ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Bakalářská práce

Rozšířená realita s využitím 360° panoramat

Jan Hamet
Otevřená informatika - Počítačové hry a grafika

Únor 2021, květen 2021
Vedoucí práce: Ing. David Sedláček, Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hamet** Jméno: **Jan** Osobní číslo: **483735**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Rozšířená realita s využitím 360 panoramat

Název bakalářské práce anglicky:

Outdoor AR with 360 panorama

Pokyny pro vypracování:

Seznamte se s možnostmi mobilních platforem pro rozšířenou realitu (např. ARcore, ARkit, AR-foundation, ReactNative AR, ...) s primárním cílem využití venku. Prozkoumejte možný způsob ukládání a sdílení lokálních modelů určených pro tracking (SLAM model) daných technologií (tj. interních datových modelů, které řeší rozpoznání pozice a orientace mobilního zařízení v reálném světě).
Navrhněte a implementujte mobilní multiplatformní aplikaci pro zobrazování více-vrstvých 360° panoramat pomocí vybrané technologie, pokud možno s využitím sdílení SLAM modelů mezi zařízeními. Aplikace bude obsahovat mapu s 'špendlíky', při příchodu na danou lokaci se zobrazí připravené panorama (např. render, fotografie daného místa v jiném čase, nebo jiná doplňková informace) v rozšířené realitě. Jako zdroj dat aplikace použijte portál EduARd.
Otestujte vytvořením datové sady prezentující vybrané části Langweilova modelu Prahy. Ve spolupráci s vedoucím vyberte a vytvořte minimálně 20 panoramatických renderů z digitální verze Langweilova modelu Prahy, které nahrajte do portálu EduARd pro jejich zpřístupnění implementované aplikaci.
Otestuje z výkonostního hlediska, změřte datavé přenosy a kvalitu zarovnání fotografií na různých zařízeních v různých denních/ročních dobách. Porovnejte také kvalitu zarovnání v porovnání s tradičním způsobem odhadu orientace, tj. odhad orientace mobilního zařízení pouze pomocí kompasu a gyroskopu.

Seznam doporučené literatury:

1] Taketomi, Takafumi, Hideaki Uchiyama a Sei Ikeda. Visual SLAM algorithms. IPSJ Transactions on Computer Vision and Applications. 2017, ročník 9
2] Practical Augmented Reality: Steve Aukstakalnis, 2017, Addison Wesley
3] Dieter Schmalstieg, and Tobias Hollerer, Augmented Reality: Principles and Practice (Usability), Addison Wesley 2016
4] https://developers.google.com/ar/
5] https://developer.apple.com/augmented-reality/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. David Sedláček, Ph.D., katedra počítačové grafiky a interakce FEL

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

Datum zadání bakalářské práce: **18.03.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **19.02.2023**

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

/ Prohlášení

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.

Abstrakt / Abstract

Tato bakalářská práce pojednává o využití mobilních knihoven pro rozšířenou realitu při zobrazování 360° panoramat Langweilova papírového modelu Staré Prahy z 19. století.

V úvodní analýze jsou nejdříve rozebrány dostupné nástroje a knihovny pro práci s rozšířenou realitou na mobilních zařízeních a poté i metody zarovnání panoramat s okolní realitou, využívající například sdílení SLAM modelů.

V další části je pak popsána implementace samotné testovací aplikace, jejímž účelem je pomocí mapy s jednotlivými body zájmu dovést uživatele na tato místa a umožnit mu zde zobrazit si vyrenderovaná panoramata.

V poslední části je shrnuto prováděné testování na dvaceti panoramatech v centru Staré Prahy. V tomto testování jsou více rozebrány samotné postupy, výhody a nevýhody využívání zarovnání pomocí ARCore Cloud anchors při umisťování i rozpoznávání. A v poslední řadě je zde shrnuto srovnání orientačních schopností rozšířené reality oproti dříve běžnému užívání gyroskopu s kompasem.

Klíčová slova: Rozšířená realita, Panorama, Langweilův model Prahy, Zobrazování panoramat

This bachelor thesis talks about the use of mobile libraries for augmented reality that are used to show 360° panoramas of Langweil paper model of Prague from 19th century.

First, there are disscussed avaiable tools and libraries for working with augmented reality on devices. After that, there are disscussed methods for alignment panoramas with surrounding reality. This methods can use, for example sharing SLAM models.

In the next section it is described the implementation of the test application. Its purpouse is guide user to the place of iterest and show him prerendered panoramas.

In the last section, the testing performed on twenty panoramas, placed in the center of the Old Prague, is summarized. Here are also disscused methods, advantages and disadvantages of using alignment by ARCore Cloud anchors. In the last, the comparsion of orientation abilities of the ARCore and gyroskop, is summarized here.

Keywords: Augmented reality, ARCore, ARKit, AR Foundation, 360° panorama, Cloud Anchors

Title translation: Augmented reality with 360 degree panoramas

Obsah /

Kapitola 1
Úvod

V posledních letech dochází ke značnému rozvoji knihoven a nástrojů umožňujících tvorbu aplikací pro rozšířenou realitu. Těmi nejpopulárnějšími a nejvíce se vyvíjejícími se staly knihovny ARCore od firmy Google a ARKit od firmy Apple, jež jsou blíže popsány v sekci 1.2.1. Každá z těchto knihoven umožňuje vývojářům na své platformě snadné a rychlé vytváření nových mobilních aplikací využívajících rozšířenou realitu. Na druhé straně svojí uniformitou pro daný operační systém, v tomto případě Android a IOS, znemožňují snadný vývoj multiplatformních aplikací, a tedy nutí vývojáře vytvářet tyto aplikace pro každý systém zvlášť. Pro řešení takovýchto nedostatků existují nad uvedenými knihovnami další rozhraní, které umožňují vytváření multiplatformních aplikací využívajících těchto knihoven v závislosti na daném operačním systému.

Zmíněné knihovny svoje využití naleznou v mnoha aplikacích zábavního, marketingového i edukačního prostředí. Příkladem takových aplikací mohou být kooperativní hry, probíhající skrze mobilní telefon přímo u vás doma, aplikace sloužící k prostorové navigaci v budovách či ve velkých městech a nebo například aplikace umožňující navrhování vybavení bytu přímo na míru.

Cílem mé bakalářské práce je zmapovat výše zmíněné multiplatformní nástroje, vytvořit mobilní aplikaci umožňující na reálných místech v Praze zobrazovat 360° panoramata z papírového Langweilova modelu Prahy a poté provést testování na reálných datech. K prohlížení takových panoramat na mobilních telefonech se dříve využívalo funkcí gyroskopu a kompasu, jež jsou součástí mobilního telefonu. Tato metoda ale není dostatečně přesná a neumožňuje tak jakékoli kvalitní zarovnání virtuálního světa s tím reálným. Z toho důvodu by měla finální aplikace využívat knihovnu pro rozšířenou realitu, která by měla s tímto zarovnáním pomoci.

1.1 Analýza problému

Na aplikaci je kladeno několik základních požadavků, jimiž jsou:

- **Implementace mapy s pozicí panoramat:** Aplikace bude umět zobrazit mapu Prahy s umístěním jednotlivých bodů/míst s panoramaty. Na mapě se také bude zobrazovat poloha zařízení, pokud bude dostupná, a základní informace o každém panoramatu.
- **Orientace v prostoru pomocí knihoven ARCore/ARKit:** Schopnost rozpoznat a dále sledovat pozici mobilního telefonu v reálném prostředí za pomoci knihovny ARCore či ARKit. Dále schopnost zarovnat virtuální pozici panoramatu s reálnou scénou.
- **Zobrazení a zarovnání více-vrstvého panorama:** V našem případě tedy zobrazení reálné scény překryté renderem z Langweilova modelu, který bude s reálnou scénou správně zarovnán.
- **Využití SLAM modelů:** Pro přesnější sledování/trackování reálného prostředí bude využívat sdílení SLAM modelů.
- **Multiplatformita:** Aplikace bude dostupná jak pro mobilní operační systém Android, tak IOS.

- **Komunikace se serverem EduARd:** Aplikace bude získávat data k zobrazovaným panoramatům a externím modelům ze serveru EduARd.
- **Alternativa pro nonAR mobilní telefony:** V případě nekompatibility zařízení s uvedenými knihovnami bude orientace v reálném prostředí nahrazena gyroskopem a kompasem.

1.2 Rozbor dostupných nástrojů pro rozšířenou realitu

Jelikož je primární funkcí této aplikace zobrazování panoramat v reálném prostředí, rozeberu zde nejdříve již výše zmíněné knihovny ARCore a ARKit. Tyto knihovny jsou v současnosti nejdostupnějším a nejvíce se vyvíjejícím nástrojem pro tvorbu mobilní rozšířené i virtuální reality. Obě knihovny v základu pracují na podobných principech a mají podobné funkce. Z toho dúvodu tu více rozeberu především platformu ARCore, s níž budu díky testovacím zařízením, které využívají OS Android, pracovat především. V závěru nynější kapitoly se podíváme na dvě multiplatformní rozhraní nad těmito knihovnami, jež se snaží propojovat funkce obou knihoven tak, aby bylo možné tvořit jen jednu aplikaci pro oba systémy.

1.2.1 ARCore

ARCore od firmy Google je platforma sloužící vývojářům mobilních aplikací pro snadnější a rychlejší implementaci rozšířené reality. Skrze několik mnoho APIs[1] nabízí přístup k základním funkcím využívaných v rozšířené, ale i virtuální realitě. Těmito základními funkcemi, jež budou více rozebrány dále, jsou Motion tracking (sledování pohybu), Enviromental understanding (rozpoznání prostředí) a Light estimation (vyhodnocení světelných podmínek). Tyto tři funkce poté společně umožňují mobilnímu telefonu, skrze pohled kamery a interní senzory, porozumět svému reálnému okolí a zobrazovat v něm virtuální objekty.

Motion tracking využívá proces nazývající se SLAM (simultaneous localization and maping). Jedná se o proces, který díky dostupným senzorům umožňuje vyhodnotit pozici zařízení vůči okolnímu světu a dále ji sledovat. V ARCoru je pak konkrétně využit vSLAM (visual SLAM) společně s IMU jednotkou (Inertial Measurement Unit). Při tomto procesu si aplikace vytváří ze získaných vizuálních dat přibližnou virtuální mapu významných bodů, se kterou následně porovnává další vstupní data a vyhodnocuje podle nich pozici zařízení. VSLAM získává vizuální vstupní data za pomoci vizuálních senzorů, v tomto případě tedy fotoaparátu mobilního telefonu. Samotný proces se pak skládá z pěti částí, jimiž jsou inicializace, sledování polohy, mapování, relokalizace a optimalizace vytvořené mapy.

- **Inicializace** vSLAMu se provádí při spuštění algoritmu a dochází při ní k prvotnímu nastavení globálního systému souřadnic a k rekonstrukci úvodní mapy okolního prostředí. Tato mapa se vytvoří základním porovnáním dvou obrázků pomocí five-point algoritmu [1]. Ve vstupních obrázcích se pomocí tohoto algoritmu nalezne 5 důležitých bodů (tzv. feature points) a z nich se vyhodnotí základní pozice kamery.
- **Sledování polohy** (neboli dále trackování) se pak provádí porovnáním jednoho vstupního obrázku s mapou uložených feature points.
- V **mapování** se do stávající mapy přidávají nové feature points. Pro ně je nejdříve vypočítána pozice v 3D prostoru, k čemuž se využívá triangulace z konkrétních klíčových snímků. Tyto nové body jsou pak následně ukládány do mapy.

[1] `https://cs.wikipedia.org/wiki/API`

- **Relokalizace** slouží k znovunalezení polohy zařízení vůči vytvořené mapě poté, co dojde k přerušení trackování. K tomuto se využívá algoritmů pro náhodné prohledávání stromu (randomized tree-based searching).
- **Optimalizace mapy** slouží k eliminování chyb způsobených například rychlým pohybem kamery. Provádí se porovnáváním každého klíčového snímku vůči ostatním. Ve chvíli, kdy je nalezena chyba, dojde k opravě zachycených pozic kamery v celém grafu/mapě.

