ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

Využití realtime 3D rekonstrukce a lokalizace ve virtuální realitě

Realtime 3D Reconstruction and Localization in Virtual Reality

bakalářská práce

Studijní program: Otevřená informatika
Studijní obor: Počítačové hry a grafika

Vedoucí práce: Ing. David Sedláček, Ph.D.

Marek Římal

Praha 2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Římal** Jméno: **Marek** Osobní číslo: **465849**

Fakulta/ústav: **Fakulta elektrotechnická**

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

Studijní program: **Otevřená informatika**

Studijní obor: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Využití realtime 3D rekonstrukce a lokalizace ve virtuální realitě

Název bakalářské práce anglicky:

Realtime 3D Reconstruction and Localization in Virtual Reality

Pokyny pro vypracování:

Prostudujte možnosti využití hloubkových dat (výstup z hloubkové kamery) pořizovaných v reálném čase ve virtuální realitě (VR). Zjistěte možnosti využití těchto dat pro lokalizaci ve známém prostoru, navrhněte a proveďte testování pro nalezení limitů technologie (v různě velkých a složitých prostorách). Navrhněte virtuální aplikaci demonstrující využití těchto dat pro navigaci ve známém prostoru bez nutnosti vytváření umělých referenčních bodů (značek) s ohledem na omezení nalezená v předchozím kroku. Aplikaci implementujte a otestujte primárně z pohledu kvality sledování pozice uživatele v prostoru, stability virtálního světa, rychlostních limitů a dalších faktorů, které mají vliv na výslednou kvalitu virtuálního světa. Porovnejte výslednou aplikaci na dvou různých zařízeních: 1) náhlavní VR displej typu Windows Mixed Reality, 2) obecný VR náhlavní displej s připojenou hloubkovou kamerou ZED mini.

Seznam doporučené literatury:

[1] Moderní počítačová grafika: Bedřich Beneš, Jiří Sochor, Petr Felkel, Jiří Žára, 2005, Computer press
[2] The VR Book: Human-Centered Design for Virtual Reality: Jason Jerald, 2016, ACM Books
[3] Augmented Reality: Principles and Practice: Dieter Schmalstieg, Tobias Höllerer, 2016, Addison Wesley
[4] Practical Augmented Reality: Steve Aukstakalnis, 2017, Addison Wesley

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: **13.02.2019** Termín odevzdání bakalářské práce: **24.05.2019**

Platnost zadání bakalářské práce: **20.09.2020**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. Datum převzetí zadání

Podpis studenta

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.“

V Praze dne 21.5. 2019

Podpis autora práce ……

Poděkování

Děkuji panu Ing. Davidovi Sedláčkovi, Ph.D. za ochotu a pomoc při vedení mé bakalářské práce a za doporučení a propůjčení kvalitní literatury.

Abstrakt

Tato práce se zabývá problematikou lokalizace ve virtuální realitě s využitím hloubkových dat pořizovaných v reálném čase, a to zejména v kontextu sledování pozice uživatele, jeho navigace ve známém prostoru a stability virtuálního světa. V práci je využito modelu získaného pomocí 3D rekonstrukce zejména jako prevence uživatelovi kolize s fyzickým světem.

První část práce je věnována analýze hloubkových kamer a lokalizačních technik a principů. Na základě analýzy jsou představena konkrétní zařízení, která jsou dále v práci využívána. Těmito zařízeními jsou Acer Windows Mixed Reality headset a kamera ZED mini.

Součástí práce je také návrh a implementace aplikace v herním enginu Unity, která umožňuje vybraná zařízení testovat na základě jejich schopnosti vlastní lokalizace v prostoru. V závěru jsou tato zařízení zhodnocena na základě výsledků získaných při testování s lidmi.

Klíčová slova: lokalizace, 3D rekonstrukce, tracking, virtuální realita, hloubková kamera, ZED mini, Acer Windows Mixed Reality

Abstract

This work deals with the problem of realtime localization in virtual reality using depth data, especially in the context of monitoring a user's position, his navigation in the known space and stability of the virtual world. A model obtained by 3D reconstruction is used to prevent any collision between the user and the physical world.

The first part of the work analyzes depth cameras and localization techniques and principles. Based on the analysis is presented concrete equipment, which is further used in the work. That is an Acer Windows Mixed Reality headset and a camera ZED mini.

