pro tvorbu logiky aplikace, např. změnou chování jednotlivých objektů na scéně. C# skript se vytvoří pravým kliknutím v panelu *Project*, *Create - C# Script*. Dvojklikem na skript je pak otevřeno vývojové prostředí.

Základně je vytvořen soubor se třídou, která dědí od `MonoBehaviour`. Ta obsahuje řadu metod, které může vývojář předefinovat a ovlivnit tak chování aplikace. Základními stavebními kameny jsou metody `Start`, automaticky volaná v daném skriptu při spuštění aplikace a `Update`, volaná při každém snímku.

Vytvořené třídě také může vývojář přidat proměnné. Důležité je, že pokud je proměnná deklarovaná jako `public`, je možné měnit její hodnotu i v prostředí *Unity Editoru*, v tzv. *Inspector window*. Tímto způsobem je také možné provázat několik tříd (tj. i skriptů) mezi sebou, tedy pomocí veřejné proměnné typu dané třídy, a provázáním konkrétních objektů v prostředí *Unity Editoru*. Přiřazením skriptu k danému objektu v herní scéně (např. k nějakému 3D objektu) pomocí *Drag&Drop* nebo *Add Component* je aktivována funkcionalita tohoto skriptu a je přiřazena k danému objektu. Pak je možné tento objekt přiřadit jako hodnotu veřejné proměnné u jiného objektu, a tímto způsobem dojde ke zmiňovanému propojení více skriptů.

2.7.3 NRSDK

NRSDK je oficiální vývojová sada pro NReal brýle postavená na herním enginu Unity [12]. Přidává se jako balíček – *asset* do jednotlivých projektů v Unity. To se může zdát jako nevýhoda, jelikož není na jednom místě, ale v každém projektu, takže zabírá více místa. Tento balíček má ale pouze přibližně 100 MB, takže se nejedná o zásadní nevýhodu. Navíc tento přístup může zmírnit problémy s kompatibilitou při přechodu na novou verzi NRSDK u nového projektu.

Princip programování brýlí NReal

Chytré brýle jsou v podstatě pouze displej, je k nim zapotřebí používat jednotku, na které běží aplikace. Na tuto jednotku bude v textu referováno jako na *výpočetní jednotku*. Výpočetní jednotkou může být například vhodný mobilní telefon, nebo je tento ovladač speciální zařízení dodávané k daným chytrým brýlím.

V případě NReal brýlí a NRSDK je umožněno aplikaci nasadit na vhodný mobilní telefon s Androidem, nebo využít výpočetní jednotku dodávanou k vývojářskému kitu brýlí NReal Light (využito v této práci, dle zadání) [12] [18].

Možnosti vývoje aplikací pro brýle NReal

V době tvorby práce lze aplikace pro NReal brýle vyvíjet pouze na platformě Unity [12]. V budoucnosti má výrobce v plánu zpřístupnit integraci NRSDK

i pro Unreal Engine [12]. Také je plánováno umožnit vývojářům tvorbu aplikací pro NReal brýle mimo stávající platformu NRSDK, podporou vývoje nativních Android aplikací [12]. Podrobnější schéma ekosystému NReal je na obrázku 2.6.

Edice NRSDK

Každá verze NRSDK vychází ve 3 edicích [18]:

- Normal – běžná verze, otestovaná, s plnou podporou
- Experimental – zpřístupňuje experimentální funkce, které mají být zveřejněny v budoucích verzích varianty Normal
- Enterprise – Nabízí přístup k surovým datům (černobílá kamera, data ze senzorů IMU, včetně magnetometru)

V této práci je využita edice Normal. Edice Enterprise byla brána v úvahu, jelikož nabízí surová data z magnetometru [18], která by mohla sloužit při implementaci komponenty *NorthProvider*. Zde ale záleží na domluvě s výrobcem, nelze tuto verzi pořídit přímo na stránkách. Proběhl pokus o kontaktování a získání Enterprise verze pro tuto práci za cenu dostupnou pro studenta či zdarma, ale po prvotní odpovědi, že žádost bude odeslána k vyřízení, nebyla obdržena žádná další odpověď. Bylo tedy pokračováno ve vývoji v edici Normal.

Princip práce s NRSDK

NRSDK poskytuje do Unity objekty jako například *NRCameraRig*, který zapouzdřuje práci s kamerou - pohledem první osoby. Pozice tohoto objektu ve scéně znázorňuje aktuální pozici a orientaci brýlí vůči virtuálnímu světu, takže vůči tomuto objektu může vývojář umístit viditelné objekty tam, kde se mají nacházet ve virtuálním světě. Pro vytvoření prosté aplikace pro NReal brýle je dostačující tento *NRCameraRig* přidat do scény, nastavit parametry pro build a výsledkem je aplikace spustitelná na ovládací jednotce NReal.

Dále NRSDK umožňuje vývojáři pracovat s uživatelským vstupem (především ukazovátkem – *Raycaster*) a reagovat na interakce s prvky uživatelského rozhraní Unity – např. s tlačítky. K práci s uživatelským vstupem slouží NRSDK Asset (objekt) nazvaný *NRInput*.

Pokud vývojář vytváří vlastní skripty, může nastat potřeba pracovat programově se zmíněnými objekty *NRCameraRig* nebo *NRInput*, např. z důvodu získání pozice kamery ve virtuálním světě, nebo kontroly stavu vstupního zařízení (je-li stisknuté tlačítko apod.). V NRSDK je napojení na tyto objekty řešeno pomocí statických tříd, takže není potřeba instanciovat služby – je to vidět např. v NRSDK zdrojovém kódu třídy *NRSessionManager*, která realizuje návrhový vzor Singleton.

NRSDK a simulace v Unity

NRSDK obsahuje mechanismus, kterým je při spuštění hry v Unity Editoru rozpoznáno, že se jedná o Unity Editor a ne reálné brýle. Díky tomu pak vytvoří herní objekty, které simulují v Unity reálné chování chytrých brýlí. V tomto simulátoru byly testovány malé změny, které nebylo nutné testovat na fyzických brýlích.

V simulátoru je pomocí kláves W, A, S a D možné simulovat chůzi - pohyb brýlí. Pomocí klávesy shift a pohybu myši je možné pohybovat s ukazovátkem. V pravé části herního okna se nachází objekt, který simuluje vstupní zařízení herních brýlí – má trackpad a tlačítka.

2.7.4 Microsoft Visual Studio

Pro programování skriptů pro Unity je v této práci využito Microsoft Visual Studio, které je výchozím IDE pro Unity. Velká výhoda je v tom, že Microsoft Visual Studio nabízí instalaci modulu pro podporu Unity. Díky této podpoře je možné např. spustit hru v Unity a využívat ladící nástroje ve Visual Studiu, tedy např. si program odkrokovat a sledovat, proč se objekty vykreslují špatně, na základě kterých hodnot vznikla chyba, apod.

2.7.5 Android Debug Bridge

Jelikož jsou pro zvolené brýle NReal aplikace vyvíjeny v současné době pouze na platformě Android, výsledná aplikace je balíček `.apk`. Je tedy potřeba mít způsob, jakým tento balíček `.apk` nainstalovat do NReal výpočetního zařízení. V tomto projektu byl využíván *Android Debug Bridge*. *Android Debug Bridge* je program, který se automaticky spáruje s Android zařízením připojeným přes USB, a umožňuje pak do zařízení instalovat `apk` balíčky [21].

Je možné využít i instalaci přes síť *WiFi* [21]. Tento způsob byl využit, spolehlivě odstranil nutnost odpojovat a připojovat NReal výpočetní jednotku při každé instalaci nové verze aplikace, a ušetřil tak velké množství času. Výpočetní jednotka nemůže být zároveň připojena k brýlím a k počítači. Brýle NReal jsou totiž k jednotce připojené pomocí konektoru typu USB-C, který ale musí být použit i pro programování pomocí ADB.

Unity má vestavěnou podporu pro ADB, která byla v této práci využita. V *Unity Editoru* lze v *Build Settings* nastavit na které zařízení připojené přes se má sestavená aplikace nasadit. Poté jej stačí jen sestavit a nasadit pomocí *Build & Run* či příslušné klávesové zkratky. Jeden cyklus sestavení a nasazení trval při tvorbě této práce přibližně 2 minuty, proto byly malé změny testovány pomocí simulátoru přímo v Unity Editoru.

2.7.6 Arduino

Arduino je značka jednočipových počítačů, založených převážně na mikrokontrolérech AVR výrobce Atmel [22]. Projekt Arduino začal v roce 2005

v Itálii, za účelem vyrobit intuitivní výukovou pomůcku k výuce elektrotechniky a programování [23]. Brzy však Arduino přesáhlo svůj původní účel, je využíváno např. v mnohých volnočasových projektech po celém světě [23] [22].

V tomto projektu je využito Arduino pro jednoduchou práci s ESP8266. Modul ESP8266 je v projektu pouze podpůrný, nejedná se o stěžejní část práce na aplikaci, nebylo tedy zapotřebí nalézt nejefektivnější platformu, ale nějakou, která umožňuje snadnou instalaci a práci. To Arduino splňuje.

2.8 Synchronizace aktuální pozice a natočení hlavy

Tato sekce popisuje, jakým způsobem je zajištěno správné pozicování objektů vzhledem k aktuálnímu natočení hlavy uživatele chytrých brýlí a jeho pozice. Toto téma je částečně zmíněno v sekci o přepočtu GPS souřadnic na relativní virtuální souřadnice vzhledem ke kameře, v sekci 2.5.