IMU jednotka se pak skládá z akcelerometru, gyroskopu a magnetometru a slouží k měření rotačních sil působících na zařízení. Toho se pak společně s vSLAMem využívá k vyhodnocení skutečné pozice a rotace kamery. Blíže o vSLAMU viz [2] a o IMU jednotce viz [3].

Enviromental understanding neboli rozpoznávání a porozumění prostředí umožňuje mobilnímu telefonu rozpoznávat horizontální a vertikální plochy (anglicky planes), které pak předkládá aplikaci pro možnost umístění 3D objektů. Detekování takových ploch probíhá tak, že prohledává svou mapu významných bodů a hledá v nich skupiny, jež se nacházejí na stejné rovině (např. desce stolu či zdi).

Light estimation ze vstupních obrázků vypočítá průměrnou intenzitu a barevnou korekci světla a předá ji aplikaci jako informaci k osvětlení virtuálního 3D objektu. To následně umožňuje přirozenější nasvícení virtuálního objektu. Toto nasvícení se provádí buď přímým předáním vypočítaných hodnot, nebo pomocí HDR mapy prostředí, kterou ARCore dokáže vygenerovat.

Další, pro naši aplikaci neméně důležitou funkcí ARCoru, je sdílení SLAM modelů pomocí Cloud Anchors, tedy tzv. Cloud Anchor sharing, jemuž se ale budeme blíže věnovat v samostatné sekci 1.3.1.

ARCore také umožňuje i další funkci, kterou je Interakce s uživatelem, kdy je po kliknutí uživatelem vyslán z tohoto místa dotyku paprsek do mapy významných bodů a je vrácena pozice zasaženého významného bodu či plochy. Toho lze poté využít k umístění virtuálního objektu.

Další funkcí je sledování obrázků, které umožňuje rozpoznat předem definované obrázky a zobrazit na nich libovolný virtuální objekt.

Poslední funkce umožňuje u mobilních telefonů s hloubkovým senzorem tvorbu hloubkové mapy, jež pomáhá aplikaci rozhodnout, zda není virtuální objekt překryt jiným reálným objektem či lidskou rukou.

Asi největší nevýhodou ARCoru v současné době je nutnost vlastnit mobilní zařízení, které je uvedeno v seznamu podporovaných zařízení na stránkách Google Developers [2]. Pro ostatní zařízení není podpora knihovny ARCore zajištěna a aplikace jej využívající nelze na těchto zařízeních spustit. Více o ARCore lze nalézt na stránkách Google Developers viz [4].

Pokud jde o srovnání rozdílů platformy ARCore s platformou ARKit, současná verze ARKitu (ARKit 4)[3] oplývá kromě výše zmiňovaných funkcí ARCoru i funkcí rozpoznávání předmětů a jejich rekonstrukcí (meshingem), sledováním pohybu lidského těla

[2] `https://developers.google.com/ar/discover/supported-devices`
[3] `https://developer.apple.com/augmented-reality/arkit/`

a v nejnovějších telefonech i přesnějším sledováním využívajícím LiDAR skener (konkrétně jde o smartphony iPhone 12 Pro, iPhone 12 Pro Max a iPadu Pro). Velkou výhodou oproti ARCoru je také funkce World anchors, která umožňuje podobně jako výše zmiňované Cloud anchors sdílení virtuálního zážitku s dalšími uživateli.

1.2.2 AR Founfation

AR Foundation je balíček do herního enginu Unity, jež umožňuje vývoj multiplatformní aplikace pro rozšířenou realitu využívající právě ARCore nebo ARKit v závislosti na své cílové platformě. Neimplementuje žádné nové AR prvky, pouze nabízí rozhraní (API) přistupující k těmto funkcím na již zmíněných platformách.

Samotné AR Foundation je postaveno na subsystémech. Každý subsystém je platformně nezávislé rozhraní se specifickou funkcionalitou. Příkladem může být rozhraní XRPlaneSubsystem jež poskytuje přístup k detekci horizontálních a vertikálních ploch.

Ke svému fungování na cílových platformách, potřebuje AR Foundation dále i balíčky pro dané specifické systémy. Těmito balíčky jsou ARCore XR Plugin pro Android a ARKit XR Plugin pro IOS. Ty pak následně poskytují implementaci daných subsystémů.

Unity také nabízí souhrn aktuálně podporovaných funkcionalit (implementovaných subsystémů) na jednotlivých platformách, jež lze vidět na obrázku 1.1.
Podrobnější informace a zdroj obrázku viz [5].

	ARCore	ARKit	Magic Leap	HoloLens
Device tracking	✓	✓	✓	✓
Plane tracking	✓	✓	✓	
Point clouds	✓	✓		
Anchors	✓	✓	✓	✓
Light estimation	✓	✓		
Environment probes	✓	✓		
Face tracking	✓	✓		
2D Image tracking	✓	✓	✓	
3D Object tracking		✓		
Meshing		✓	✓	✓
2D & 3D body tracking		✓		
Collaborative participants		✓		
Human segmentation		✓		
Raycast	✓	✓	✓	
Pass-through video	✓	✓		
Session management	✓	✓	✓	✓
Occlusion	✓	✓		

Obrázek 1.1. Seznam podporovaných funkcionalit v AR Foundation.

1.2.3 ViroReact

ViroReact od firmy ViroMedia je další platforma sloužící ke snadnému vývoji multiplatformních AR aplikací, tentokrát za využití React Native[4]. Tato platforma se skládá ze dvou hlavních částí, jimiž jsou:

- Nativní 3D renderovací engine sloužící k zobrazování obsahu (3D objektů, obrázků, videa,..).
- Konkrétní rozšíření Reactu pro vytváření virtuální a rozšířené reality.

Tyto rozšíření pak, konkrétně v případě rozšířené reality, využívají již nám známý ARCore a ARKit.

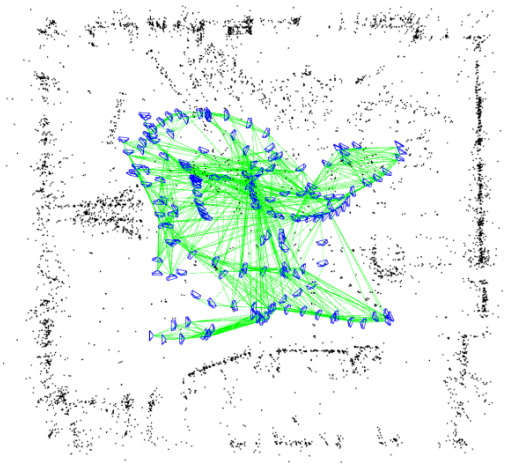
ViroReact bohužel nenabízí plnou podporu všech funkcionalit implementovaných v ARCoru a ARKitu. Mezi těmi chybějícími je především absence funkce Cloud Achors umožňující sdílet svůj virtuální zážitek s ostatními a dále novější funkce ARKitu jako je sledování lidského těla a meshing.

ViroMedia také nabízí i platformu ViroCore, což je rozšíření pro vývojáře Android aplikací umožňující rychlejší a snadnější vytváření AR aplikací využívajících pouze ARCore.

Tato firma po posledním vydání verze ViroReact v2.17.0 v říjnu 2019 ukončila svůj vývoj a otevřela ViroReact i ViroCore jako OpenSource program. Více viz [6].

1.3 Sdílení SLAM modelů

SLAM model sestává z virtuální 3D mapy významných bodů (feature points), jež se vytváří při trackování reálného prostředí mobilním zařízením. Takovýto model pak může sloužit k rychlejší inicializaci nového zařízení na daném místě tak, že převezme tuto mapu a začne podle ní porovnávat vizuální data z kamery. V takovémto SLAM modelu poté také lze vytvářet různě umístěné anchory (kotvy), ke kterým se mohou fixovat virtuální objekty, a lze tak například sdílet stejný virtuální zážitek s více uživateli.



Obrázek 1.2. Vizualizace SLAM modelu společně s vyhodnocením jednotlivých pozic kamery při jeho vytváření. Zdroj obrázku [7].

[4] `https://reactjs.org`

1.3.1 Cloud anchors

Cloud anchors je funkce umožňující uživatelům přidávat do AR scény virtuální objekty a sdílet je s několika dalšími uživateli. Uživatelé pak mohou s objekty také interagovat ve stejném sdíleném reálném prostředí. Příkladem využití mohou být například různé kooperativní virtuální stavebnice, využívající jako základnu stůl uprostřed místnosti, nebo trvalé umístění virtuálních objektů na konkrétních místech tak, že jsou viditelné i pro další uživatele. Samotný Cloud anchor je tedy ve skutečnosti virtuální bod v SLAM 3D mapě s přesně zadanými souřadnicemi.

Pro svoji funkci využívají Cloud anchors připojení ke službě ARCore Cloud Anchor API, skrze kterou provádí sdílení a rozpoznávání těchto anchorů. To je důvod proč všechny aplikace využívající Cloud Anchors potřebují aktivní internetové připojení. Práce s Cloud anchory se dělí na dvě části, jimiž jsou hosting (sdílení), během kterého uživatel sdílí svůj Cloud anchor do cloudové služby, a resolving (rozpoznávání), během kterého za pomoci této služby rozpoznává a hledá přesné umístění anchoru v reálné scéně.

Hosting. Pro sdílení Cloud anchoru ARCore využívá takzvanou 3D feature mapu (dříve zmiňovaný SLAM model) kolem vybraného anchoru. Pro získání této mapy je třeba dostatečně kvalitně zmapovat objekt či oblast zájmu ze všech úhlů. Čím kvalitnější bude výsledná mapa, tím robusnější a snadnější bude pozdější rozpoznávání Cloud anchoru. Po dostatečném zmapování okolí našeho anchoru je možné zavolat metodu *ARAnchorManager.HostCloudAnchor*, díky níž ARCore nahraje vizuální data, pozici anchoru a kamery do ARCore Cloud Anchor service. Ta si z dat vytvoří finální 3D mapu a vrátí unikátní CloudAnchorID, díky kterému jej půjde dále rozpoznat.

Limitace velikosti této 3D mapy není v dokumentaci blíže specifikována, ale podle dostupných pokusů lze vytvořit mapu až o velikosti menšího náměstí.

Resolving. ARCore periodicky volá metodu *ARAnchorManager.ResolveCloudAnchorId* pro vybrané CloudAnchorID a kontinuálně tak odesílá vizuální data z kamery zařízení do ARCore Cloud Anchor API. Zde se tyto data porovnávají s uloženou 3D mapou a pokud nalezne shodu, vrátí služba pózu (pozici a orientaci) anchoru.

Cloud anchor sharing je primárně určený ke sdílení lokálních zážitků/prvků na jasně odlišitelných objektech reálného prostředí. Tedy primárně pro zážitek dějící se kolem nějakého reálného objektu (nábytku, sochy apod.), jež může sloužit jako centrální bod zájmu. Rozpoznávání okolních velkých ploch, jako v našem případě ulic či náměstí, již není pro tuto funkci tak jednoduché. Proto je třeba při hostingu provést dostatečně kvalitní mapování většího okolí tak, aby se dosáhlo kvalitní mapy pro pozdější rozpoznávání. Zároveň se zde ale objevuje i omezení v podobě závislosti na konkrétních světelných podmínkách a ročním období.

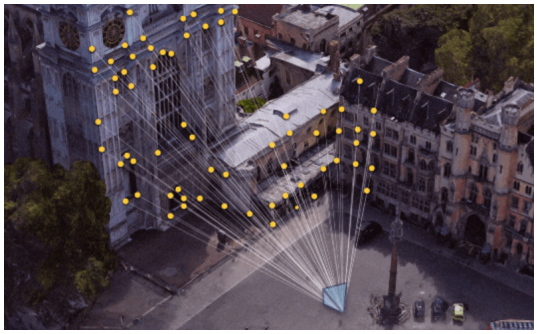
Pro funkci Cloud anchor sharing je také potřeba propojit svoji aplikaci s Google Console a ve vytvořeném projektu přidat ARCore Cloud Anchor API. To pak vyžaduje po aplikaci autentifikaci pomocí API klíče či OAuth klienta. Životnost vytvořených Cloud anchorů je také omezena na maximálně 365 dní (pouze 1 den při použití API klíče). Bližší informace jsou dostupné zde [8].