This work also concerns the design and implementation of the application in the Unity Game Engine. The application is used to test chosen devices based on their ability to localize in space. In conclusion, these devices are evaluated based on the results obtained with human testing.

Key words: localization, 3D reconstruction, tracking, virtual reality, depth camera, ZED mini, Acer Windows Mixed Reality,

Obsah

Seznam grafů

Seznam tabulek

Seznam obrázků

1 Úvod a motivace

Správná lokalizace uživatele v prostoru je jedním z klíčových měřítek kvality virtuální reality. Dnešní běžně prodávaná zařízení pro virtuální realitu jsou schopna poměrně kvalitní lokalizace uživatele na omezeném území. Tento prostor nemá více než několik metrů čtverečních a je geometricky jednoduchý. Například Windows Mixed Reality povoluje prostor o maximální velikosti zhruba 4 x 4 metry. HTC Vive tento prostor navyšuje zhruba o půl metru na obou osách.

Zajímá mě, zda je možné tento prostor rozšířit. Chci ověřit, jak si stejná zařízení povedou mimo oblast své spolehlivé zóny a otestovat, zda je možné je využít pro realtime lokalizaci uživatele i v rozsáhlejším prostoru, než který původně povolují. Pokusím se mezi dnešní technologií najít i jiná zařízení, která by se pro tento úkol mohla hodit.

Prostor, ve kterém se uživatel vybaven zařízením pro virtuální realitu může pohybovat, není omezen pouze rozsahem této oblasti, ale také neznámostí prostoru kolem. Aby měl uživatel jistotu, že nekoliduje s okolním fyzickým prostředím, je nutné mu vymezit prostor, ve kterém se lze bezpečně pohybovat. Bezpečný prostor je v případě současných zařízení pro virtuální realitu vymezen jednoduchým mnohoúhelníkem, který aproximuje hranici mezi bezpečným prostorem a fyzickými objekty. Uživatel tak kvůli aproximaci přichází o další prostor, ve kterém by se mohl pohybovat, neboť nemá jistotu, zda v místě mimo mnohoúhelník je fyzický objekt, či zda je tam volný prostor odseknutý aproximací.

Zajímá mě, zda by se prostor vymezující tuto bezpečnou oblast mohl nějakým šikovným způsobem zpřesnit. Chtěl bych vyzkoušet, jakým způsobem by se v tomto kontextu dala využít 3D rekonstrukce. Pomocí 3D rekonstrukce by podle mého šel jednoduchým způsobem získat model fyzického prostředí, který by mnohem lépe aproximoval prostor pro bezpečný pohyb uživatele než jednoduchý mnohoúhelník. Zaměřím se tedy i na charakteristiky 3D rekonstrukce za využití dnešní dostupné technologie. V souvislosti s tím, jak získaný model bude odpovídat fyzickému světu, mě dále zajímá, zda by takto vzniklý 3D model šel sám o sobě využít jako plnohodnotný model virtuálního prostředí.

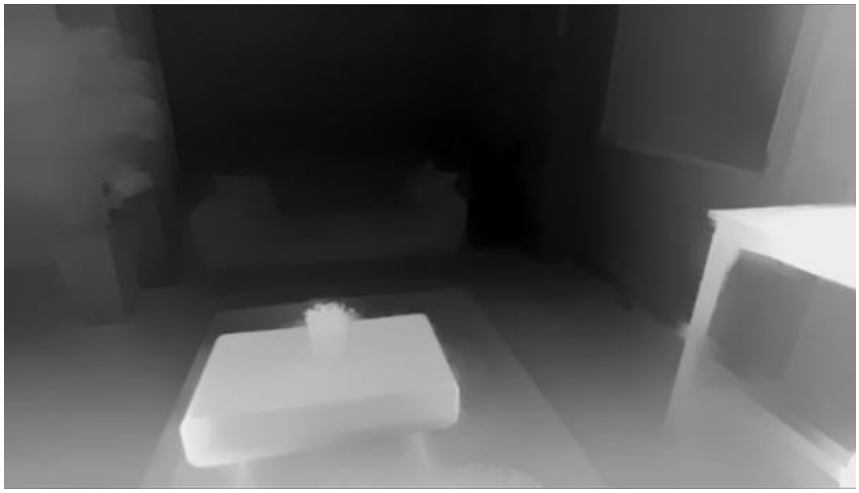
Cílem této práce je otestovat možnost využití dnešní technologie pro lokalizaci uživatele, a to zejména v kontextu sledování jeho pozice. Navrhnu demonstrační aplikaci, ve které schopnost lokalizace uživatele využiji k jeho navigaci ve známém prostoru. Uživatel bude plnit scénář ve virtuálním světě, jehož plněním bude navigován ve fyzickém světě. Pokud souřadné systémy virtuálního světa a reálného světa zůstanou synchronizovány, události vyvolané ve virtuálním světě budou mít svůj efekt i ve světě fyzickém. Při hodnocení jednotlivých zařízení se zaměřím zejména na schopnost uživatele tento scénář vykonat a na nepřesnost při jeho konání. Schopnost lokalizace uživatele v prostoru podpořím modelem získaným pomocí 3D rekonstrukce.