Při použití chytrých brýlí NReal a prostředí Unity s NRSDK bylo zjištěno, že při zapnutí aplikace se kamera nachází na virtuálních souřadnicích $(0, 0, 0)$ a pohledem směřuje dopředu, tj. ve směru vektoru $(0, 0, 1)$. Poté je již virtuální svět zafixován a kamera se v něm pohybuje na základě reálného pohybu NReal brýlí, zjištěného interně pomocí senzorů NReal brýlí – vývojář k těmto datům mimo *Enterprise edici NRSDK* nemá přístup (viz sekci 2.7.3).

Brýle NReal Light, použité v této práci, snímají svůj pohyb se 6 stupni volnosti (označeno 6DoF v oficiální dokumentaci). To znamená, že kromě tříosého natočení brýlí je navíc snímán pohyb dopředu/dozadu, doleva/doprava a nahoru/dolů [24].

Pokud by toto 6DoF snímaní bylo dokonalé, stačilo by tedy při spuštění aplikace zjistit natočení hlavy vůči severu a GPS lokaci brýlí pouze jednou. O zbytek by se pak postaraly brýle samotné pomocí 6DoF snímání, a stačilo by již měnit pozice členů týmu. Tento extrémní přístup nebyl využit, pro vyšší přesnost a kvůli zkušenostem byl zvolen přístup častějších synchronizací lokace i natočení hlavy, ale výhodou je, že díky 6DoF snímání není nutné počítat vše manuálně a GPS lokaci společně s natočením vzhledem měřit s každým vykreslovaným snímkem.

2.9 Varianty nasazení

Varianty nasazení testované v této práci se nacházejí v tabulce 2.7. V sekcích jednotlivých variant je uvedeno, k jakému účelu varianta slouží. Také je zde výpis variant komponent, které jsou v dané variantě obměněny. Vypisovány jsou pouze varianty komponent:

- MemberInfoProvider
- ActiveUserInfoProvider
- NorthProvider

Pokročilejší popis dané varianty se nachází na diagramech nasazení v každé sekci. U diagramů nasazení nejsou z důvodu šetření místem zobrazovány všechny softwarové komponenty.

Varianty jsou uvedeny ve stejném pořadí, v jakém byly postupně vyvíjeny, na diagramech nasazení jednotlivých variant lze tedy sledovat, jak se systém měnil v průběhu vývoje.

2.9.1 Varianta A

Varianta A slouží k základnímu otestování funkčnosti pozicování objektů a výpisu dat na základě falešných statických souřadnic a dalších falešných údajů členů týmu. Skládá se z variant komponent:

- FakeStaticMemberInfoProvider
- FakeStaticActiveUserInfoProvider
- FakeStaticNorthProvider

K fyzickému otestování stačí pouze NReal výpočetní jednotka a brýle, jedná se tedy o vhodnou variantu k testování v domácích podmínkách. Diagram nasazení varianty A je na obrázku 2.7.

2.9.2 Varianta B

Varianta B slouží k pokročilejšímu otestování funkčnosti pozicování objektů, včetně periodického obnovení pozice – nově se zde objevují pohyblivé souřadnice, ačkoliv falešné. Geografická pozice aktivního uživatele je již určena dynamicky dle reálných souřadnic ze systému *MpsgWeb*. Skládá se z variant komponent:

- FakeDynamicMemberInfoProvider
- MpsgWebActiveUserInfoProvider
- FakeStaticNorthProvider

K fyzickému otestování stačí stejně jako u varianty A pouze NReal výpočetní jednotka a brýle, jedná se tedy o vhodnou variantu k testování v domácích podmínkách. Diagram nasazení varianty B je na obrázku 2.8.

2.9.3 Varianta C

Varianta C již slouží k pokročilým testům celého systému, s napojením na externí systém testovací aplikace MpsgWeb. Skládá se z variant komponent:

- MpsgWebMemberInfoProvider
- MpsgWebActiveUserInfoProvider
- IotNorthProvider

K fyzickému otestování je již kromě brýlí a výpočetní jednotky zapotřebí IOT modul pro určování severu a mobilní telefon pro odesílání dat o pozici (*ActiveUserInfoProvider*).

Tato varianta je vhodná pro testy v exteriéru a měření přesnosti při použití reálných měnících se souřadnic a údajů o členech týmu. Testování je také možné provádět s více zúčastněnými členy s mobilními telefony a simulovat tak reálný případ užití. Pro měření přesnosti je možné využít více mobilních telefonů (či jiných GPS zařízení s připojením na Internet) bez nutnosti dalších zúčastněných skutečných osob. Diagram nasazení varianty C je na obrázku 2.9.

2.9.4 Varianta D

Varianta D slouží k testům integrace modulu chytrých brýlí a modulu kompasu se systémem DTA. Skládá se z variant komponent:

- DtaMemberInfoProvider
- DtaActiveUserInfoProvider
- IotNorthProvider

K fyzickému otestování je kromě brýlí a výpočetní jednotky zapotřebí také IOT modul pro určování severu.

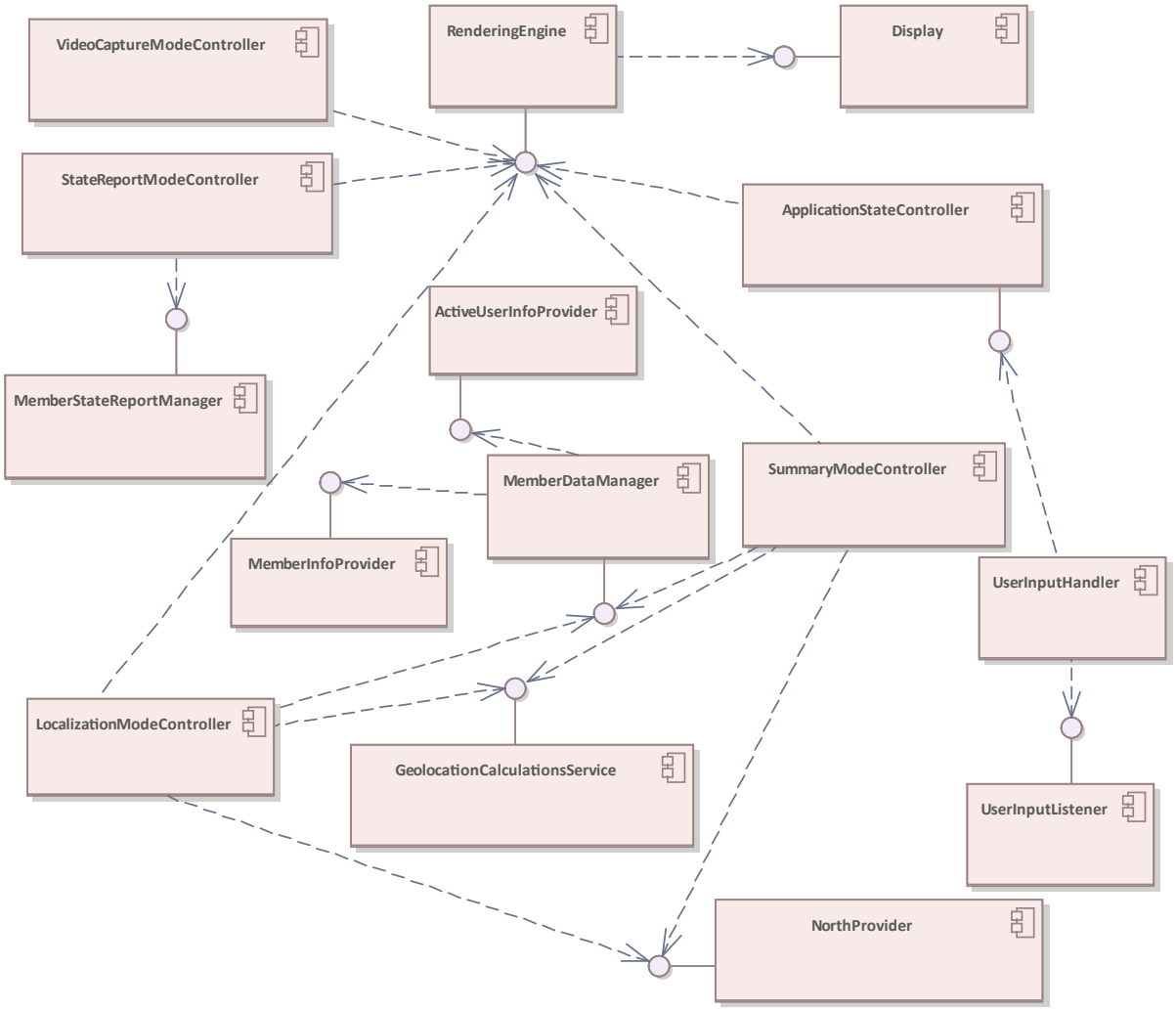
Testy integrace je možné provádět i v domácích podmínkách, mohou být přehrávány reálné mise ze systému DTA, přičemž musí být určeno, které informace z množiny přijímaných informací přísluší aktuálnímu uživateli chytrých brýlí. Diagram nasazení varianty D je na obrázku 2.10. Tato varianta není v této práci implementována.