1.3.2 Earth cloud anchors

Nebo-li také AR Location anchors, jak se nazývají u ARKitu jsou určeny primárně ke sdílení AR zážitku ve vnějších prostorách. Pro svoji funkci využívají lokalizaci pomocí GPS v kombinaci s globální lokalizací.

Globální lokalizace je funkce, jež se v současné době zpřístupňuje pro navigaci ve velkých světových městech, kde je rozpoznávání přesné polohy pomocí GPS díky okolnímu rušení značně nepřesné, a tak se tato funkce kombinuje s VPS (Visual positioning service). Ta umožňuje za pomocí kamery zařízení porovnávat vizuální data s uloženou 3D mapou prostředí a při shodě tak vyhodnotit přesnou pozici zařízení. Pro správné fungování je třeba mít dostupnou kvalitní mapu okolního prostředí ve všech denních dobách i ročních obdobích a algoritmus schopný tyto změny v případě nutnosti odfiltrovat tak, aby tato lokalizace fungovala kdykoli a kdekoli. Více o globální lokalizaci lze nalézt v tomto článku [9].

Apple ve svém ARKitu již má tuto funkci v současné době implementovanou a lze ji takto využít k umisťování anchorů v cca 50 velkých světových městech, které má pro tento účel dostatečně zmapován. Google v současné době pracuje na propojení této globální lokalizace s AR cloud anchors, a tak je zde tato funkce zatím nedostupná.



Obrázek 1.3. Vizualizace funkce VPS rozpoznávající reálnou scénu.

1.4 Orientace v prostoru pomocí vnitřních senzorů

K orientaci v prostoru, především pak k vyhodnocení rotace mobilního zařízení, se nejběžněji využívá kombinace tří senzorů, jimiž jsou kompas, gyroskop a akcelerometr. Tyto senzory bývají nejčastěji společně součástí jedné mobilní IMU jednotky.

Kompas v mobilním zařízení pracuje na principu magnetometru, který měří okolní magnetická pole. Kvůli tomu může být ale jeho přesnost ovlivněna okolními magnetickými poli dalších zařízení a tak jeho měření nejsou stoprocentně přesná.

Akcelerometr měří zrychlení ve třech osách XYZ. A používá se tedy především k měření změny polohy zařízení. Dokáže ale také měřit úhel natočení zařízení, jelikož zaznamenává také směr gravitační síly, díky čemuž je i v klidovém režimu vertikální osa vždy vychýlena. Nám poté stačí znát maximální výchylky na těchto osách a lze z nich vypočítat výsledný úhel natočení. Nevýhodou měření rotace zařízení pomocí akcelerometru je jak fakt, že jeho naměřené hodnoty nejsou pro použití ve VR/AR aplikacích dostatečně přesné, tak skutečnost, že tuto rotaci dokáže měřit jen v horizontálních osách, a tedy pro měření směru potřebuje nějaký další senzor, například kompas.

Gyroskop měří úhlovou rychlost ve třech osách. Lze z něj tedy získat všechny tři rotace roll, pitch i yaw k přesnému určení rotace zařízení. Také není ovlivněn, pokud dojde k náhlému zatřesení či jinému zrychlení zařízení, tak jako v případě akcelerometru. Díky své přesnosti je vhodný pro odhad orientace ve VR/AR aplikacích.

1.4.1 Sensor fusion

Použití samostatného gyroskopu, může někdy způsobit, že za běhu aplikace bude docházet ke driftu. Tedy chybě v odhadované rotaci, což může mít za následek až konstantní ujíždění (drift) obrazu v jednom směru. Proto, aby se tento drift vykompenzoval, se používá pro určování orientace spojení dat dalších z výše zmíněných senzorů, které se kombinují za použití komplementárního filtru.

Data z akcelerometru spojeného s magnetometrem obsahují vysokofrekvenční šum, který je třeba před vstupem do komplementárního filtru odstranit pomocí low-pass filtru. Data z gyroskopu naproti tomu obsahují drift, jež se, jak už bylo zmíněno, projevuje postupným narůstáním chyby a lze jej tak odstranit high-pass filtrem. V komplementárním fitru se už jen hodnoty zkombinují podle předem specifikovaných vah. Tím dostaneme přesné měření gyroskopu s odstraněným driftem. Více informací k implementaci takového filtru, například v Javě, zde [10].

1.5 Mapové nástroje

Pro správné nalezení lokací s panoramaty a Cloud anchory je třeba navigovat uživatele na tato místa pomocí mapy. Běžně se pro zobrazování map využívají rastrové nebo vektorové tiles (dlaždice), které umožňují rychlý a nenáročný datový přenos po internetu a zároveň nám takto umožňují renderovat jen konkrétní oblast mapy s libovolným zoomem.

Každá dlaždice mapy má běžně rozměr 256x256 pixelů a patří do konkrétní úrovně zoomu (přiblížení). Zoom na úrovni 0 se skládá pouze z jedné dlaždice a obsahuje mapu celé Zeměkoule. Počet dlaždic na každou další úroveň se pak vždy násobí čtyřmi.

V našem případě bychom chtěli jako mapu využít 2D render Langweilova papírového modelu Prahy z ptačí perspektivy. V něm pak budeme následně zobrazovat polohu jednotlivých lokací společně s polohou uživatele. Jelikož bude tato bude operovat jen v rámci Staré Prahy, postačí nám tak mapa zobrazující pouze tuto oblast.

1.5.1 Mapbox - dynamicky generovaná mapa

Mapbox je jedna z nejrozšířenějších platforem pro dynamickou mapovou navigaci a mapové zobrazování jak pro webové prohlížeče, tak pro mobilní aplikace. Skrze svou platformu nabízí Mapbox přístup k mapovým podkladům, informacím o dopravě i vizuální navigaci využívající také AR.

Mapbox dále nabízí rozšíření do herního enginu Unity, umožňující vývojářům získávat mapová data z reálného světa pro vlastní aplikace a také dále umožňuje propojení těchto dat i s rozšířenou realitou pro tvorbu location-based AR.

Cena za využívání nástrojů této platformy je závislá na počtu měsíčních uživatelů. Do 25 tisíc uživatelů měsíčně je její užívání zdarma. Více informací k této platformě lze nalézt zde [11].

1.5.2 Lokální mapa

Lokální mapa vyžaduje uložení mapových dat přímo na zařízení a tím pádem zabírá aplikace, využívající tento druh mapy, výrazně větší datový prostor než dynamické mapy. Zároveň pokud při využití takovéto mapy potřebujeme synchronizovat pozici v reálném světě, nejčastěji udávanou za pomocí GPS souřadnic, je třeba provést projekci z tohoto sférického/eliptického souřadnicového systému do 2D souřadnicového systému pixelů. Toho lze docílit využitím projekce s názvem Web Mercator projection.

Web mercator projection je v současné době standardem pro drtivou většinu webových mapových aplikací a jak už bylo zmíněno výše, umožňuje konverzi GPS souřadnic do pixelových 2D souřadnic na mapě. Vstupní GPS souřadnice se skládají z úhlových souřadnic zeměpisné délky (longitude) a zeměpisné šířky (latitude).

Zeměpisná šířka se měří od hlavního poledníku procházejícího Královskou observatoří v Greenwich v Anglii a nabývá kladných hodnot ve východním a záporných hodnot v západním směru. Zeměpisná délka se naproti tomu měří od rovníku a je oproti zeměpisné délce konstantní. Pro přepočet jednotek zeměpisné šířky je totiž třeba uvažovat také aktuální zeměpisnou délku, jelikož poloměr například polární rovnoběžky bude značně menší než poloměr rovníku.

Web mercator projection využívá k projekci souřadnic tuto formuli:

$$x = \lfloor (256/2\pi) \cdot 2^{zoomlevel} \cdot (\lambda + \pi) \rfloor pixels$$

$$y = \lfloor (256/2\pi) \cdot 2^{zoomlevel} \cdot (\pi - \ln [\tan (\pi/4 + \varphi/2)] \rfloor pixels$$

Kde λ je zeměpisná šířka a φ je zeměpisná délka. Výstupem rovnic jsou souřadnice pixelu na mapě v dané úrovni přiblížení, které lze již poté využít k umístění bodu v mapě. Více o web mercator projeci zde [12].

Jelikož náš render nebude ve standardizovaném formátu pro web mercator, je třeba jej k tomuto přizpůsobit. Toho docílíme tím, že nejdříve nalezneme úroveň zoomu s podobným rozlišením na metr čtvereční a poté pomocí afinní transformace přetransformujeme souřadnicový systém do druhého systému, či podle toho transformujeme samotný render. Takovou transformační matici lze získat například pomocí funkce Matlabu s názvem *fitgeotrans()*, jíž předáme určité množství párů shodných bodů v obou obrázcích či souřadných systémech. Další možností může být využití programu MapTiler Desktop, který umožňuje interaktivně vybírat tyto body přímo v mapě a poté nám vyrenderovat takto transformovaný obrázek přímo jako vrstvu v mapových tilech. Tento program ale v neplacené verzi pracuje pouze s obrázky s menším rozlišením než 10000px v delší straně a tak jej můžeme využít pouze jako referenci pro pozdější transformování obrázku s větším rozlišením.

1.6 Zobrazení více-vrstvého panoramatu

V tomto případě je naším úkolem zobrazit jedno panorama, které překrývá nějakou reálnou scénu. Dalším rozšířením ale může být například přidání další vrstvy panoramatu v podobě popisků vyobrazovaných objektů.

360° panoramata se běžně ukládají v podobě klasické 2D fotografie, jež je ale od pohledu lehce deformovaná. Fotografie se pak jako textura mapuje na objekt sféry, v našem případě na její vnitřní stranu. Sféru poté zobrazujeme kolem uživatele a pro získání efektu poloprůhlednosti nám stačí míchat texturu panoramatu s texturou snímku kamery na základě zvolené intenzity průhlednosti.

Pro možné přidání dalších vrstev, konkrétně popisků, v budoucích verzích aplikace lze využít tyto dva postupy:

- Přidání další sféry s panoramatem. Tato možnost vyžaduje přidat další totožnou sféru, na které by se zobrazovalo panorama obsahující pouze popisky budov. Mezi

vzniklými vrstvami panoramat se pak může jednoduše přepínat změnou hodnoty aplha každého panoramatu či renderováním vždy pouze jedné sféry.

Panorama s popisky by bylo pravděpodobně vytvářet při renderování z Langweilova modelu tak, že by se tyto popisky nejdříve umístili na svá místa kolem kamery a poté by se zvlášť vyrenderoval pohled do modelu a pohled do panorama s popisky.

- Zobrazování lokačně závislých popisků za pomoci platformy MapBox. Jednou z mnoha funkcí této platformy je možnost zobrazovat v rozšířeně realitní scéně objekty umístěné podle své GPS pozice. Toho by se tak dalo využít k zobrazení popisků kolem význačných památek, které by nebyli závislé na pozici uživatele a pozici panoramatu.

1.7 Rešerže podobných aplikací

Při hledání inspirace u podobných aplikacích jsem narazil především na aplikaci MARK AR. Tato aplikace pracuje na velmi podobném principu, na jakém by měla fungovat i výsledná aplikace mé bakalářské práce. Mezi další aplikace, které mne při mém průzkumu zaujali, byli Google Street view a funkce Live View, jež je součástí aplikace Google maps.