2 Teoretická část

2.1 Hloubkové kamery

Prvním krokem směrem k cíli mé práce je výběr toho správného zařízení pro sledování pozice uživatele v prostoru. Takové zařízení by mělo být schopné sledovat uživatelovu pozici i v rozsáhlejších a složitějších prostorách. Aby zařízení mohlo sledovat pozici uživatele po celou dobu jeho pohybu v takovémto prostoru, přijde mi nejrozumnější, aby se toto zařízení pohybovalo v prostoru společně s uživatelem. Zařízení tak bude sledovat svou vlastní polohu v prostoru, která bude totožná s polohou uživatele. Schopnost zařízení své vlastní lokalizace je založený na měření vzdáleností jednotlivých bodů ve scéně. Zařízeními, která v tomto ohledu vynikají, jsou hloubkové kamery. Právě ony budou tématem této první kapitoly.

Hloubkovou kamerou je taková kamera, která dokáže nějakým způsobem zjistit, jak jsou jí body ve scéně vzdáleny. Takováto kamera dokáže převést 3D prostor do 2D obrazu, kde hodnota každého pixelu odpovídá jeho vzdálenosti od kamery. Takový obraz lze vidět na obr. 1, kde vzdálenost bodu od kamery je znázorněna odstínem šedi. Na základě pozic pozorovaných bodů ve scéně je následně možné určit relativní pozici kamery vůči těmto bodům.



Obr. 1 Získáno ze zdroje [5]

Hloubkové kamery nyní rozdělím právě podle toho, jakým způsobem získávají hloubkovou informaci. Za každou skupinu vyberu konkrétního zástupce, který podle mého svými charakteristikami vhodně zastupuje danou skupinu. Charakteristiky, které mě budou zajímat, bude zejména počet FPS[1], neboť se tato práce zabývá lokalizací v reálném čase. Dále mě bude zajímat rozsah, ve kterém jsou kamery schopné měřit vzdálenost a jejich FOV[2]. Jednu z kamer budu využívat i pro 3D rekonstrukci scény, při které by podle mého měl uživatel vidět jak generovanou 3D síť[3], tak reálný svět okolo něj. Kameru tedy chci využít ve spojení s headsetem i jako náhlavní *video see-through displej*[4], a proto se zaměřím se i na rozlišení kamery. Je nutné zmínit, že existuje vztah mezi FPS a rozlišením. Obecně by šlo říci, že při snížení rozlišení lze

[1] *Frames Per Second* – česky snímky za sekundu.
[2] *Field Of View* – čeky zorné pole. Udává se ve stupních na vertikální a horizontální ose kamery
[3] Anglicky 3D mesh
[4] *Video see-through displej* je označení pro displej propojený s kamerou. Kamera poskytuje živý videozáznam fyzického světa, který se následně mixuje s virtuálními prvky. Tento mix se následně zobrazí uživateli, který tam vidí fyzickou scénu obohacenou o virtuální prvky.

zvýšit FPS, proto se pokusím u jednotlivých kamer uvádět rozlišení při podobném počtu FPS. Na závěr kapitoly na základě vyjmenovaných charakteristik rozhodnu, které kamery jsou pro mé účely nejvhodnější.

2.1.1 Výpočet hloubky

V této kapitole se zaměřím na tři základní principy pro měření vzdálenosti pomocí optických sensorů. Rád bych podotkl, že některé přístupy se mohou vzájemně kombinovat.