Název	Popis	Implementace
MpsgComponentsConfigurator	Inicializace správných variant ostatních komponent	Tato práce
MemberInfoProvider	Získání pozic a dalších informací o členech týmu	Tato práce
ActiveUserInfoProvider	Získání pozice a dalších informací o uživateli chytrých brýlí	Tato práce
MemberDataManager	Periodické získávání informací o členech a jejich správa	Tato práce
NorthProvider	Získání vektoru severu vzhledem k aktuálnímu světu renderovanému v chytrých brýlích	Tato práce
GeolocationCalculationsService	Převod GPS souřadnic na souřadnice ve světě renderovaném v 3D brýlích	Tato práce
ApplicationStateController	Ovládání stavu aplikace	Tato práce
SettingsController	Ovládání nastavení aplikace	Tato práce
MemberStateReportManager	Zajišťuje nahlášení stavu aktivního uživatele (StateReportMode) do externího systému	Tato práce
UserInputHandler	Reakce na uživatelské vstupy	Tato práce
LocalizationModeController	Řízení činnosti režimu LocalizationMode	Tato práce
SummaryModeController	Řízení činnosti režimu SummaryMode	Tato práce
StateReportModeController	Řízení činnosti režimu StateReportMode	Tato práce
VideoCaptureModeController	Řízení činnosti režimu VideoCaptureMode	Tato práce

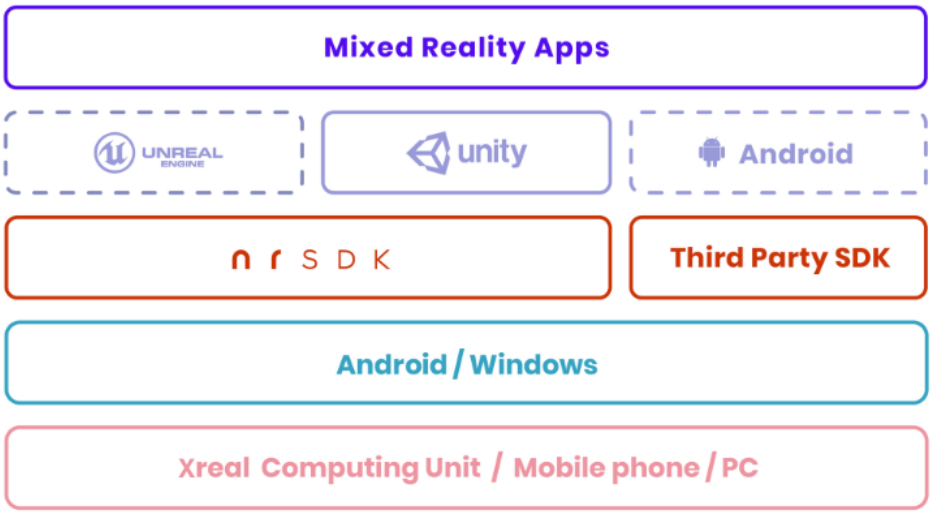
Tabulka 2.5: Přehled softwarových komponent implementovaných v této práci

Název	Popis	Implementace
RenderingEngine	Prostorové manipulace s objekty, přepínání scén	Externí
Display	Zobrazení obrazu v chytrých brýlích	Externí
UserInputListener	Zachytávání uživatelských vstupů	Externí

Tabulka 2.6: Přehled externích softwarových komponent



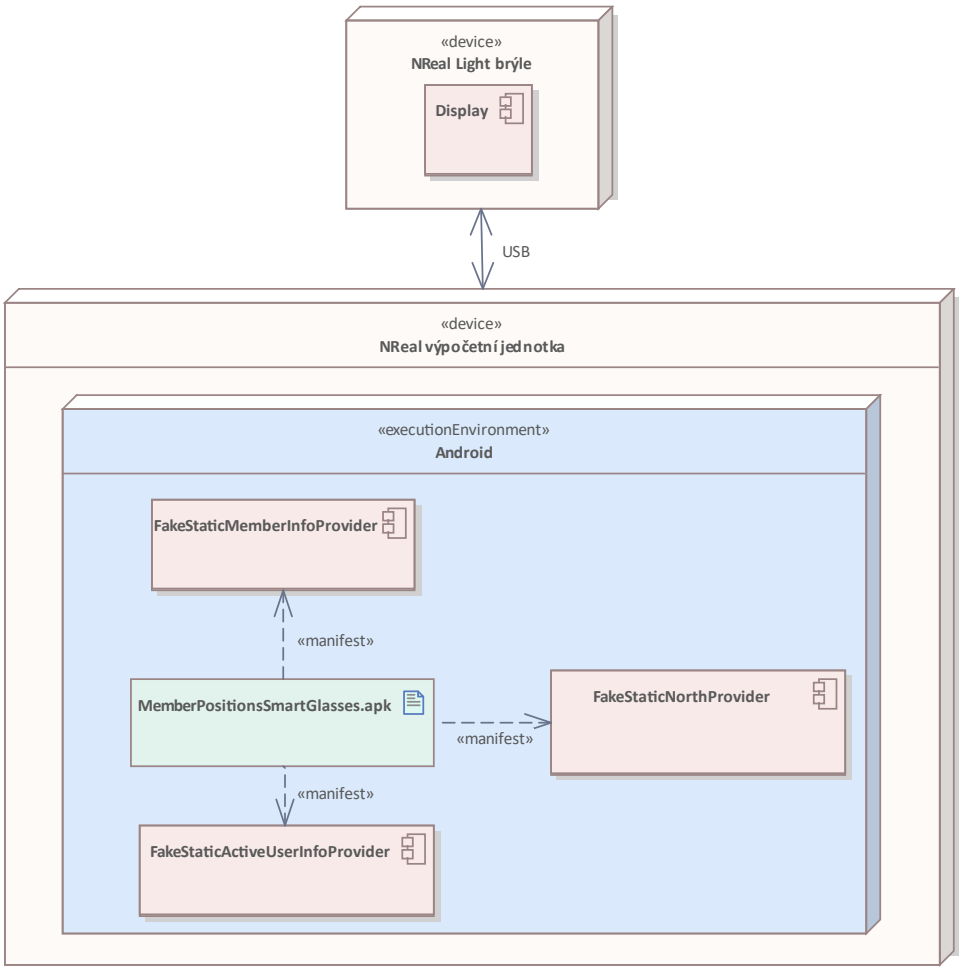
Obrázek 2.5: Diagram komponent



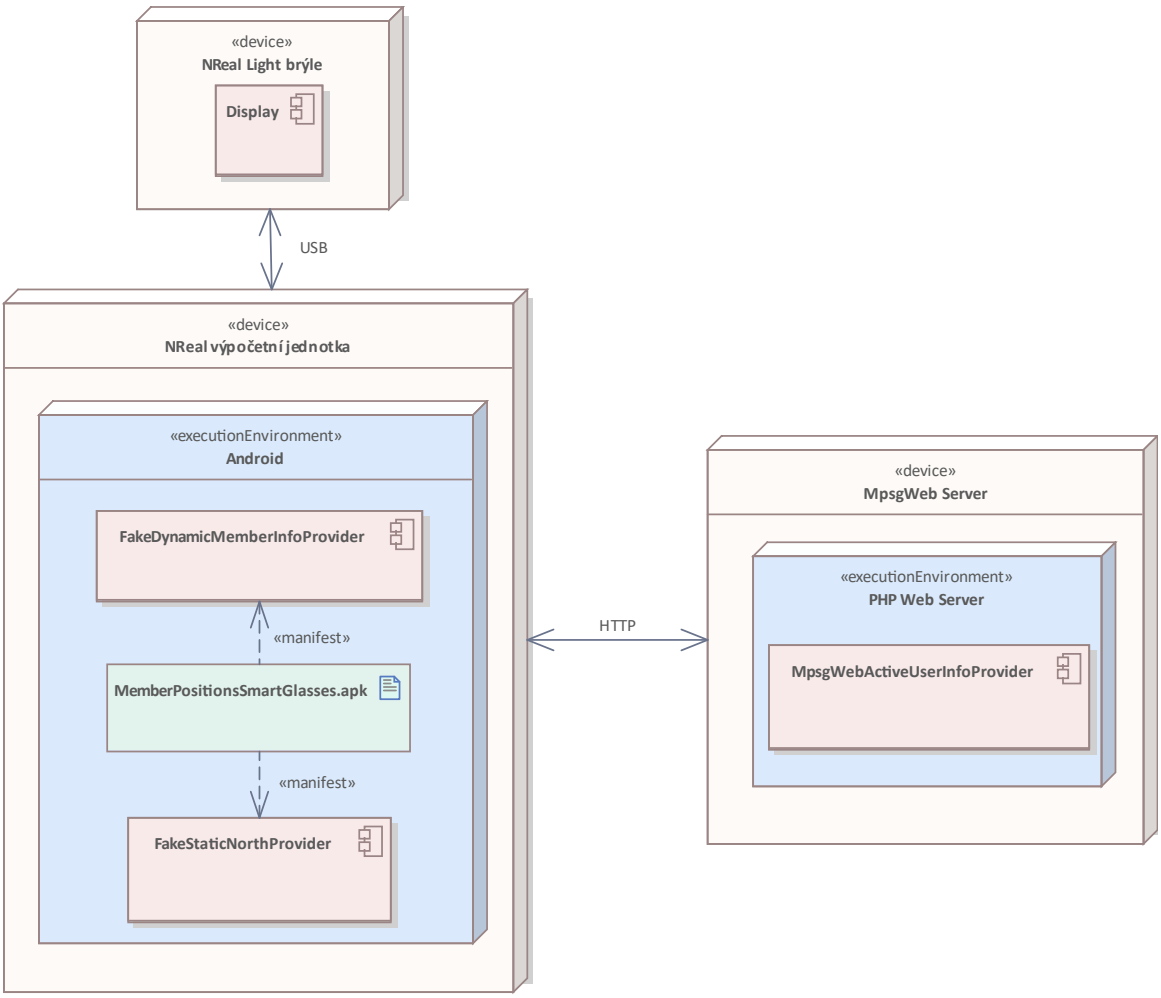
Obrázek 2.6: NReal ekosystém

Název	**Rozsah činnosti**
Varianta A	Základní zobrazování statických pozic
Varianta B	Zobrazování pohyblivých pozic
Varianta C	Zobrazování pozic a dalších dat z testovací webové aplikace (HTTP)
Varianta D	Zobrazování pozic a dalších dat ze systému DTA (WebSocket)