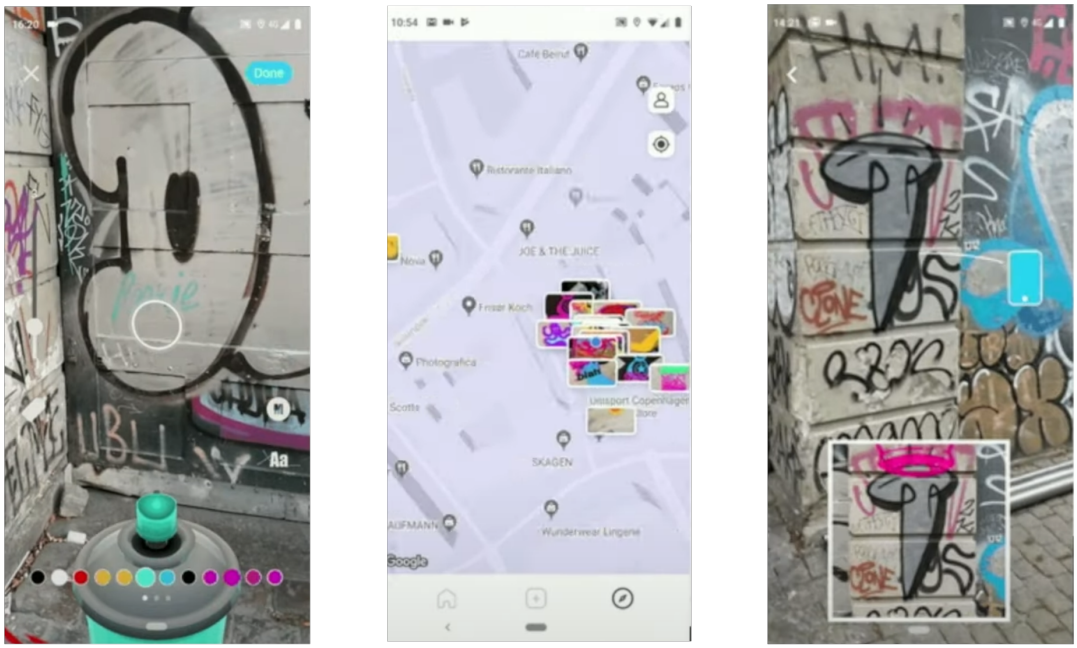
MARK [5] je mobilní aplikace dostupná pro Android i IOS vytvořená v herním enginu Unity. Dle svého vlastního popisu se jedná o rozšířeně realitní sociální síť, jež umožňuje na různých místech po světě vytvářet virtuální značky v podobě nápisů, nálepek, a především grafiti a ty pak sdílet s dalšími uživateli, kteří se je mohou vydat hledat. Aplikace se dělí na dvě hlavní části, jimiž jsou mapa s rozmístěním již vytvořených grafity společně se sociálními příspěvky a AR scéna, v níž se vytvářejí nové grafity a hledají/zobrazují nově nalezené. K umisťování těchto grafity využívá MARK funkce Cloud anchors zmíněnou již v předchozích sekcích. AR režim je zde vytvářen za pomoci knihovny ARCore v případě Android zařízení, nebo ARKit v případě IOS.

Jak už bylo zmíněno, AR režim se dělí na dvě části, kterými jsou vytvoření grafity a jeho nalezení. Vytvoření grafity probíhá v několika krocích:

- Nejprve se naskenuje plocha, na kterou chce uživatel kreslit, tak aby se vytvořila dostatečně podrobná 3D feature mapa.
- Poté je uživatel instruován, aby tapl na naskenovaný povrch, čímž se zahájí proces vytváření Cloud anchoru.
- V tuto chvíli může uživatel začít kreslit samotné grafity stejně jako na obrázku 1.4 vlevo.
- Po dokončení a publikování grafity se metadata obsahující samotné grafity, návodný obrázek místa umístění a především pak Cloud anchor ID vytvořeného anchoru uloží do Mark AR databáze.

Proces nalezení tohoto konkrétního grafitti probíhá pak už jen zjištěním Cloud anchor ID jeho anchoru společně s načtením jeho metadat z databáze. Toto ID se pak pošle do ARCore Cloud Anchor API a uživatel už je jen naváděn na konkrétní místo dokud ARCore Cloud Anchor API nenalezne shodu. Aplikace bohužel není v České republice dostupná. Blíže je tato aplikace popsána v přednášce o dlouho trvajících Cloud anchorech dostupné zde [6].

[5] `https://www.mark.app/pc/index.html`
[6] `https://youtu.be/op-zs0mJOHO`



Obrázek 1.4. Snímky z aplikace MARK AR: Levý obrázek - kreslení grafitti. Prostřední - mapa s již vytvořenými grafitti. Pravý obrázek - hledání Cloud anchoru.

Google Street view [7] je aplikace od firmy Google, umožňující uživatelům nejen zobrazit si ve svém mobilním telefonu panoramatické snímky z různých koutů světa, ale také nahrávat další.

Aplikace má tedy hlavní dvě funkce:

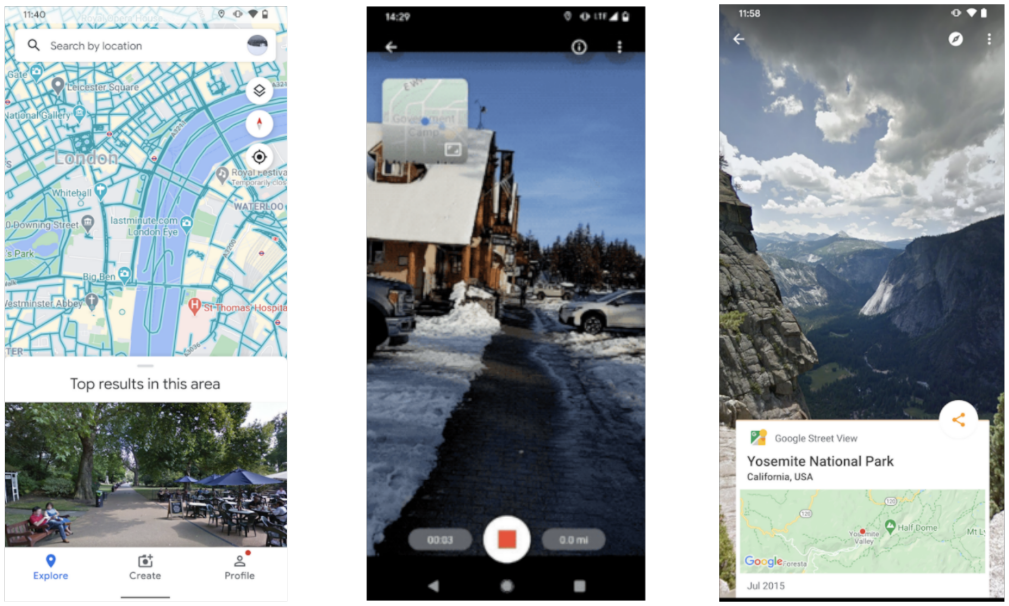
- První z nich je prohlížení 365° panoramat, za pomoci posouvání obrazu prstem po obrazovce nebo otáčením samotného zařízení. Toto automatické otáčení využívá ke své funkci pouze běžné vnitřní senzory zařízení a jsou tak zde vidět drobné nedokonalosti v podobě driftu obrazu.
- Další funkcí je vytváření a sdílení nových panoramatických snímků a to buď v podobě jednotlivých 360° panoramat, nebo kontinuálního snímání okolní scény za chůze. Zmíněné kontinuální snímání využívá funkcí knihovny ARCore, která ze získaných snímků skládá celá panoramata. Uživatel tak nemusí vlastnit speciální 365° kameru a přesto se může zapojit do rozšiřování světové databáze Google Street view.

Aplikace také nabízí zobrazení mapy, na které lze vidět pokrytí street view snímky. To lze na mapě vidět v podobě plných modrých a šrafovaných čar. Modré zobrazují pokrytí speciálními Street view auty a šrafované ukazují snímky od uživatelů. Snímky z aplikace lze vidět na obrázku 1.5.

Google maps Live view [8] zpřístupňuje uživatelům nový způsob navigace v podobě navigačních značek a tras v rozšířené realitě. Za tímto účelem je zde využívána knihovna ARCore, která pomáhá rozpoznat prostředí kolem uživatele a umisťovat do něj tyto navigační značky.

Aplikace je také schopna rozpoznávat významné památky a zobrazovat u nich dodatečné popisky popisující například název této památky a vzdálenost od uživatele. Poslední schopností aplikace je nasměrování uživatele na polohu vybraných objektů a

[7] `https://blog.google/products/maps/anyone-can-share-their-world-with-street-view/`
[8] `https://blog.google/products/maps/new-sense-direction-live-view/`

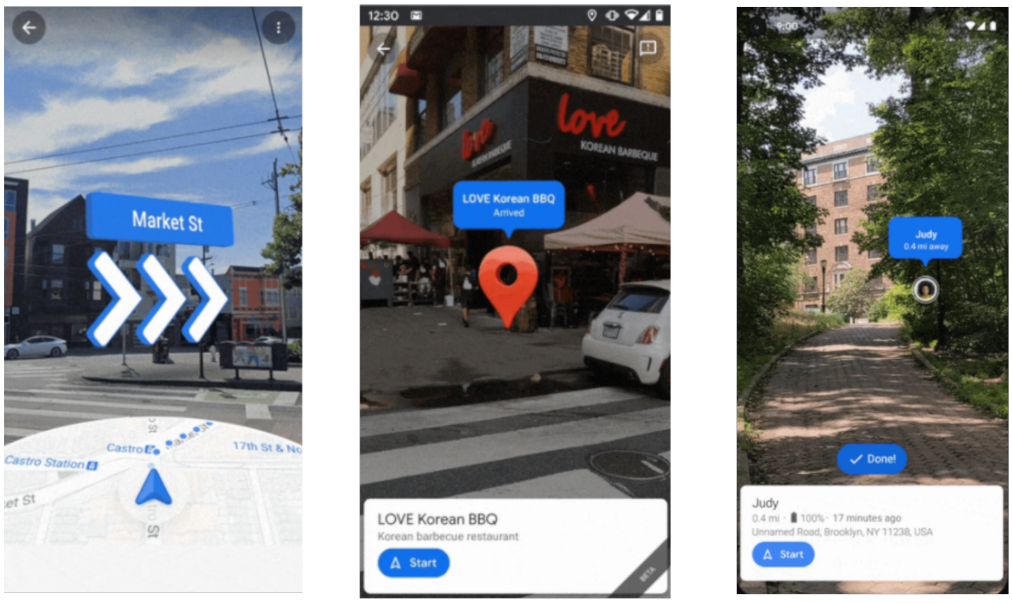


Obrázek 1.5. Snímky z aplikace Google Street view: Levý obrázek - mapa pokrytí. Prostřední - nahrávání nových snímků. Pravý obrázek - zobrazení panorama.

osob. Toho se docílí vybráním daného cíle a přepnutím do Live view. Zde je uživatel vyzván k naskenování svého okolí, tak aby Live view rozpoznal kde se nachází. Po tomto rozpoznání už dokáže zobrazit přesný směr a vzdálenost místa kde se cíl nachází.

Do Live view módu se lze z běžných Google map přepnout pouze v lokalitách s dostatečně kvalitním pokrytím v Street view databázi, jež je využívána jako reference pro rozpoznání polohy uživatele.

Snímky z aplikace lze vidět na obrázku 1.6.



Obrázek 1.6. Snímky z aplikace Google map - Live view: Levý obrázek - navigace na místo. Prostřední - zobrazování bodů. Pravý obrázek - hledání přátel.

Kapitola 2
Řešení

Součástí mého samostatného projektu předcházejícího této bakalářské práci bylo otestování a porovnání nástrojů pro rozšířenou realitu zmíněných v předchozí kapitole. K tomuto účelu jsem vytvořil tři mobilní aplikace. Jednu nativní, psanou v programovacím jazyce Kotlin určenou pouze pro Android a využitou tedy především pouze jako referenční zdroj, druhou psanou v Unity za využití AR Foundation a třetí psanou v React nativ za využití knihovny ViroReact. Každá z těchto aplikací dokázala po spuštění zobrazit pohled skrze fotoaparát kamery zkombinovaný s pohledem do testovacího 360° panoramatu.

Na těchto aplikacích jsem poté pozoroval rozdíly ve schopnostech orientace v reálném prostředí a rychlosti reakce na pohyb zařízení. Z testování jsem vyvodil, že požadované orientační schopnosti těchto aplikací jsou téměř totožné.

Další částí mého průzkumu bylo zjistit, jak jsou na tom tyto platformy s funkcí sdílení SLAM modelů, popsaných také více v přechozí kapitole. ViroReact, kvůli ukončení svého vývoje v roce 2019, již nemá ve své funkci implementovanou podporu Cloud anchors a proto je pro mou budoucí aplikaci nevyhovující. Jelikož je u této aplikace vyžadováno i multiplatformní využití, je z tohoto důvodu nejvhodnějším nástrojem Unity s jeho AR Foundation.