Ještě před tím, než přejdu k jednotlivým principům, vysvětlím pojem triangulace. **Triangulace** je způsob, jak lze zjistit souřadnice třetího vrcholu v trojúhelníku. Pokud známe vzdálenost mezi dvěma body v trojúhelníku a libovolné dva úhly, jsme schopni vypočítat souřadnice třetího bodu pomocí trigonometrických vztahů.

První skupinou jsou tzv. **time-of-flight** kamery. Kamery založené na principu time-of-flight se skládají ze dvou částí. První částí je světelný zdroj, který vysílá světelné paprsky. Druhou částí je kamera, která tyto paprsky zachytává. Přímá ToF kamera měří dobu mezi vysláním paprsku světelným zdrojem a jeho opětovnému zachycení kamerou. Na základě této doby je schopna vypočítat, jak je vzdálený bod, od kterého se světelný paprsek odrazil. Nepřímá ToF kamera vzdálenost určuje na základě rozdílu fáze vysílaného a přijímaného světla, tedy elektromagnetického vlnění. [15] V praxi se většinou využívá infračerveného světla, které je pro lidské oko neviditelné.

Jako zástupce pro ToF kamery jsem si vybral produkt Basler Time-of-Flight Camera. Udáje jsem získal ze zdroje [6].

FPS: 20
Rozsah: 0 – 13 m
FOV: 57° horizontálně a 43°vertikálně
Rozlišení: 640 x 480 px

Další skupinou jsou **structured light** kamery, které se také skládají ze dvou základních součástí, z kamery a projektoru. Projektor promítá speciálně navržený 2D světelný vzor na objekty ve scéně. Kamera snímá scénu osvětlenou projektorem. Pokud by scéna byla plochá, tedy bez žádného 3D zakřivení, pak by byl promítaný světelný vzor na snímku pořízeném kamerou velmi podobný vzoru vyslanému. V případě prostorové 3D scény je tento vzor deformován 3D geometrií. Právě na základě rozdílu mezi vyslaným vzorem a snímaným vzorem se vygeneruje hloubková mapa. [16] Světlo se opět často vysílá infračervené, aby bylo mimo lidské viditelné spektrum.

Jako zástupce pro structured light kamery jsem si vybral produkt Kinect. Údaje o tomto zařízení jsem získal ze zdroje [7].

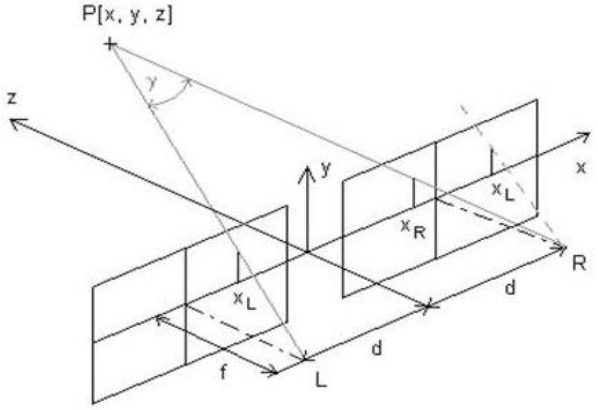
FPS: 30
Rozsah: 0,5 – 5 m
FOV: 56° horizontálně a 46°vertikálně
Rozlišení: 650 x 460 px

Poslední skupinou jsou kamery založené na principu **stereovidění**. Při psaní této kapitoly jsem čerpal z [8]. Stereovidění je vlastně způsob, jakým vidí člověk. Při pohledu na objekt vidí levé a pravé oko objekt z lehce jiného úhlu. Každé oko tak vytváří na sítnici jiný obraz. To nám umožňuje vidět objekt prostorově (stereoskopicky). V případě stereo kamer se místo očí využívá dvou kamer, jejichž vzdálenost je známá. Obecně platí, že čím větší je vzdálenost mezi kamerami, tím větší je přesnost v měření hloubky. V některých případech se tyto kamery využívají jak k měření hloubky, tak jako náhlavní displej pro

rozšířenou realitu. U takových kamer se tato vzdálenost pohybuje kolem 65 mm, což je průměrná pupilární distance u člověka.

Kamera tedy získá dva rozdílné stereoskopické snímky. Každý snímek obsahuje scénu z jiného úhlu. Důležitá je tzv. **úhlová paralaxa** (úhel γ na obr. 2*)*. To je úhel, který svírají úsečky spojující pozorovaný bod a ohniska obou kamer (bod P je pozorovaný bod a body L a R jsou ohniska kamer). Je zřejmé, že čím blíže je pozorovaný objekt ke kameře, tím je úhlová paralaxa větší.