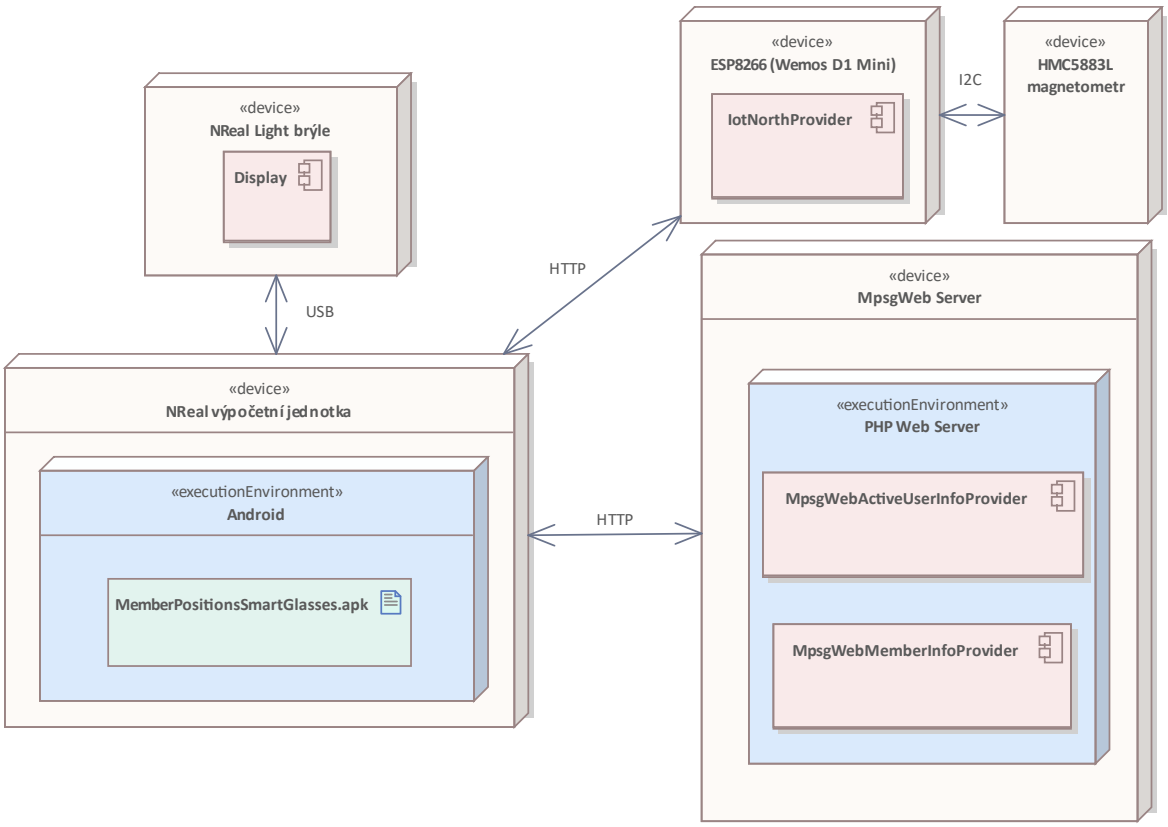
Tabulka 2.7: Přehled variant nasazení



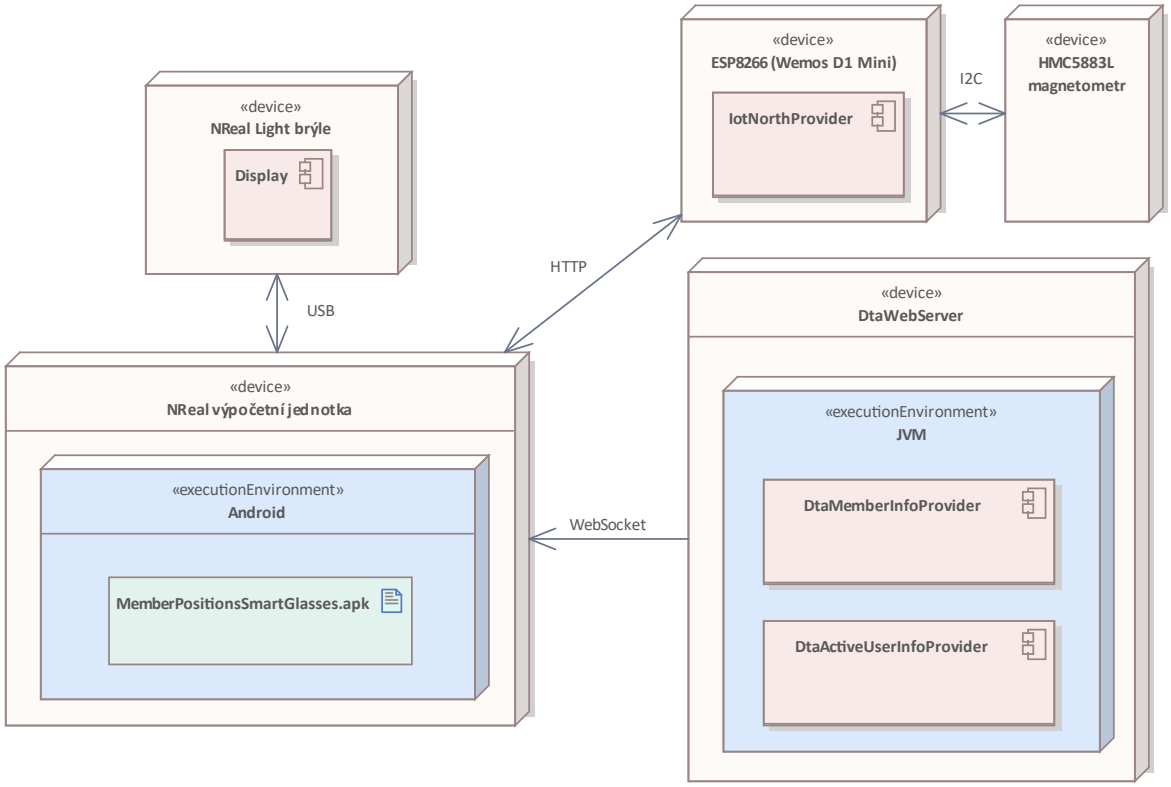
Obrázek 2.7: Diagram nasazení varianty A



Obrázek 2.8: Diagram nasazení varianty B



Obrázek 2.9: Diagram nasazení varianty C



Obrázek 2.10: Diagram nasazení varianty D

Kapitola 3

Popis realizace

3.1 Struktura projektu

Součástí projektu jsou kromě projektu v Unity i testovací JSON tratě s lokacemi členů, Python skript pro převod formátu GPX na JSON trať využitelnou při testování, užitečné skripty, zdrojový kód textu této diplomové práce v LaTeX aj. Tyto části jsou členěny do podadresářů kořenového adresáře, jejich popis se nachází v tabulce 3.1.

Soubor	Popis
analysis	Diagramy – projekt v Enterprise Architect
src	Vlastní implementace a utility (více v tabulce 3.2)
resources	Jinam nezařazené zdroje
thesis	Text diplomové práce, se zdrojovými kódy a grafikou

Tabulka 3.1: Popis kořenového adresáře

Dále bude popsán adresář `src`, jako stěžejní adresář této práce. Obsahuje především adresář se zdrojovými kódy. Zajímavé utility jsou v adresáři `utils`. Celý popis je v tabulce 3.2.

Soubor	Popis
arduino	Implementace modulu kompasu a jeho kalibrace
php	Implementace serveru MpsgWeb
unity	Projekt v Unity, popsaný v sekci 3.3
utils	Utility

Tabulka 3.2: Popis adresáře src

3.2 Vývojové prostředí

Aplikace byla vyvíjena pomocí nástrojů více popsaných v sekci 2.7. Verze použitých nástrojů jsou uvedeny v tabulce 3.3. Verze nástrojů dle dokumentace výrobců není nutné dodržet přesně, např. dokumentace NRSDK deklaruje, že jakákoliv budoucí verze Unity bude kompatibilní, v dokumentaci Unity je psáno, že je zpětně kompatibilní, zároveň ale tato dokumentace varuje před migrací projektů na vyšší verze a doporučuje projekt zálohovat (jedná se o nevratný proces). Proto je doporučeno v případě problémů nahlédnout do tabulky 3.3 a případně nainstalovat uvedené verze nástrojů.