2.1 Návrh řešení

Výsledná aplikace se bude skládat ze tří částí/režimů. Těmi jsou: navigační mapa s vyobrazením pozice jednotlivých panoramat, administrátorský režim pro tvorbu Cloud anchorů a uživatelský režim pro zobrazení panoramat a rozpoznávání Cloud anchorů. Dále bude aplikace komunikovat s ARCore Cloud Anchor API a edukačním portálem EduARd ze kterého si bude stahovat potřebná panoramata.

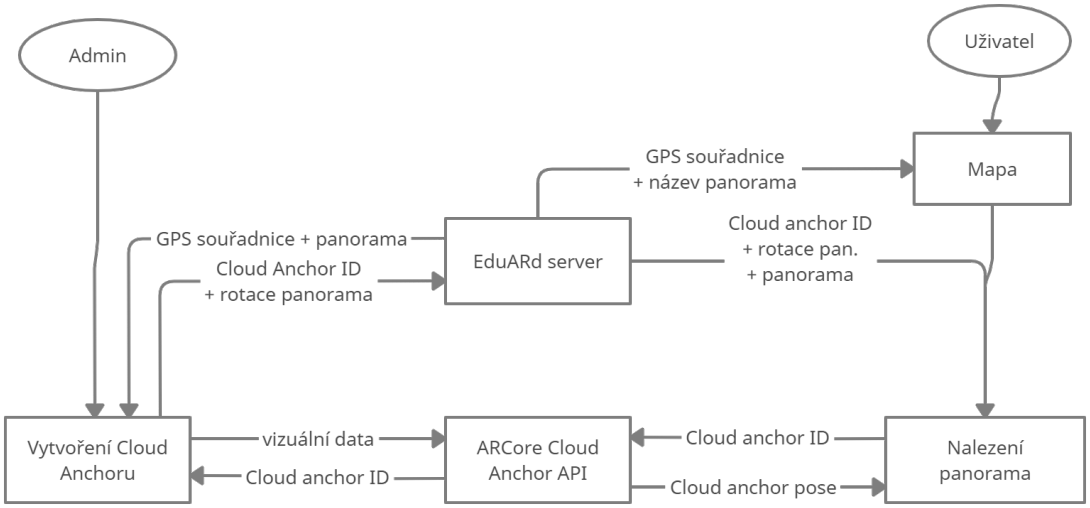
Jako mapový podklad pro samotnou mapu by měl sloužit render Langweilova modelu Prahy. Samotná mapa by pak měla umět zobrazovat pozici jednotlivých lokací s panoramaty a uživatelovu polohu. Po příchodu na místo s panoramatem by aplikace měla nabídnout uživateli přepnutí do AR režimu, kde se mu zobrazí samotné panorama. V tomto režimu bude panorama prvotně zarovnáno pouze podle údajů kompasu a pokud dojde během prohlížení panoramatu k nalezení Cloud anchoru, zarovná se panorama podle něj.

Aplikace bude dále také v samostatném buildu obsahovat administrátorský režim, jež bude umožňovat vytvářet jednotlivé Cloud anchory pro každé panorama. Vždy si bude pamatovat datum vytvoření, Cloud anchor ID a rotaci panoramatu pro přesné zarovnání. Tyto údaje bude poté možno vyexportovat do dokumentu .csv, jež se bude dále využívat k nahrávání těchto dat na platformu EduARd. Na platformě EduARd se dále budou ukládat ke každému panoramatu tyto informace:

- Název panoramatu/místa

- Samotný soubor obsahující panorama
- Pozici panoramatu udanou v GPS souřadnicích
- Cloud anchor ID příslušného Cloud anchoru
- Rotaci panoramatu

Aplikace by také měla obsahovat českou, anglickou a německou lokalizaci. Níže na obrázku 2.1 je nastíněno schéma přenosu dat uvnitř aplikace a její komunikace s externími službami.



Obrázek 2.1. Schéma přenosu dat mezi částmi aplikace.

2.2 Implementace

Aplikace je implementována v herním enginu Unity, verzi 2020.3.1 s přidanou URP (Universal Render Pipeline)[1], využívanou především k vykreslování sféry panoramatu. Dále můj projekt obsahuje balíčky AR Foundation, ARCore XR Plugin a ARCore Extension, jež obsahuje rozšíření umožňující využívat v aplikaci funkci Cloud Anchors.

Implementaci lze rozdělit na několik jednotlivých částí, kterými jsou:

- Mapa a mapové funkcionality
- Uživatelský panorama režim
- Administrátorský AR režim
- Komunikace se serverem EduARd

2.2.1 Mapa a mapové funkcionality

Jelikož je jedním z požadavků na tuto aplikaci zobrazení mapy v podobě renderu z Langweilova modelu, nebylo možné v základu využít funkce platformy MapBox a jejích dynamických mapových podkladů. Z tohoto důvodu bylo třeba implementovat vlastní základní mapové funkce. Prvním krokem bylo upravení mapy Langweilova modelu tak, aby bylo možné využít běžnou Web mercator projection (dále WMP) pro správné zobrazování bodů v mapě. Nejdříve jsem se pokusil vytvořit projekční matici, jež by umožňovala transformaci souřadnic z WMP do souřadnic Langweilvy mapy.

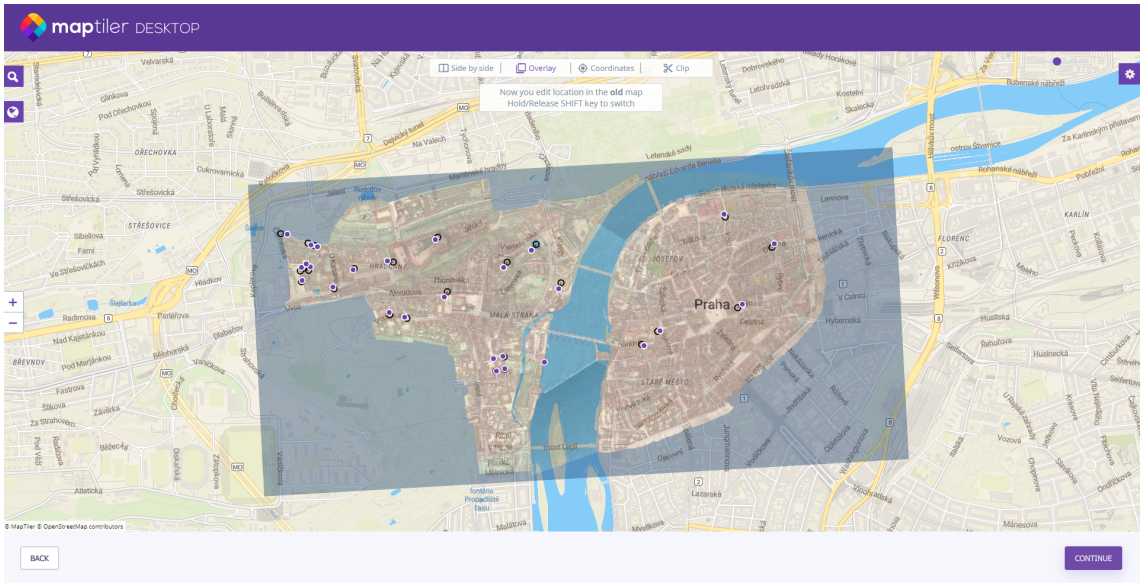
[1] `https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@10.2/manual/index.html`

K tomu jsem využil program Matlab a funkci **fitgeotrans()**, jež slouží běžně k výpočtu projekční transformace mezi dvěma obrázky. Vstupem této funkce jsou páry shodných bodů v obou obrazech ve formě souřadnic. Bohužel při aplikování takto získané transformace na souřadnice získané z WMP nebylo dosaženo dostatečné přesnosti v umístění zadaných bodů.

V druhém pokusu jsem využil programu MapTiler desktop [2], díky kterému jsem namapoval mapu Langweilova modelu na standartní mapy určené pro Web mercator. K tomuto účelu využívá program MapTiler stejný postup jako dříve zmíněná funkce Matlabu, s rozdílem možnosti určování těchto shodných párů bodů interaktivně přímo z obrázků. Výsledkem je afinní transformace, jež se aplikuje na obrázek Langweilova modelu tak, aby se přizpůsobil standartní mapě viz obrázek 2.2.

Na výstupu tohoto druhého pokusu už je vizuálně patrné, že Langweilův model, a tudíž i mapa využívaná v mé aplikaci, je lehce odkloněný od skutečného rozložení budov a ulic v oblasti Staré Prahy. Toto je dobře patrné na obrázku 2.3, kde je v pravé části vidět soupis jednotlivých shodných bodů a nalezené odchylky po provedení afinní transformace.

Pro odstranění těchto nedostatků jsem výsledný obrázek po afinní transformaci ještě upravil v obrázkovém editoru s pomocí Warpové transformace tak, aby hlavní ulice a orientační body odpovídali referenční standartní mapě pro WMP.



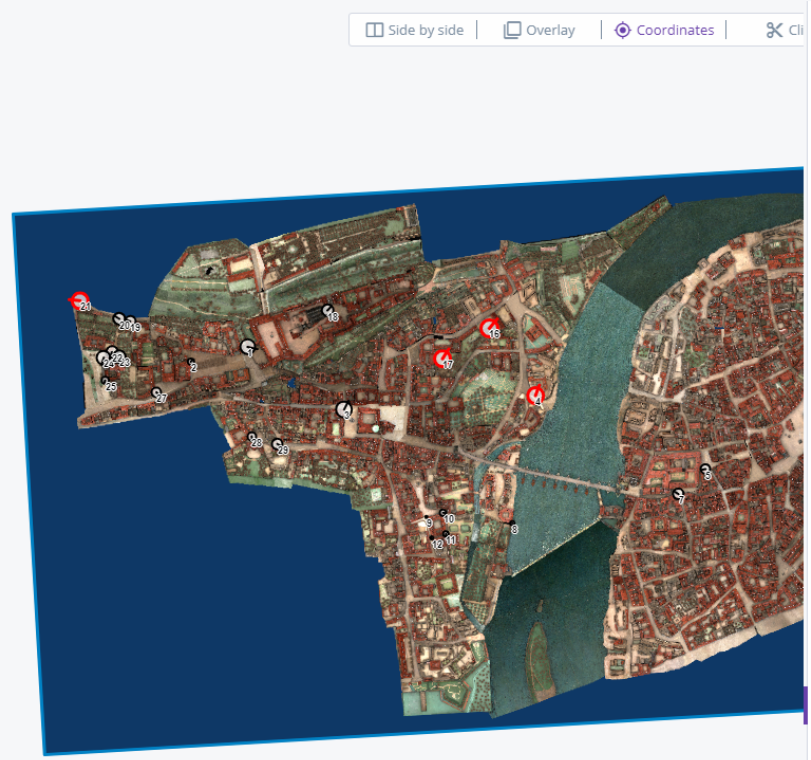
Obrázek 2.2. Obrázek zobrazující afinní transformaci aplikovanou na mapu Langweilova modelu.

Finální rozlišení této mapy je 15240 x 8192px a použitá úroveň zoomu pro WMP je 19. Tato mapa je v Unity vložena do samostatného canvasu, jež je nezávislý na kameře. Souřadnicový systém mapy má počátek v levém horním rohu a výsledné pozice bodů na mapě se pak z WMP souřadnic přepočítávají už jen posunutím počátku souřadnicové soustavy.

Běžná uživatelská interakce s mapou v podobě scrollování a zoomu je pak převáděna do pohybu kamery v osách X a Y v případě scrollu a Z v případě zoomu.

O projekci z GPS souřadnic, pohyb v mapě a další mapové funkce se starají tyto třídy:

[2] `https://www.maptiler.com`



	X	Y	Error [px]
1	1602797.052	6461861.153	94.17
2	1602483.765	6461809.009	20.86
3	1603233.565	6461622.154	91.50
4	1604152.288	6461696.359	103.17
5	1604930.365	6461303.160	36.55
7	1604801.344	6461199.894	37.14
8	1604015.498	6461056.668	7.35
9	1603605.380	6461081.528	2.28
10	1603692.512	6461099.967	15.76
11	1603697.048	6460998.135	10.12
12	1603626.704	6460983.662	4.23
13	1605573.285	6461495.069	94.59
14	1605853.037	6461981.584	69.04
15	1605472.009	6462225.652	54.33
16	1603951.766	6462003.481	113.30
17	1603717.592	6461863.152	107.89
18	1603155.175	6462062.128	54.89
19	1602163.191	6461998.245	47.93
20	1602109.536	6462015.098	53.83
21	1601892.655	6462089.585	101.40
22	1602084.269	6461816.372	63.44
23	1602114.202	6461797.115	63.53
24	1602046.848	6461788.660	65.01
25	1602059.813	6461704.057	21.87
27	1602320.252	6461646.721	32.66
28	1602769.661	6461438.134	34.32
29	1602908.352	6461407.103	45.39

Obrázek 2.3. Vyobrazení odchylek jednotlivých bodů po affiní transformaci.