Optické osy kamer uvažujme rovnoběžné s osou *z* souřadnicového systému kamery a jejich obrazové roviny leží v rovině $z = 0$. Stereovidění lze použít i při nedodržení těchto podmínek převodem na tento nejjednodušší tvar. To je však mimo rozsah této práce.



Obr. 2 Získáno z [8]

Na obr.2 značí *2d* vzdálenost mezi optickými osami kamer, *f* je jejich ohnisková vzdálenost, x_L a x_R jsou souřadnice bodu *P* v levém a pravém snímku.

Pro určení souřadnic bodu *P* je nutné tento bod identifikovat v obou snímcích. Jeho souřadnice *x, y, z* získáme ze vztahů:

$$x = x_L \frac{2d}{x_L - x_p} \qquad y = y_L \frac{2d}{x_L - x_p} \qquad z = \frac{2\,df}{x_L - x_p} - f$$

Problém nalezení korespondence bodu *P* v pravém i levém snímku je zjednodušen tím, že bod nalezený na snímku z jedné kamery bude na snímku z druhé kamery ležet na průmětu spojnice ohniska a pozorovaného bodu do obrazové roviny druhé kamery. Při hledání korespondencí se tedy lze omezit pouze na množinu bodů ležících na průmětu této spojnice.

Jako zástupce pro kameru využívající stereovidění jsem si vybral produkt ZED mini. Údaje jsem získal ze zdroje [5].

FPS: 15
Rozsah: 0,5 – 12 m (lze zvýšit nastavením zařízení)
FOV: 90° horizontálně a 60°vertikálně
Rozlišení: 4416 x 1242 px

2.1.2 Výběr pro účely této práce

Hlavním aspektem při výběru kamery pro mě byl poměr počtu FPS a rozlišení. V tomto ohledu si nejlépe vedou kamery založené na principu stereovidění. Tyto kamery produkují dostatečný počet snímků za vteřinu, což je umožnuje využívat v reálném čase. Díky vysokému rozlišení a FOV je lze používat i jako náhlavní display pro rozšířenou realitu. Kamery založené na stereovidění si vedou nejlépe i co se týče rozsahu, ve kterém jsou schopné měřit vzdálenost.

Další velkou výhodou kamer založených na stereovidění je ten, že nejsou tolik závislé na světle jako tomu je u ToF a structured light kamer. Stereo kamery tak lze využívat jak vevnitř, tak venku [5]. Tento aspekt také podporuje mou myšlenku lokalizace na větším a různorodějším prostoru.

Na základě charakteristik výše uvedených kamer mi přijdou kamery založené na principu stereovidění nejvhodnější pro můj účel. Pro další postup jsem tedy zvolil stereokamery.

2.2 Lokalizace

V předešlé kapitole jsem vysvětlil, co je to hloubková kamera. Popsal jsem různé principy, na jakých hloubkové kamery fungují, a jaké jsou jejich charakteristiky. Na základě toho jsem se rozhodl pro další postup využít kamery založené na principu stereovidění. Kapitola tedy bude směřovat k metodám lokalizace za využití právě zmiňovaných stereokamer.

Kapitolu začnu vysvětlením základních pojmů souvisejících s lokalizací. Pomocí těchto pojmů následně vysvětlím pojem lokalizace v kontextu této práce a vysvětlím, proč jsou tyto pojmy důležité. Stěžejní částí této kapitoly bude lokalizace pomocí optických sensorů, kterými obecně hloubkové kamery disponují.

Tracking je termín používaný k popisu dynamického chování systému vzhledem ke svému okolí. [1] Tracking vlastně znamená identifikace polohy zařízení v průběhu času. [14] Abych mohl správně umístit virtuální objekty na správné místo ve fyzickém světě, je nutné znát alespoň relativní pozici a orientaci sledovaného zařízení vzhledem k fyzickým objektům. Pro účely mé aplikace, která běží v reálném čase, musí být výpočet jeho polohy, tedy pozice a orientace, neustále aktualizován. Pro zjednodušení čtení budu termín v rámci této práce skloňovat podle pravidel českého pravopisu.