Nástroj	**Použitá verze**	**Omezení**
NRSDK	1.10.2	Použita nejvyšší verze, která podporuje NReal Light DevKit použitý v této práci
Unity	2022.3.26f1	Pro NRSDK 1.10.2 stačí verze 2018.4.x a vyšší
MS Visual Studio Community 2022	17.9.6	Unity Editor lze propojit s jakýmkoliv vývojovým prostředím, pro jistotu použita verze instalovaná společně s Unity Editorem
Android Debug Bridge	35.0.1-11580240	–

Tabulka 3.3: Použité verze SW nástrojů

Při nastavení vývojového prostředí byl při tvorbě aplikace největší problém s instalací správné verze Android SDK. Android SDK je instalováno společně s Unity, aby mohlo sestavovat výslednou aplikaci, ale bohužel je instalována pouze jedna konkrétní verze SDK ke konkrétní verzi Unity. Výpočetní jednotka NReal Light ale potřebuje nižší verzi, než byla nainstalována společně s Unity. Je možné změnit v nastavení cestu k Android SDK, ale pokud tak vývojář učiní, je okamžitě upozorněn, že to není doporučeno. Tímto způsobem bylo příslušné SDK nainstalováno, byla změněna cesta v nastavení, ale nebylo dosaženo úspěchu. Překopírování souborů do adresáře `platforms` v místě, kde je instalováno Android SDK nainstalované pomocí Unity, taktéž nepomohlo. NRSDK dokumentace navíc vybízí, aby bylo příslušné Android SDK nainstalováno pomocí SDK Manageru, který je součástí Android Studio, takže manuální instalace pomocí kopírování není v souladu s touto dokumentací.

Nakonec se ukázalo, že problém při původním pokusu o změnu cesty byl v přístupových právech k souborům SDK – Android Studio SDK Manager totiž SDK ve výchozím nainstaloval do `C:\Program\Files\Android`, kam Unity nemělo právo zapisovat. Instalace Android SDK do adresáře `C:\Android` (lze zvolit vlastní cestu při startu SDK Manageru) problém vyřešilo. Unity nabízí ještě další možnost, změnu cesty Android NDK, ale to bylo bez problému ponecháno na výchozí cestě.

Pro import NRSDK byly následovány pokyny v oficiální dokumentaci NRSDK 1.10.2. NRSDK je staženo jakožto Unity balíček, importováno do Unity. Poté je zapotřebí v Unity nakonfigurovat *Build Settings,* zejména přepnout platformu na Android. Velkou část dalšího nastavení pak zvládne průvodce *Project Tips,* který je součást NRSDK. Dostupný je přímo přes menu v Unity, vytvořeném po importu NRSDK balíčku (*NRSDK* | *Project Tips* | *Accept all*). Pokud nějaká nastavení nedokázal průvodce nastavit, musí být nastaveny manuálně. Toto nastavení se nachází v dokumentaci NRSDK citeNrealc2019.

Jako poslední zmíněné je pro správně nastavené vývojové prostředí zapotřebí instalovat podporu pro Unity na straně *Microsoft Visual Studio,* pokud tak nebylo učiněno automaticky při instalaci Unity, např. pokud před instalací Unity bylo již *Microsoft Visual Studio* dostupné a neinstalovalo se společně s Unity (stejně jako při tvorbě této práce). To je možné vykonat pomocí *Microsoft Visual Studio installeru.*

3.3 Projekt v Unity

Unity projekt se nachází v adresáři `src/unity/MemberPositionsSmartGlasses`. V projektu se nachází několik otevřených Unity scén, reprezentujících jednotlivé režimy aplikace (viz. sekci 2.2). Projekt obsahuje 2 typy herních objektů:

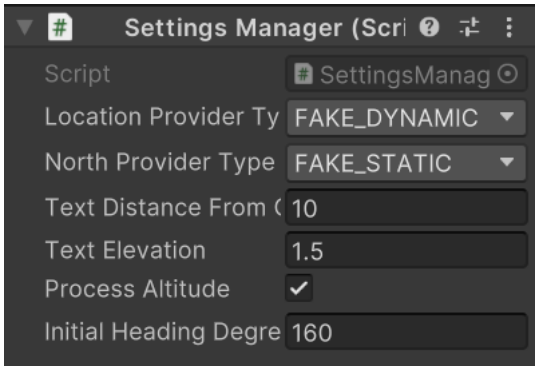
1. Společné objekty pro všechny režimy
2. Objekty specifické pro jeden konkrétní režim

V následující sekci jsou popsány jednotlivé scény. U scén jsou zdůrazněny důležité implementační detaily a do detailu rozebrány některé objekty, které se ve scénách nacházejí. Jsou popsány i některé důležité skripty navázané na tyto objekty. Častým případem jsou *prázdné* herní objekty, které slouží pouze k navázání stejnojmenného skriptu, který něco konkrétního při běhu aplikace vykonává. S ohledem na tuto skutečnost budou termíny *objekt* a *skript* budou v těchto případech používány zaměnitelně.

3.3.1 Scéna StartupScene - společné objekty

Společné objekty jsou všechny neviditelné, nejedná se o 3D objekty, pouze o prázdné herní objekty, které na sebe mají navázány skripty. Tyto objekty slouží k uchování stavu napříč různými scénami. Jedná se zejména o následující objekty:

- SettingsManager
- ApplicationStateController
- MpsgComponentsConfigurator
- UserInputHandler



Obrázek 3.1: Nastavení hodnot SettingsManageru v Unity

- MemberDataManager

Pro společné objekty je v jejich skriptu v metodě *Awake* zavolána funkce *DontDestroyOnLoad*, poskytnutá UnityEnginem. Tímto způsobem je v Unity zajištěno, že tyto objekty nebudou zničeny po přepnutí scény, zůstanou platné i v jiných režimech. Zároveň jsou některé z těchto objektů řešeny pomocí návrhového vzoru singleton – jedná se totiž o objekty, které spravují celou aplikaci, je tedy zapotřebí právě jedna instance z nich. Při pokusu o instancializaci dalšího herního objektu daného typu je v jeho metodě *Awake* zkontrolováno, zda již byl nějaký objekt tohoto typu vytvořen, a v tomto případě je zavolána metoda *Destroy* a funkce *Awake* je předčasně opuštěna. Tento princip se nachází např. ve skriptu *ApplicationStateController.*

Tato scéna je v Unity Editoru označena jako aktivní a při startu aplikace se spustí jako první. Tato scéna ale nemá žádné viditelné objekty, takže uživatel si jejího načtení ani nevšimne. Po inicializaci všech jejích objektů je v metodě *Start* objektu *ApplicationStateController* načtena první viditelná scéna, určena atributem *InitialMode* tohoto *ApplicationStateControlleru.* Ten je možné měnit i v rozhraní *Inspector window* v *Unity Editoru.* Tímto způsobem byly měněny testované scény při vývoji konkrétních scén. Zároveň je tímto způsobem možné určit, která scéna se bude spouštět jako první.

SettingsManager

Objekt SettingsManager sdružuje některá důležitá nastavení, aby byla dostupná na jednom místě. Tato nastavení je možné před sestavením aplikace změnit v *Inspector window* v pravé části *Unity Editoru.* Nastavení jsou na obrázku 3.1. Objekt je řešený návrhovým vzorem Singleton.

ApplicationStateController

Objekt ApplicationStateController řeší ve stejnojmenném skriptu přepínání jednotlivých režimů – pomocí přepínání Unity scén. K tomu využívá *SceneManager* a *SceneUtilities*, které jsou součástí Unity [13].

Scény musí být v projektu otevřené a všechny ostatní, kromě scény načítané jako první, musí být v Unity Editoru deaktivovány pomocí pravého kliknutí

na scénu a volby *Unload Scene.* První načítaná Unity scéna v tomto projektu musí být vždy *StartupScene,* protože obsahuje společné objekty, které by jinak nebyly inicializovány.

První *užitečná* načtená scéna, tj. první načtený režim aplikace je pak zvolen v *Inspector window* u tohoto skriptu, v atributu *InitialMode.*

Při načítání je na scény v kódu referováno pomocí jejich názvu, není tedy vhodné názvy měnit. Přiřazení jednotlivých názvů k módům je určeno ve skriptu *ApplicationStateController* v položce *SceneNameMappings.*

MpsgComponentsConfigurator

Skript *MpsgComponentsConfigurator* sdružuje společné objekty, a instancializuje služby, které nejsou řešeny pomocí Unity objektů. podle nastavení v objektu *SettingsManager.* Vytváří pak objekt typu *MpsgComponentsConfig,* který uchovává ve veřejné statické proměnné *Config.* V této proměnné jsou uloženy jednotlivé instance všech sdružovaných objektů. Zde získávají své závislosti ostatní skripty, které tyto instance potřebují.

Důležité je, že instancializace je prováděna v metodě *Awake* – tato metoda je totiž u všech skriptů vykonána ještě před metodou *Start,* kde ostatní skripty instancializované objekty získávají. Tímto způsobem je tedy zajištěno, že instancializace bude provedena včas. Nestane se, že by jiný objekt v metodě *Start* získal null místo skutečného objektu.

UserInputHandler

Skript *UserInputHandler* řeší reakce na uživatelské vstupy. V této práci je v metodě *Update* v každém snímku pouze kontrolováno stisknutí *APP tlačítka* (viz sekci 2.6.1), v reakci na nějž je případně změněn aplikační režim pomocí *ApplicationStateControlleru.*

MemberDataManager

Skript *MemberDataManager* získává z *MemberInfoProvideru* informace o členech, které poté spravuje. Správa je řešena pomocí objektu typu *Dictionary,* kde jsou informace organizovány pomocí klíče *Id* u daného člena.