- **ProjectionTransform** jež obsahuje funkci *LatLongToPixels*, která zajišťuje přepočet GPS souřadnic na souřadnice v mapě.
- **MapUserInputManager** jež zajišťuje pohyb kamery, a tedy i scroll a zoom v mapě, podle uživatelových gest. Při scrollování si tento skript kontroluje, zda se uživatel dotkl jedním prstem obrazovky a v každé další iteraci aplikace vypočítává vektor mezi předchozí a novou pozicí na obrazovce. Ten je poté vynásoben vhodnou konstantou získanou testováním a aplikován na pozici kamery v osách X a Y. Při zoomu se kontroluje, zda dochází k dotyku dvěma prsty a v každém snímku (framu) aplikace vypočítává vzdálenost mezi těmito dvěma prsty. Rozdíl těchto vzdáleností oproti předchozímu framu se pak po vynásobení konstantou aplikuje na pozici kamery v ose Z. Nakonec jsou obě tyto hodnoty vynásobeny koeficientem závislým na aktuálním zoomu mapy.
- **MapMarkPlacementManager** se stará o umisťování značek s vyznačením míst pro panoramata do mapy. Po zavolání funkce vždy smaže předchozí značky, projde seznam dostupných panoramat a pro každé znich umístí značku na pozici získanou transformací z GPS souřadnic. Značky se takto umisťují do statického canvasu již obsahujícího samotnou mapu.
- **MapPanelController** se stará o zobrazování podrobností o panoramatu po rozkliknutí dané značky. Také omezuje pohyb v mapě, pokud je zobrazeno info, a povoluje jej při zaregistrování opětovného dotyku mimo oblast informačního panelu. O zobrazení informací o správném panormatu se stará skript **MapMarkInfo**.

Dále jsou zde pomocné scripty jako **MapMarkScaller**, jenž se stará o přeškálování mapových značek kdykoli dojde k změně úrovně zoomu, a **DevicePozitionMapMark**, jenž

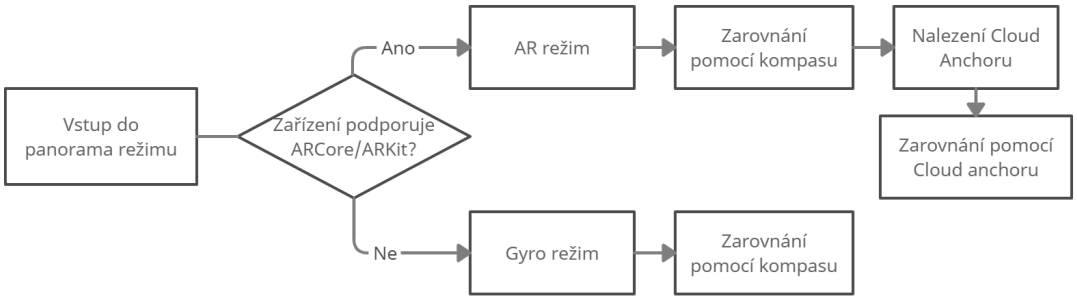
podle vstupních souřadnic a rotace mění pozici a natočení značky označující polohu uživatele a jeho zařízení.

O zjišťování polohy zařízení se stará script **DeviceLocationManager**, který si nejdříve při inicializaci aplikace, zjišťuje zda má práva pro sledování polohy a poté si v každé smyčce vyžádá údaje o poloze. Dále také měří rotaci zařízení vůči severnímu pólu a umožňuje k této hodnotě přístup dalším částem aplikace. Toto natočení vůči severnímu pólu je získáváno výpočtem průměru z hodnot v posledních dvaceti měřeních tak aby se odfiltrovaly nechtěné výchylky měření. V případě že nemůže z nějakého důvodu získat informace o aktuální poloze, zobrazí uživateli varovnou zprávu. Vizuální náhled mapové části aplikace lze vidět na obrázku 2.5 v levo.

2.2.2 Uživatelský panorama režim

Uživatelský AR režim se dělí na dvě samostatné scény (AR a Gyro) v závislosti na tom, zda mobilní zařízení podporuje AR knihovnu či nikoliv. Obě tyto scény se při volání aditivně načítají do hlavní scény a mažou při odchodu z panoramatického režimu. Zobrazení panoramatu v tomto režimu se provádí namapováním textury panoramatu na vnitřek sféry, která se ve scéně instanciuje kolem uživatele.

O přiřazení konkrétní scény a výsledného způsobu zarovnání panoramatu se rozhoduje podle diagramu zobrazeného na obrázku 2.4



Obrázek 2.4. Schéma postupu v uživatelském panorama režimu.

První AR scéna využívá funkce dříve zmiňovaných knihoven ARCore a ARKit a obsahuje tyto základní objekty:

- **AR Session**, jež ve scéně kontroluje celý cyklus rozšířené reality. Běžně také kontroluje, zda má uživatel nainstalovánu podporu pro AR (ARCore/ARKit) na svém zařízení a vynucuje jeho instalaci či update.
- **AR Session Origin** slouží k transformaci trackovatelných features (feature points a rovin) z prostoru AR session, kde jsou všechny objekty umístěny relativně k počátku AR session, do prostoru Unity. Zároveň se také využívá k renderování point cloudu či horizontálních a vertikálních rovin. AR Session Origin si dále uchovává odkaz na kameru, která je mu přiřazena jako potomek. Kamera obsahuje AR pose driver, jež řídí její pozici v AR prostoru. Dále obsahuje skript AR Camera manager, který především zpřístupňuje obrázky/textury z reálné kamery pro vykreslování pozadí. Dále také nabízí funkci Light estimation pro počítání okolního osvětlení.
- **ARCoreExtensions** se stejně nazývaným scriptem. Ten umožňuje aplikaci přístup k objektům AR foundation, které právě toto rozšíření využívá pro práci s Cloud anchory.

- **Canvas**, který obsahuje tlačítko pro návrat do mapového režimu a scrollbar, který se využívá pro změnu úrovně průhlednosti panoramatu. Zároveň je zde také snackbar pro zobrazování debugovacích zpráv.
- **PanoramaClientManager** jež kontroluje celý cyklus této AR scény. Při vstupu do scény nejprve načítá texturu konkrétního panoramatu. V případě, že ji nemůže najít v adresáři na zařízení, požádá o její stažení ze serveru. V dalším kroku provede umístění sféry s panoramatem do scény na aktuální pozici kamery a orotuje jí tak, aby byla zarovnána s magnetickým severním pólem. Tato rotace se spočítá tak, že se od úvodní rotace kamery odečte změřený úhel k severnímu magnetickému pólu získaný z DeviceLocationManageru.

  Po inicializaci sféry se provede samotné volání Cloud anchor API pomocí metody *ResolveCloudAnchorID*, kde jako vstupní proměnou posíláme Cloud anchor ID daného panoramatu. V tuto chvíli se začnou cyklicky odesílat vizuální data do Cloud anchor API a zde se porovnávají s referenční mapou dříve vytvořeného Cloud anchoru. Po nalezení shody se původně instanciovaná sféra nahradí novou, jejímž rodičem se stane pozice právě nalezeného anchoru. Na takto instanciovanou sféru se pak už jen aplikuje rotace, jež je uložena společně s dalšími informacemi o panoramatu na serveru EduARd.

Druhá Gyro scéna je určena pro starší zařízení, které nepodporují knihovny ARCore a ARKit a jak už vyplývá z názvu, využívají ke své orientaci v prostoru zejména měření získaná z gyroskopu. Gyro scéna obsahuje tyto objekty:

- **Main Camera**. Tato kamera je dynamická a slouží k zobrazování sféry. Je řízena scriptem **FreeARCameraManager**, který ji po inicializaci přiřadí jako potomka pod vytvořený objekt CameraContainer. Na tento objekt je pak následně v každém updatu aplikována rotace získaná z gyroskopu. Drift gyroskopu je zde kompenzován algoritmem využitým z dřívějšího projektu naší katedry.

  Dále tento script při inicializaci pomocí WebCamTexture prochází dostupné fyzické kamery zařízení, z nichž vybírá zadní kameru a texturu získanou z této kamery nastavuje jako pozadí pro komponent RawImage umístěný v CameraTexture Canvasu. V průběhu aplikace se pak dle rozlišení této textury vypočítává aspect ratio a předává se do aspect ratio fitteru, který se stará o to, aby textura vyplňovala celou oblast obrazovky.
- **Background Camera**. Tato kamera je statická a pouze snímá CameraTexture Canvas a umožňuje tak zobrazit na displeji pohled fyzické kamery.
- **CameraTexture Canvas** v němž se nachází RawImage určený pro namapování textury z fyzické kamery.
- **UI Canvas**. Tento canvas obsahuje stejně jako u AR režimu scrollbar umožňující změnu průhlednosti panoramatu, tlačítko pro návrat do mapového režimu a dále také snackbar pro zobrazování debugovacích zpráv.
- **FreeARController**. tento objekt obsahuje script FreeARManager, který se stará o načítání textury panoramatu a zarovnání sféry podle kompasu.

Dále se zde už přímo nachází objekt sféry s panoramatem, který je zarovnán se severním pólem díky tomu, že počáteční nulová rotace sféry vždy směřuje na sever.

2.2.3 Administrátorský režim

Administrátorská část aplikace slouží k vytváření Cloud anchorů pro jednotlivá panoramata. Na hlavní obrazovce tohoto režimu se zobrazují informace k panoramatu právě vybranému z dostupné rozbalovací nabídky. Těmito informacemi jsou:

- Název panoramatu
- Cloud anchor ID
- Datum vytvoření a expirace Cloud anchoru
- Název souboru s panoramatem
- Rotace panoramatu
- Světové souřadnice

Dále se zde nachází tlačítko pro export .csv souboru s daty všech panoramat, jehož účel bude vysvětlen v sekci 2.2.4 a tlačítko umožňující vstup do administrátorského AR režimu pro tvorbu Cloud anchorů. Náhled této obrazovky lze vidět na obrázku 2.5 uprostřed.

Administrátorský AR režim se opět nachází v samostatné scéně a stejně jako uživatelské režimy se v případě potřeby aditivně načítá k hlavní scéně.

Součástí této scény jsou:

- **AR Session**, **AR Session Origin** a **ARCoreExtensions** stejně jako u uživatelského režimu.
- **Canvas**. Ten obsahuje stejně jako v uživatelském režimu scrollbar pro změnu průhlednosti, snackbar pro debugovací zprávy a tlačítko pro návrat na administrátorskou obrazovku.

  Mimo to obsahuje ale také textové pole pro instrukce, tlačítka pro potvrzení uložení rotace panoramatu a zahájení procesu nahrávání Cloud anchoru a slider pro nastavení počtu dnů životnosti Cloud anchoru.
- **PanoramaHostManager**, který obsahuje řídící script tohoto režimu **ARHostManager**.

Řídicí cyklus lze rozdělit do několika fází. Na počátku řídícího cyklu se nejdříve stejně jako v uživatelských režimech provede načtení textury panoramatu. Poté je uživatel vyzván k tapnutí na obrazovku. Z místa dotyku s obrazovkou se vyšle paprsek do scény a pokud narazí na některou z horizontálních AR plane, jež vytváří ARCore při rozpoznávání reálného prostředí, dojde v tomto místě k vytvoření anchoru.

V tomto bodě se instanciuje sféra s panoramatem a uživatel je vyzván k manuálnímu zarovnání poloprůhledného panoramatu s reálnou scénou, čehož docílí pohybem prstu po obrazovce.