3.3.2 Scéna LocalizationMode

Scéna LocalizationMode implementuje Lokalizační režim (viz sekci 2.2.1). Nacházejí se v ní následující herní objekty:

- LocalizationModeController
- LocationPinTemplate

LocalizationModeController

Skript *LocalizationModeController* zajišťuje chod lokalizačního režimu. Získává od *MemberDataManageru* aktuální údaje o členech, využívá *GeolocationCalculationsService* k jejich zpracování a následně pozice vykresluje pomocí značek (jinak špendlíků, *LocationPinů*) v prostoru.

Za zmínku stojí jeho metody *UpdatePin* a *ScaleFactor*. Metoda *UpdatePin* trasnformuje data na značky v prostoru. Využívá metodu *ScaleFactor,* která určuje velikost pinu – k pozicování jsou totiž využívány reálné jednotky pro vzdálenost (objekt vzdálený 100 metrů bude 100 metrů od kamery ve scéně), je tedy zapotřebí je adekvátně zvětšit, aby byly viditelné.

LocationPinTemplate

Tento objekt je šablonou pro tvorbu značek reprezentujících členy. Do objektu *LocalizationModeController* je dodán pomocí jeho atributu *LocationPinTemplate.* V této práci se jedná o obrácený jehlan, který byl vytvořen v Blenderu.

3.3.3 Scéna SummaryMode

Scéna SummaryMode implementuje Režim přehledu (viz sekci 2.2.2). Nacházejí se v ní následující herní objekty:

- SummaryModeController
- SummaryCanvas
- ScrollbarController
- ToggleManager

SummaryModeController

Tento skript zajišťuje chod režimu přehledu. Proces získání dat je podobný jako u *LocalizationModeControlleru,* ale následně informace vypisuje do karet v objektu *SummaryCanvas.*

SummaryCanvas

Objekt SummaryCanvas je plátno, na kterém jsou vykreslovány jednotlivé karty s informacemi o členech týmu.

Nachází se zde objekt *Scroll View*, který pomocí Unity komponenty ScrollRect zajišťuje, že obsah v něm bude možné rolovat. Bylo zvoleno horizontální rolování. Rolovaným obsahem je jeho potomek, objekt nazvaný *Content,* který má na sebe navázánu Unity komponentu *Horizontal Layout Group*. Ta určuje, že potomci objektu *Content* budou uspořádány do horizontální řady. Na objektu *Content* je také navázána Unity komponenta *Content Size Fitter,*

která zajišťuje, že tyto jednotliví potomci (prvky horizontální řady) nepřesáhnou původní velikost obdélníku *Content,* ale že *Content* prostorově poroste společně s narůstajícím počtem prvků řady.

Potomkem objektu *Content* je *ListItemTemplate,* který je neaktivní, ale vytváří se podle něj nové karty. Jeho potomky jsou již pouze objekty, kam se vypisují data.

Konkrétní objekty, kam jsou data vypisována jsou v kódu nalezeny pomocí jejich názvu metodou *Transform.Find* v UnityEnginu. Není tedy vhodné tyto objekty přejmenovávat.

ScrollbarController

Tento skript zajišťuje reakci horizontální rolovatelné řady karet na uživatelský pohyb prstem po trackpadu. Jelikož se jedná pouze o věc spadající pod režim přehledu, nikoliv celý kontext aplikace, není reakce na tento vstup řešen pomocí skriptu *UserInputHandler,* ale napřímo s využitím NRInputu. V případě že je detekován dotek na trackpadu, je zkoumán relativní pohyb po trackpadu a na základě toho vypočítán posun obdélníka.

ToggleManager

Skript *ToggleManager* zajišťuje správu zaškrtávacích políček na jednotlivých kartách. Zajišťuje zejména kontrolu závislostí mezi nimi, mezi které patří např.:

- Pokud je nově zaškrtnuto políčko *Toto jsem já,* původní políčko *Toto jsem já* je automaticky odškrtnuto
- Na kartě, kde je zvoleno *Toto jsem já* nelze ovládat políčko *Zobrazit v MR*

3.3.4 Scéna StateReportMode

Scéna StateReportMode implementuje Režim hlášení stavu (viz sekci 2.2.3). Nacházejí se v ní následující herní objekty:

- StateReportModeController
- StateReportCanvas

StateReportModeController

Tento skript zajišťuje chod režimu hlášení stavu. Definuje listenery jednotlivým tlačítkům, ty jsou pak vyvolány v metodě *OnPointerClick* u jednotlivých tlačítek vlastního typu *StateReportButton.* Vyvolání *OnPointerClick* již zajišťuje UnityEngine, jelikož *StateReportButton* implementuje rozhraní *IPointerClickHandler.*

StateReportCanvas

Objekt *StateReportCanvas* je plátno, na kterém jsou vykreslena jednotlivá tlačítka pro hlášení stavu. Tlačítka na sebe mají navázán skript *StateReportButton,* popsaný v předchozí sekci o *StateReportModeController.*

3.3.5 Scéna VideoCaptureMode

Scéna VideoCaptureMode implementuje Režim videozáznamu (viz sekci 2.2.4). Nacházejí se v ní následující herní objekty:

- VideoCaptureModeController
- VideoCaptureControlsCanvas

VideoCaptureModeController

Tento skript zajišťuje chod režimu videozáznamu. Tento kód byl částečně převzat z oficiálních příkladů práce s NRSDK, konkrétně z demonstrační scény RGBCamera-Record, která je součástí NRSDK.

VideoCaptureControlsCanvas

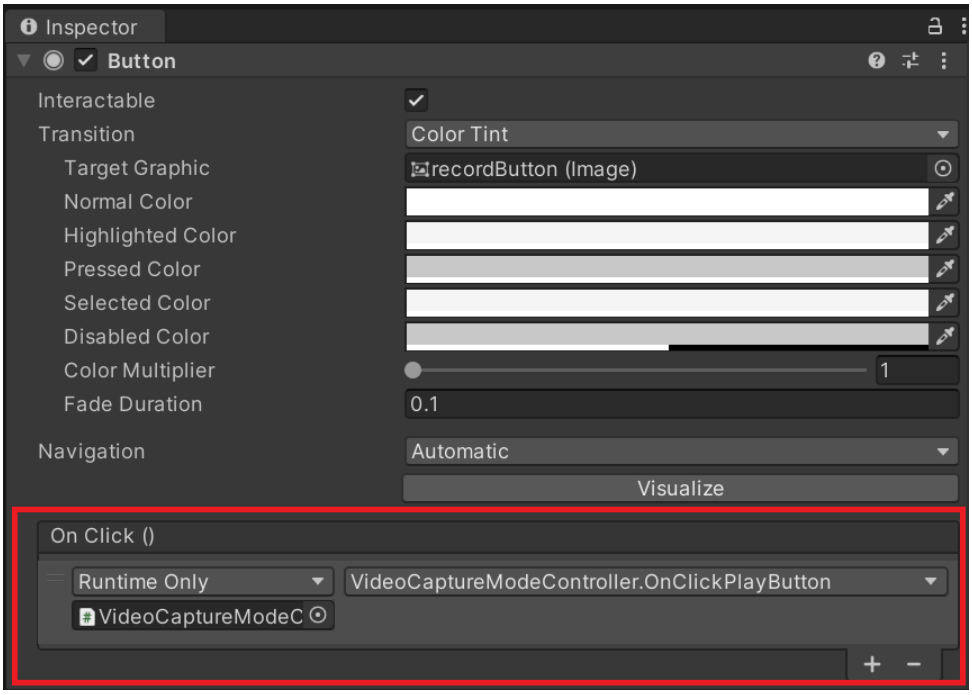
Objekt *VideoCaptureControlsCanvas* je plátno, které obsahuje ovládací prvky videozáznamu. Důležitý je objekt *ControllerPanel* a jeho potomek *recordButton*, u kterého stojí za zmínku způsob navázání listeneru spouštěného kliknutím – přímo v Unity Editoru je u něj v *Inspector Window* přiřazen herní objekt *VideoCaptureModeController* a zvolena funkce která se má v případě kliknutí vyvolat, tedy *OnClickPlayButton.* Tento způsob navázání listeneru je zobrazen na obrázku 3.2.

3.4 Implementace MpsgWeb

Testovací aplikace *MpsgWeb* slouží především jako *MemberInfoProvider*, dále pak pro reportování stavu. Byla vytvořena pomocí *php* jako jednoduchá webová aplikace. Důležité je pro její nasazení mít správně vytvořenou *MySQL* databázi s názvem *mpsg*, kterou lze vytvořit např. pomocí SQL skriptu *mpsg.sql* v adresáři `resources` v kořenovém adresáři projektu.

3.5 Implementace modulu kompasu IotNorthProvider

Kód modulu kompasu se nachází v adresáři `src/arduino` Pomocí Arduino IDE byl vytvořen jednoduchý modul kompasu, využívající ESP8266 a HMC5883L.



Obrázek 3.2: Navázání listeneru na tlačítko pro spuštění videozáznamu

Reprezentuje zde komponentu *IotNorthProvider*. Kalibrace byla řešena způsobem viditelným ve skriptu *calibration.ino,* fungujícím tím způsobem, že uživatel s magnetometrem pomalu otáčí a umožňuje tak programu naměřit minimální a maximální hodnoty, ze kterých je poté při běhu programu odvozeno otočení oproti severu.

Modul kompasu se při zapnutí přihlásí na WiFi hotspot určený v kódu. Je výhodné pro tento modul při testování využívat lokální síť, aby byla komunikace rychlejší. Při takovémto testování, aby nemusela být IP adresa nahlašována vždy znovu, je lokální IP adresa odeslána na v kódu určený server. Kód samotného modulu kompasu se nachází v souboru *iot_north_provider.ino*.

Kapitola 4

Testování

Aplikace byla vzhledem ke své podstatě (jedná se o aplikaci s uživatelským rozhraním na nestandardní platformě) testována manuálními testy. V následujících několika podsekcích jsou popsané navržené scénáře a výsledky testování.

4.0.1 Přepínání režimů

Přepínání režimů může být testováno v jakékoliv variantě nasazení ze sekce 2.9, dostačující je však varianta A, která navíc zaručí data k zobrazení (viz sekci 2.9.1). Pro otestování přepínání režimů byl navržen následující scénář:

1. Uživatel spustí aplikaci *MemberPositionsSmartGlasses.*
2. *Aplikace by měla být spuštěna v režimu přehledu, uživatel uvidí horizontální řadu karet.*
3. Uživatel stiskne tlačítko APP v horní části kruhu na výpočetní jednotce.
4. *Aplikace by měla být přepnuta do lokalizačního režimu, v prostoru okolo uživatele by mělo být zobrazeno několik jehlanů.*
5. Uživatel opět stiskne tlačítko APP.
6. *Aplikace by měla být přepnuta do režimu hlášení stavu, měla by být zobrazena plocha se čtyřmi tlačítkami značícími různé stavy.*
7. Uživatel opět stiskne tlačítko APP.
8. *Aplikace by měla být přepnuta do režimu videozáznamu, měl by být zobrazen panel s ovládacími prvky.*
9. Uživatel opět stiskne tlačítko APP.
10. *Aplikace by se měla přepnout zpět do režimu přehledu.*

Tento scénář lze opakovat několikrát v jednom běhu a výsledek by měl být vždy stejný. Tento test proběhl bez problémů.

4.0.2 Zobrazení pozic členů týmu v lokalizačním režimu

Tento scénář slouží k otestování základního zobrazení pozic členů týmu v rozšířené realitě. Je určen pro variantu nasazení A, která zobrazuje konkrétní statické pozice (viz sekci 2.9.1). Scénář je následovný:

1. Uživatel spustí aplikaci *MemberPositionsSmartGlasses*.
2. Pomocí tlačítka APP přepne aplikaci do lokalizačního režimu.
3. *Aplikace by měla pomocí jehlanů zobrazit následující členy:*
   - #2, Jiří Novotný – 100 metrů směrem dopředu
   - #3, Monika Horská – 1000 metrů směrem doprava
   - #4, Štěpán Hertl – 1000 metrů směrem dozadu doleva
4. Uživatel udělá několik kroků vpřed.
5. *Vzdálenost v metrech od člena #2 zobrazená na štítku u něj by se měla snižovat o jednotky metrů.*

4.0.3 Skrytí a zobrazení pozice člena týmu

Tento test je koncipován pro variantu nasazení A (viz sekci 2.9.1), aby bylo možné přesně určit, které členy má uživatel skrývat a zobrazovat. Pro otestování skrytí nebo zobrazení pozice konkrétního člena týmu byl navržen následující scénář:

1. Uživatel spustí aplikaci *MemberPositionsSmartGlasses*.
2. Uživatel v režimu přehledu deaktivuje zaškrtávací políčko *Zobrazit v MR* u člena s ID 2.
3. *V lokalizačním režimu v prostoru okolo uživatele by se nyní měli nacházet pouze členové #3 a #4.*
4. Uživatel v režimu přehledu deaktivuje zaškrtávací políčko *Zobrazit v MR* u člena s ID 3.
5. *V lokalizačním režimu v prostoru okolo uživatele by se v tuto chvíli měl nacházet pouze člen #4.*
6. Uživatel v režimu přehledu aktivuje zaškrtávací políčko *Zobrazit v MR* u člena s ID 2.
7. *V lokalizačním režimu v prostoru okolo uživatele by se v tuto chvíli měli nacházet pouze členové #2 a #4.*

4.0.4 Nahlášení změny stavu z aplikace v chytrých brýlích

Tento scénář je zapotřebí provádět na variantě nasazení C, aby bylo možné využít funkcionalitu hlášení stavu pomocí testovací aplikace *MpsgWeb*. Scénář je následovný:

1. Uživatel spustí aplikaci *MemberPositionsSmartGlasses.*
2. Uživatel v režimu přehledu aktivuje pole *Toto jsem já* u člena s ID 2.
3. Uživatel v režimu hlášení stavu vybere stav *DANGER 2.*
4. Uživatel v aplikaci *MpsgWeb* načte stránku `/get_positions.php?id=2`.
5. *Ve výsledku by měl být vypsán atribut status s hodnotou 2.*
6. Uživatel v chytrých brýlích v režimu hlášení stavu vybere stav *OK.*
7. Uživatel v aplikaci *MpsgWeb* opět načte stránku `/get_positions.php?id=2`.
8. *Ve výsledku by měl být vypsán atribut status s hodnotou 0.*

4.0.5 Pořízení videozáznamu

Tento scénář je možné provádět na jakékoliv z variant nasazení. Scénář je následovný:

1. Uživatel spustí aplikaci *MemberPositionsSmartGlasses.*
2. Uživatel v režimu videozáznamu pomocí ukazovátka stiskne červené kruhové tlačítko.
3. *Kruhové tlačítko by mělo zezelenat.*
4. Uživatel opakovaným stiskem tlačítka postupně projde přes všechny režimy zpět do režimu videozáznamu.
5. *Kruhové tlačítko by mělo být stále zelené.*
6. Uživatel pomocí ukazovátka stiskne kruhové tlačítko.
7. *Kruhové tlačítko by mělo zčervenat.*
8. Uživatel změní nastavení *Zahrnout MR.*
9. Uživatel pomocí ukazovátka stiskne kruhové tlačítko.
10. *Kruhové tlačítko by mělo zezelenat.*
11. Uživatel po několika sekundách pomocí ukazovátka stiskne kruhové tlačítko.
12. *Kruhové tlačítko by mělo zčervenat.*

Obrázek 4.1: Režim přehledu – demonstrace

13. *V Interním sdíleném úložišti výpočetní jednotky by se nyní měla nacházet 2 nová videa:*
    - Jedno video s viditelnými MR objekty a přepínáním mezi režimy
    - Druhé video bez viditelných MR objektů (neměl by být vidět ani ovládací panel nahrávání videa)

4.1 Zjištěná úskalí

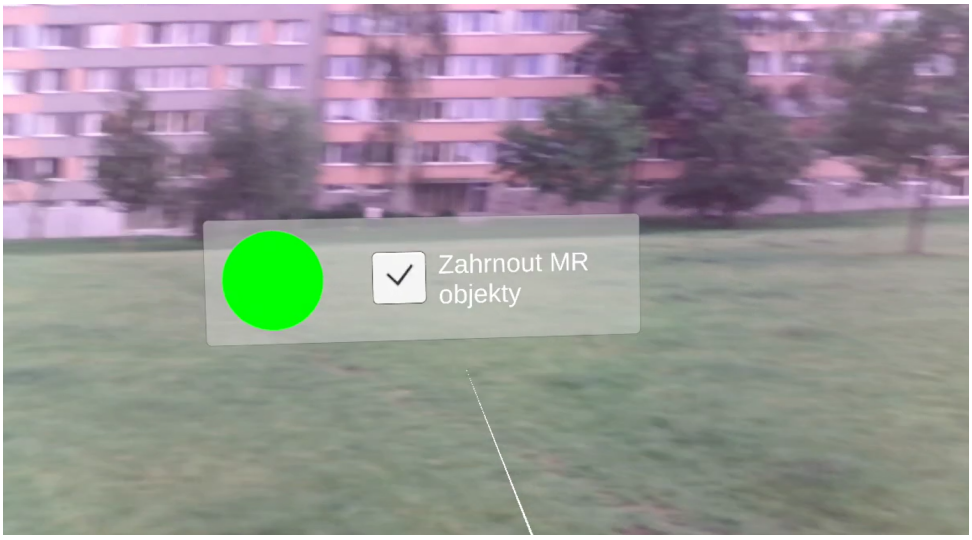
Při testování a nahrávání demonstračního videa bylo zjištěno několik nevýhod souvisejících s použitelností prototypu. Hlavní nevýhodou je, že při silném denním světle byly některé objekty špatně rozpoznatelné, na první pohled neviditelné, jak je možné vidět na některých záběrech z demonstračního videa. Dále se ukázalo být nevhodné testování varianty C v exteriéru (viz sekci 2.9.3) s komponentou MpsgWebMemberInfoProvider fungující přes HTTP. Přestože jednotlivé komponenty samostatně fungovaly jak mají, načítání dat v reálných podmínkách bylo příliš pomalé (oproti komponentě *DtaMemberInfoProvider*). Zároveň cesty nahrávané pomocí mobilního telefonu do GPX bylo potřeba začistit manuálně, kvůli nepřesnosti GPS mobilního telefonu.