Po potvrzení rotace tapnutím na tlačítko pro uložení se sféra nahradí prefabem vizuálně pomáhajícím základnímu naskenování okolí Cloud anchoru. Pro zjišťování kvality naskenovaného/nasnímaného prostředí se využívá funkce *EstimateFeatureMapQualityForHosting*, jež jako vstup přijímá pózu kamery. Tato funkce vrací jednu ze tří hodnot Bad, Sufficient a Good odpovídající kvalitě vytvořené SLAM mapy z aktuálního pohledu kamery. Instanciovaný prefab vizuálně zobrazuje tyto hodnoty v podobě červené, oranžové a zelené barvy.

Po základním naskenování okolí tohoto prefabu ze všech úhlů a překonání hranice *featureMapQualitytreshold* je zpřístupněno tlačítko pro zahájení nahrávání Cloud anchoru na server. Takto vytvořená základní feature mapa ale není často dostačující pro náš styl využívání Cloud anchorů a tak je nutné provést detailnější naskenování i širšího okolí. K indikaci kvality naskenování zde slouží QualityMark jež je součástí canvasu. Toto lze vidět na obrázku 2.5 vpravo.

Po tom, co administrátor nastaví pomocí slideru požadovanou životnost cloud anchoru a potvrdí zahájení sdílení, zavolá se funkce *AnchorManager.HostCloudAnchor*. Ta provede volání do ARCore Cloud anchor API a začne odesílat získaná vizuální data.

Po dokončení nahrávání je uživateli/administrátorovy zobrazena zpráva o výsledku nahrávání a v případě úspěchu se provede uložení času vytvoření, rotace panoramatu a

cloud anchor ID vráceného z ARCore Cloud anchor API. Uživatel je poté automaticky vrácen na hlavní obrazovku administrátorského režimu.

2.2.4 Komunikace se serverem EduARd

Již z diagramu na obrázku 2.1 je vidět, že ke každému panoramatu se na serveru ukládá hned několik informací. Těmito informacemi jsou název, GPS souřadnice, obrázek panoramatu, rotace panoramatu a Cloud anchor ID. Tyto údaje o panoramatu jsou na serveru uloženy ve formě jednotlivých úkolů. Tyto úkoly jsou pak dohromady součástí jedné knížky/učebnice. Komunikace se serverem probíhá za pomoci třídy UnityWebRequest. V tomto procesu komunikace jsem se inspiroval u dalšího projektu z předchozích let, který umožňoval stahování různých assetů z tohoto serveru.

Prvním krokem při této komunikaci je přihlášení k serveru, tedy získání přístupového tokenu po inicializaci aplikace. V současnou chvíli se token získává zadáním přihlašovacích údajů ke konkrétnímu účtu. To by mělo být později nahrazeno API klíčem. Zároveň se tímto voláním kontroluje, zda má zařízení přístup k internetu. Pokud by tomu tak nebylo, je uživatel upozorněn zprávou a kontrola připojení je poté periodicky opakována.

Po získání přístupového tokenu se ze serveru získává informace o knížce obsahující požadovaná panoramata. Zde se vyhledá údaj o poslední aktualizaci knížky a pokud je tato aktualizace novější, než je poslední verze knížky uložená v zařízení, zahájí se její stahování.

Po stažení této knížky ve formě XML souboru se z ní postupně načtou veškerá data a uloží se do datové struktury panoramat. Tato struktura má z důvodu snadnějšího prohledávání formu slovníku kde klíčovým názvem je název panoramatu. Pokud ve struktuře dané panorama již existuje dojde pouze k aktualizování hodnot získaných ze serveru.

Jelikož mohlo při aktualizaci knížky dojít k změně souborů s panoramaty, je uživatel dotázán na jejich opětovné stažení. V případě že uživatel svolí, dojde ke smazání již nepotřebných souborů a provede se stažení všech nových souborů s panoramaty do lokálního adresáře. V případě že uživatel odmítne, probíhá stahování těchto panoramat individuálně při vstupech do panoramatických režimů.

Server z důvodu bezpečnosti neumožňuje upload nových dat z aplikace, proto je třeba data získaná při tvorbě Cloud anchorů v administrátorském režimu, zadat na server manuálně ve webovém prohlížeči. Pro snadnější čitelnost těchto dat je využíváno exportu do CSV souboru.

2.2.5 Hlavní řídící smyčka aplikace

Tato smyčka se nachází ve scriptu **MainAppController** a jak už vyplývá z názvu, kontroluje průběh celé aplikace a zajišťuje přesun mezi jednotlivými scénami a režimy. Při inicializaci aplikace také zajišťuje načtení uložených dat do datové struktury s panoramaty. Tato datová struktura se nachází ve scriptable objektu, jež je přístupný všem částem aplikace.

Samotná smyčka se dělí do tří režimů/módů, kterými jsou:

- Map mode. V tomto módu je na obrazovce zobrazena mapa. Smyčka v tomto režimu prochází veškerá panoramata a porovnává jejich vzdálenost od aktuální pozice zařízení. Pokud tato vzdálenost klesne pod vstupní hranici panoramatu, jež je nastavena na 10 metrů, zobrazí se uživateli dialog umožňující po potvrzení vstup do panoramatického režimu. Pokud si uživatel nepřeje vstoupit do panoramatického režimu,

aplikace si tuto volbu pamatuje až do odchodu ze zobrazovacího rádiusu daného panoramatu.

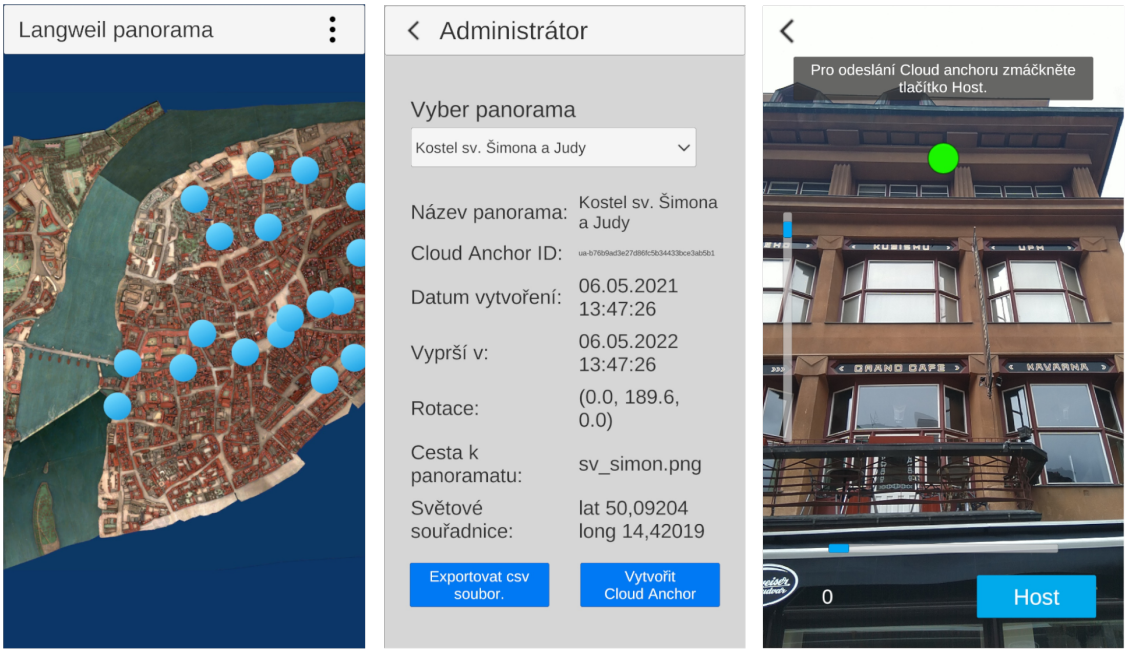
- Admin mode. Přepnout se do tohoto režimu lze v nabídce hlavního menu mapového režimu. V tomto režimu se zobrazí administrátorská obrazovka s údaji o panoramatu a pozastaví se mapový režim.
- AR mode. V tomto módu se aplikace přepne do jednoho z panoramatických režimů a pozastaví se mapové procesy.

Tato smyčka se také stará o komunikaci s uživatelem ve formě jednotlivých dialogů.

2.2.6 Kontrola podpory AR a jazyková lokalizace aplikace

Kontrola podpory AR knihoven se provádí po prvním spuštění čistě naisntalované aplikace a dochází při ní k vytvoření dočasného objektu ARSession, s jehož pomocí se provádí kontrola dostupnosti knihoven a jejich případná instalace. Pokud zařízení nepodporuje ani jednu z knihoven, jsou v aplikaci AR režimy vypnuty a používá se pouze Gyro režim.

Poslední implementovanou součástí aplikace je podpora více jazyků uživatelského rozhraní. Za tímto účelem se využívá balíčku Unity Localization package, který zpřístupňuje metody pro vícejazyčné texty uživatelského rozhraní. Pro každý text/zprávu se vytvoří záznam v tabulce, do kterého lze přidat překlady do dalších jazyků. Výsledný překlad se pak v aplikaci volí automaticky podle systémového nastavení.



Obrázek 2.5. Snímky uživatelského prostředí z implementované aplikace.

Kapitola 3
Závěr

3.1 Testování

Součástí testování mé aplikace bylo především porovnat kvality zarovnání panoramat v jednotlivých režimech a také schopnost rozpoznávání Cloud anchorů na různých zařízeních. Za tímto účelem jsme společně s vedoucím práce vybrali 21 lokalit v oblasti Staré Prahy, na kterých jsem poté vytvořil Cloud anchory pro pozdější rozpoznávání. Testování probíhalo na třech mobilních telefonech, konkrétně Honor 8, jež byl díky absenci podpory ARCore užíván pouze pro porovnávání s gyro režimem, dále pak Nokia 6.1, na němž probíhalo vytváření Cloud anchorů, a ASUS ZenFone AR A002.

3.1.1 Vytváření Cloud anchorů

Cloud anchors, jak je již popsáno v kapitolách výše, jsou primárně určeny pro vytváření a uchovávání rozšířené reality kolem nějakého konkrétního objektu či bodu. Tento objekt se pak nasnímá z různých úhlů a data se pošlou na Cloud anchor API.

V našem případě jsme potřebovali nasnímat okolí kolem nás, jež v některých místech bylo příliš vzdálené, nebo v případě některých budov příliš jednotvárné. Pro vytvoření Cloud anchoru na novém místě bylo tedy třeba nejdříve provést základní nasnímání kolem pomocného prefabu a poté rozhlédnutím kolem sebe nasnímat i okolí. Aby ale výsledná 3D mapa tohoto okolí byla dostatečně přesná a bylo možné docílit pozdějšího rozpoznání Cloud anchoru i z jiného úhlu, bylo také nutné naskenovat i jednotlivé budovy z různých úhlů.

Skrze různá místa bylo pak možné rozeznat výrazné rozdíly v kvalitě a rychlosti vytváření anchorů, jež se řídily těmito parametry:

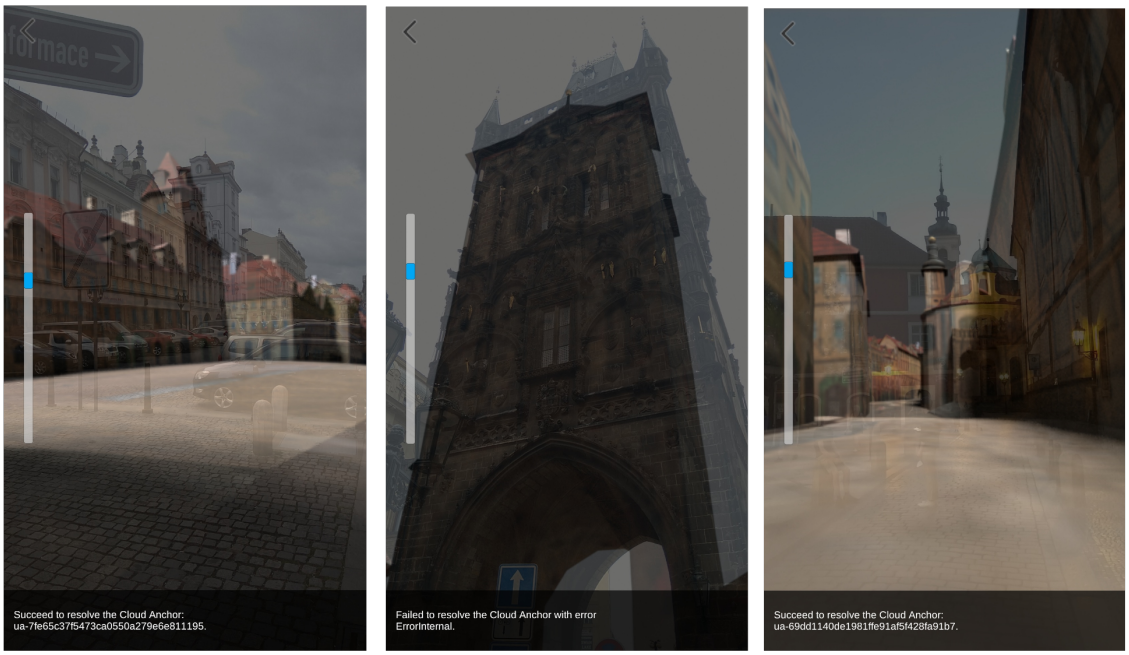
- **Vzdálenost budov a jiných objektů od místa vytvoření Cloud anchoru.** Nejznatelnější vliv tohoto faktoru byl znát při vytváření Cloud anchoru ve východní části Staroměstského náměstí. Zde byli všechny budovy vzdáleny více než 60 metrů a i po oskenování z více úhlů nebyla kvalita Cloud anchoru nijak uspokojivá. Naopak v uzavřených místech a ulicích bylo naskenování mnohem snazší a kvalitnější.
- **Členitost a rozmanitost povrchu skenovaných budov.** Zde byl rozdíl nejpatrnější v ulici 17. listopadu před Právnickou fakultou UK. Budovy zde byli dost jednotvárné a ploché, kvůli čemuž se nemělo vytváření Cloud anchoru podle čeho kvalitně orientovat, ani po opakovaném skenování z různých míst. Naproti tomu před budovou U Černé Matky Boží v Celetné ulici byla tato budova díky své členitosti kvalitně naskenována hned při prvním průchodu.
- **Výrazný bod zájmu pomáhající orientaci.** Jelikož skenovat celé 360° okolí panoramatu v dostatečné kvalitě je velice náročné, jsou velkou pomocí pro vytváření Cloud anchorů budovy či památky, které jsou dominantou daného místa. S využitím předpokladu, že tyto objekty budou jedním z prvních míst na které se uživatel podívá, lze tak kvalitním naskenováním právě tohoto objektu docílit rychlého nalezení Cloud anchoru a zarovnání podle něj.

3.1.2 Srovnání zarovnání gyro, AR-kompas a AR-Cloud anchor režimu

Stoprocentního zarovnání nebylo možné víceméně nikdy dosáhnout a to kvůli velkému počtu ovlivňujících faktorů, jež mají vliv na srovnání reálné scény s virtuální. Příkladem těchto faktorů může být odlišná pozice, ze které se panorama zobrazuje, oproti té, z níž se panorama renderovalo ve 3D modelu. V případě míst, kde se porovnávané budovy nachází blízko od pozorovatele tak vzniká i při malém posunu velký rozdíl v umístění obou scén oproti sobě. Taktéž ani sám Langweilův model neoplývá stoprocentní přesností, a tak je s těmito občasnými odchylkami třeba počítat.

Zarovnání v AR režimu pouze pomocí kompasu bylo závislé na přesnosti měření kompasu v daný okamžik, ale také na přesnosti renderování panoramat z 3D modelu, jež byli renderované tak, aby střed vyrenderovaného obrazu ukazoval na sever. Proto u některých panoramat, kde došlo při renderování k odchylce správné rotace, dochází při zarovnání kompasem k odchylkám i o 30° stupňů (takovým příkladem je panorama umístěné v ulici U milosrdných). Při ideálních podmínkách je ale zarovnání pomocí kompasu téměř shodné se zarovnáním pomocí Cloud anchorů. Konkrétní příklad lze vidět na prostředním obrázku 3.1.

U zarovnání pomocí Cloud anchorů bylo, podle očekávání, dosaženo největší přesnosti. Tato přesnost je ale ze všeho nejvíce ovlivněna pozicí pozorovatele, jelikož po nalezení Cloud anchoru se panorama umístí na přesně danou pozici, kde bylo umístěno administrátorem. Pokud se uživatel nachází ve vzdálenosti větší než 5 m, může už docházet k deformaci obrazu, jelikož se začne přibližovat k okraji sféry. Toto by ale mělo být možné vykompenzovat zvětšením rozměrů sféry s panoramatem (současný reálný rozměr je někde kolem poloměru 8 m). Příklad dosažených výsledků lze vidět na postranních obrázcích 3.1.



Obrázek 3.1. Ukázky zarovnání v AR režimech (kompas - uprostřed, Cloud anchor - po stranách).

V gyro režimu bylo užito také zarovnání pomocí kompasu. Tato implementace ale měla i přes přidanou korekci driftu místy problém s udržením původního zarovnání. Faktem znemožňujícím více objektivní porovnání oproti ostatním režimům se při testování stala chyba v implementaci kamery, jež způsobovala znatelné přiblížení v obrazu kamery a tedy i rozdílné poměry velikostí budov v panoramatu a na obrazu kamery.

3.1.3 Snímání rotace a rozpoznávání Cloud anchorů za různých světelných podmínek

Při tomto testování jsem zjišťoval, jak moc je ARCore ovlivňován při rozpoznávání své pozice v reálném prostředí okolním osvětlením. Změny v kvalitě v rozpoznávání svého okolí v podobě občasných záseků sféry během rotace jsem zaznamenal až za úplné tmy v ulicích s horším osvětlením. Ve všech těchto případech se ale stále jednalo o jednoznačně kvalitnější odhad rotace, než umožňovala aplikace využívající pouze gyroskop.

Dalším faktorem, na který jsem se zaměřil, bylo rozpoznávání Cloud anchorů závislé na okolním osvětlení. V tomto případě jsem již dosahoval horších výsledků. Za menšího šera (při jasné květnové obloze v cca 21:00) byla aplikace stále schopna vcelku bez problémů najít jednotlivé Cloud anchory. Po tomto času se ale schopnost rozpoznat další Cloud anchor s přibývající tmou rapidně zhoršila a za tmy již aplikace nedokázala rozpoznat ani Cloud anchor umístěný poblíž dobře osvětlené Staroměstské radnice.

Jako poslední faktor v této části, jež jsem měl zkoumat, byl vliv ročního období na rozpoznávání Cloud anchorů. Tuto část jsem ale bohužel nebyl schopen prakticky otestovat, jelikož v zimě, kdy by byl rozdíl nejznatelnější, aplikace ještě nebyla hotova. Proto můžu jen vyvozovat teoretické závěry a tedy, pokud se zaměřím na mnou testovaná panoramata, tak je pravděpodobné, že by k žádné velké odchylce vlivem jiného ročního období nemělo docházet, jelikož jako hlavní záchytné body pro rozpoznávání Cloud anchorů se užívají fasády okolních domů. Tyto fasády by neměli být nijak zvlášť ovlivňovány jak podzimním obdobím, kdy opadávají listy ze stromů, tak i sněhem v mrazivějších zimních dnech.

3.1.4 Výkonnostní a datové nároky AR režimů

Po výkonnostní stránce lze říci, že aplikace na obou mobilních telefonech podporujících ARCore běžela v uživatelském režimu plynule a bez jakýchkoliv problémů. U administrátorského režimu, při vytváření Cloud anchorů, již bylo viditelné menší zpoždění v obrazu oproti skutečnosti a místy i lehké zasekávání obrazu při skenování větších ploch. Toto chování bylo patrné u obou zařízení.

Pro změření datových toků mezi aplikací a internetem jsem využíval aplikaci Trafic Monitor [1], jež nabízí denní přehledy využití dat jednotlivými aplikacemi. Podrobnější analýzy těchto datových přenosů, jež by umožňovali rozlišit jednotlivá volání API metod jsem bohužel nenašel.

Datové přenosy lze v tomto případě rozdělit na dvě části, a to komunikaci se serverem EduARd, ze kterého bylo třeba stáhnout potřebná panoramata, kde se v případě stažení všech panoramat najednou jednalo o cca 120MB, a na komunikaci s ARCore Cloud anchor API při nahrávání a rozpoznávání anchorů.

V případě komunikace s ARCore Cloud anchor API se v uživatelském režimu jednalo průměrně o 4 MB na nalezení jednoho panoramatu. Při vytváření Cloud anchorů se jednalo přibližně o hodnoty kolem 10 MB na vytvoření jednoho Cloud anchoru.

[1] `https://play.google.com/store/apps/details?id=com.radioopt.widgethl=en`

3.2 Shrnutí

Naším cílem bylo vytvořit testovací aplikaci jež by za pomoci rozšířené reality a sdílení SLAM modelů umožňovala zobrazovat panoramata z Langweilova modelu Prahy. Tuto aplikaci pak otestovat ve srovnání s dříve používanými metodami a vyhodnotit její použitelnost.

Z testování nám vychází, že nástroje dostupných rozšířeně realitních knihoven v oblasti orientace zařízení více než uspokojivě nahrazují starší metody využívající gyroskop a kompas. Dále nám navíc umožňují provádět kvalitnější zarovnání panoramat, díky využívání Cloud anchorů, bez přílišných datových či hardwarových nároků na uživatelovo zařízení.

Během testování aplikace se objevilo také mnoho jejích nedostatků, na kterých by bylo třeba dále ještě zapracovat. Jako jeden z nejvýraznějších lze zmínit nepřesnost lokalizace pomocí GPS, která je již v takovéto městské zástavbě značně nepřesná a často byla při testování aplikace zdrojem problémů. Dále by stálo za to provést ještě přesnější warp Langweilovy mapy tak, aby především body s panoramaty byly v mapě umístěny na správných místech a více se zaměřit na způsob renderování panoramat tak, aby přesněji odpovídaly pohledům z reálného místa.

Pokud se v budoucnu objeví zájem o zpřístupnění aplikace širší veřejnosti, či další spolupráci s Muzeem Hlavního města Prahy, rád se budu věnovat dokončení aplikace tak, aby byla připravena pro běžné používání.

Literatura

[1] Nister, D. An efficient solution to the five-point relative pose problem. *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*. IEEE Comput. Soc, 2003, s. II-195-202. Dostupné na DOI 10.1186/s41074-017-0027-2. Dostupné na `http://ieeexplore.ieee.org/document/1211470/`.

[2] Taketomi, Takafumi, Hideaki Uchiyama a Sei Ikeda. Visual SLAM algorithms. *IPSJ Transactions on Computer Vision and Applications*. 2017, ročník 9, č. 1. ISSN 1882-6695. Dostupné na DOI 10.1186/s41074-017-0027-2. Dostupné na `http://ipsjcva.springeropen.com/articles/10.1186/s41074-017-0027-2`.

[3] *Inertial measurement unit*. Dostupné na `https://en.wikipedia.org/wiki/Inertial_measurement_unit`.

[4] *Fundamental concepts of ARcore*. Dostupné na `https://developers.google.com/ar/discover/concepts#motion_tracking`.

[5] *Unity, AR Foundation overview*. Dostupné na `https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html`.

[6] *ViroMedia, ViroReact overview*. Dostupné na `https://docs.viromedia.com/docs/viro-platform-overview`.

[7] Mur-Artal, Raul a Juan D. Tardos. Visual-Inertial Monocular SLAM With Map Reuse. *IEEE Robotics and Automation Letters*. 2017, ročník 2, č. 2, s. 796-803. ISSN 2377-3766. Dostupné na DOI 10.1109/LRA.2017.2653359. Dostupné na `http://ieeexplore.ieee.org/document/7817784/`.

[8] *ARCore Cloud anchors overview*. Dostupné na `https://developers.google.com/ar/develop/unity-arf/cloud-anchors/overview`.

[9] *Global localization with AR*. Dostupné na `https://ai.googleblog.com/2019/02/using-global-localization-to-improve.html`.

[10] *Sensor fusion*. Dostupné na `http://plaw.info/articles/sensorfusion/`.

[11] *Mapbox overview*. Dostupné na `https://docs.mapbox.com/help/getting-started/?size=n_10_n`.

[12] *Web Mercator projection*. Dostupné na `https://en.wikipedia.org/wiki/Web_Mercator_projection`.