4.2 Demonstrační materiály

Pomocí *Režimu videozáznamu* byly pořízeny demonstrační materiály obsažené v archivu `media.zip` v přiložených souborech. Obrázky 4.1, 4.2, 4.3 a 4.4 zobrazují snímky z jednotlivých režimů v hotové aplikaci.

Obrázek 4.2: Režim lokalizace – demonstrace

Obrázek 4.3: Režim hlášení stavu – demonstrace



Obrázek 4.4: Režim videozáznamu – demonstrace

Kapitola 5
Závěr

V této kapitole jsou shrnuty výstupy práce a náměty pro případnou budoucí práci na tomto tématu.

5.1 Shrnutí výstupů

V rámci práce byl vytvořen návrh architektury aplikace pro zobrazování pozic členů týmu v chytrých brýlích. Architektura byla vytvořena z oddělených komponent s dostatečně jemnou granularitou, aby bylo v budoucnu možné komponenty nahrazovat, např. s ohledem na změnu hardwaru na specializovaný hardware vytvořený pro tuto aplikaci.

Pomocí návrhu byl vytvořen prototyp v několika variantách, z nichž první slouží především pro velmi jednoduché testy spojené s chováním aplikace a jejího rozhraní, s každou další variantou se rozsah testů zvyšuje a aplikace se tak přibližuje reálnému případu užití.

Aplikace byla otestována manuálními testy a jednotkovými testy. Obě tyto kategorie pomohly odladit přesnost a použitelnost aplikace, ale reálné testy v exteriéru ve variantě C odhalily úskalí popsaná v sekci 4.1.

Zadání práce bylo splněno, i přes to, že pro vytvoření reálně využitelného systému (nikoliv pouze prototypu) je třeba zavést změny popsané v sekci 5.2.

5.2 Náměty do budoucna

Budoucí práce na tomto tématu by se měla soustředit především na zlepšení přesnosti a použitelnosti systému. Pro lepší použitelnost by měl vzniknout specializovaný hardware, který by integroval modul kompasu a GPS přímo do brýlí, aby nebylo nutné přidělávat modul kompasu přímo na hlavu a muset tak zařizovat další hardware s vlastním zdrojem napětí.

S tím souvisí i použití přesnějšího GPS senzoru, než nabízejí mobilní telefony – při testech v exteriéru často aplikace vykazovala podivné chování způsobené právě nepřesnými daty z telefonu, a způsobovala náročnější ladění.

Posun od tohoto prototypu by mohl pokračovat cestou decentralizace, myšleno využitím jiné sítě než Internetu, pro dosažení kratší odezvy systému

a tím i příjemnějšího uživatelského zážitku a vyšší užitečnosti. Také by tím bylo možné dosáhnout vyšší bezpečnosti.

Příloha A

Seznam použité literatury

1. THAMPAN, Amit; RAZAK, Adil; ABHAY, K; AKASH, R; MANU, M. Evolution of Augmented Reality (AR) and Virtual Reality (VR). *International Journal of Research Publication and Reviews* [online]. 2023, roč. 2023, č. 4, s. 1 [cit. 2024-05-23]. ISSN 2582-7421. Dostupné z: https://ijrpr.com/uploads/V4ISSUE4/IJRPR12239.pdf.
2. LEE, Lik-Hang; HUI, Pan. Interaction Methods for Smart Glasses: A Survey. *IEEE Access* [online]. 2018, roč. 6, s. 28712–28732 [cit. 2024-05-24]. ISSN 2169-3536. Dostupné z DOI: 10.1109/ACCESS.2018.2831081.
3. LOPES, Pedro; YOU, Sijing; ION, Alexandra; BAUDISCH, Patrick. Adding Force Feedback to Mixed Reality Experiences and Games using Electrical Muscle Stimulation. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* [online]. 2018-04-21, s. 1–13 [cit. 2024-05-24]. ISBN 9781450356206. Dostupné z DOI: 10.1145/3173574.3174020.
4. TANG, Karen P.; LIN, Jialiu; HONG, Jason I.; SIEWIOREK, Daniel P.; SADEH, Norman. Rethinking location sharing. *Proceedings of the 12th ACM international conference on Ubiquitous computing* [online]. 2010-09-26, s. 85–94 [cit. 2024-05-24]. ISBN 9781605588438. Dostupné z DOI: 10.1145/1864349.1864363.
5. ČVUT, Katedra Počítačů FEL. *DIGITAL TRIAGE ASSISTANT* [online]. c2024. [cit. 2024-05-24]. Dostupné z: https://cs.fel.cvut.cz/en/page/digital-triage-assistant.
6. TREMOSA, Laia. *Beyond AR vs. VR: What is the Difference between AR vs. MR vs. VR vs. XR?* [online]. [2002]. [cit. 2024-05-24]. Dostupné z: https://www.interaction-design.org/literature/article/beyond-ar-vs-vr-what-is-the-difference-between-ar-vs-mr-vs-vr-vs-xr%5C#what_is_virtual_reality_(vr)?-5.
7. DEMPSEY, P. *The Teardown: HTC Vive virtual reality headset*. Sv. 11 [online]. c2024. [cit. 2024-05-24]. Č. 7. ISSN 1750-9637. Dostupné z DOI: 10.1049/et.2016.0731.

8. ALAKÄRPPÄ, Ismo; JAAKKOLA, Elisa; VÄYRYNEN, Jani; HÄKKILÄ, Jonna. Using nature elements in mobile AR for education with children. In: *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services* [online]. New York, NY, USA: ACM, 2017-09-04, s. 1–13 [cit. 2024-05-24]. ISBN 9781450350754. Dostupné z DOI: 10.1145/3098279.3098547.
9. XREAL. *Hand Tracking* [online]. [2024]. [cit. 2024-05-24]. Dostupné z: https://xreal.gitbook.io/nrsdk/development/hand-tracking.
10. SPEICHER, Maximilian; HALL, Brian D.; NEBELING, Michael. What is Mixed Reality? *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* [online]. 2019-05-02, s. 1–15 [cit. 2024-05-23]. ISBN 9781450359702. Dostupné z DOI: 10.1145/3290605.3300767.
11. XREAL. *Design Guide* [online]. [2024]. [cit. 2024-05-24]. Dostupné z: https://xreal.gitbook.io/nrsdk/design-guide/displaying.
12. NREAL. *Introducing NRSDK* [online]. c2019. [cit. 2024-05-24]. Dostupné z: https://nrealsdkdoc.readthedocs.io/en/latest/Docs/Unity_EN/Discover/IntroducingNRSDK.html.
13. TECHNOLOGIES, Unity. *Rotation and orientation in Unity* [online]. c2024. [cit. 2024-05-24]. Dostupné z: https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html.
14. KUMAR, Muneendra. World geodetic system 1984: A modern and accurate global reference frame. In: *Marine Geodesy* [online]. 1988, sv. 12, s. 117–126 [cit. 2024-05-24]. Č. 2. ISSN 0149-0419. Dostupné z DOI: 10.1080/15210608809379580.
15. CRAIGHEAD, Jeffrey; BURKE, Jenny; MURPHY, Robin. Using the Unity Game Engine to Develop SARGE: A Case Study: A Case Study. *Computer*. 2007.
16. CRAIGHEAD, J.; GUTIERREZ, R.; BURKE, J.; MURPHY, R. Validating the Search and Rescue Game Environment as a robot simulator by performing a simulated anomaly detection task. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* [online]. IEEE, 2008, s. 2289–2295 [cit. 2024-05-24]. ISBN 978-1-4244-2057-5. Dostupné z DOI: 10.1109/IROS.2008.4650746.
17. BRITANNICA. *Latitude and Longitude* [online]. [2024]. [cit. 2024-05-24]. Dostupné z: https://www.britannica.com/science/latitude.
18. XREAL. *Frequently Asked Questions* [online]. [2024]. [cit. 2024-05-24].
19. INC, Honeywell. *3-Axis Digital Compass IC HMC5883L* [online]. c1995-2024. [cit. 2024-05-24]. Dostupné z: https://www.digikey.cz/en/htmldatasheets/production/786600/0/0/1/asd2613-r.
20. TECHNOLOGIES, Unity. *Unity Editor Features* [online]. c2024. [cit. 2024-05-24]. Dostupné z: https://docs.unity3d.com/Manual/EditorFeatures.html.

21. GOOGLE. *Android Debug Bridge (adb)* [online]. [2024]. [cit. 2024-05-24]. Dostupné z: `https://developer.android.com/tools/adb`.

22. ARDUINO. *What is Arduino?* [online]. Monza: Arduino s.r.l., c2021 [cit. 2021-08-10]. Dostupné z: `https://www.arduino.cc/en/Guide/Introduction`.

23. KUSHNER, David. *The making of Arduino: How five friends engineered a small circuit board that's taking the DIY world by storm* [online]. New Jersey: IEEE, c2021 [cit. 2021-08-11]. Dostupné z: `https://spectrum.ieee.org/the-making-of-arduino`.

24. XREAL. *NRSDK Overview* [online]. [2024]. [cit. 2024-05-24]. Dostupné z: `https://xreal.gitbook.io/nrsdk/v/v1.10.2`.

Příloha B

Seznam přiložených souborů

K této práci byly přiloženy 2 soubory:

- `muzatmat_DP2024_projekt.zip` – archiv, obsahuje kořenový adresář projektu, struktura projektu dle sekce 3.1.
- `muzatmat_DP2024_media.zip` – demonstrační média.