Kalibrace je proces srovnávání hodnot naměřených na dvou různých zařízeních. Jedno zařízení je referenční a druhé se má tzv. zkalibrovat. Zařízení referenční lze nahradit známými referenčními hodnotami nebo v kontextu geometrického měření známým souřadným systémem. Cílem kalibrace je určit parametry pro správné fungování kalibrovaného zařízení. Zatímco tracking znamená provádět výpočty průběžně v čase, kalibrace se většinou provádí jen jednou za čas. [1] Referenční zařízení může zastupovat člověk.

Lokalizace je jednoduše určení polohy objektu v prostoru. V kontextu této práce je objektem zařízení, které má uživatel na hlavě, čímž je RGB stereo kamera. Lokalizací zařízení tak lokalizujeme i uživatelovu hlavu. Pro správné lokalizování zařízení je nutné, aby jeho souřadný systém odpovídal skutečnému souřadnému systému fyzického světa. Zarovnání těchto dvou souřadných systému se říká **registrace**. Statická registrace znamená zarovnání těchto souřadných systému v momentě, kdy se zařízení nehýbe. To se provede kalibrací tohoto zařízení. Dynamická registrace je průběžné zarovnávání těchto souřadných systému během pohybu zařízení. To se provádí pomocí trackingu. [1]

Správnost registrace bude klíčovým měřítkem schopnosti lokalizace vybraných zařízení, kterou budu hodnotit pomocí mé demonstrační aplikace. Abych mohl uživatele správně navigovat ve fyzickém světě čistě podle objektů ve virtuálním světě, je nutné, aby tyto světy měly vzájemně registrované souřadnice. V rámci této práce mě bude zajímat zejména realtime lokalizace, proto je největší otázkou to, jak přesný bude tracking vybraných zařízení, neboť ten je zodpovědný za dynamickou registraci. Tracking tedy bude stěžejním tématem celé této kapitoly.

IMU

Vývoji IMU jednotek z velkých mechanických systému k levným mikroskopickým zařízením bylo jedním z nejdůležitějších kroků k vysoce přesnému trackingu. Touto jednotkou je vybavena většina zařízení podporujících tracking. IMU je elektronické zařízení obsahující akcelerometr, gyroskop a často i magnetometr. Jedná se o zařízení, které je velice rychlé a přesné. Je vestavěno do zařízení a poskytuje tak měření ihned.

IMU se typicky skládá ze tří základních součástí. **Gyroskop** je zařízení pro výpočet úhlové rychlosti, **akcelerometr** měří zrychlení a **magnetometr** měří magnetickou sílu. Každé z těchto zařízení poskytuje hodnoty ve třech ortogonálních osách. [4]

Ačkoli je IMU jednotka schopná vysoce přesného sledování orientace zařízení ve 3D prostoru. Při sledování pozice si nevede příliš dobře. Samotné IMU ani nerozpozná konstantní pohyb od žádného, neboť akcelometr měří pouze zrychlení. Pro 6-DOF tracking je tedy nutné IMU jednotku podpořit trackingem pomocí jiného sensoru. IMU však stále zůstává velmi důležitou součástí 6-DOF trackingu.[1]

IMU lze podpořit trackingem pomocí visuálních sensorů. Tracking tedy podpořím daty získaných pomocí z hloubkových kamer. Právě tracking pomocí visuálních sensorů (optický tracking) bude tématem zbytku této kapitoly.

2.2.1 Optický tracking

Optický tracking je založen pouze na datech získaných z optických sensorů. Dnes je dnes jedním z nejdůležitějších tracking principů pro virtuální a rozšířenou realitu. [1] Dokonce i levné malé kamery poskytují velmi bohatá data. Produkují miliony nezávislých pixelů pořízených najednou, které lze analyzovat pomocí technik počítačového vidění.

Význačné body

Základním principem optického trackingu je rozpoznávání význačných bodů v okolí. Význačné body se vyznačují tím, že jsou na snímcích pořizovaných kamerou jednoduše rozpoznatelné a rozlišitelné od ostatních bodů ve svém okolí, viz obr. 3. Takovéto body jsou význačné zejména pro kameru. Pro člověka většinou význam nemají. Často se jedná například o rohy objektů nebo změny v texturách. Dále se od význačných bodů také očekává vlastnost jejich opakovatelného rozpoznání z více různých úhlů. [1]

Obr. 3 Detekce význačných bodů na snímku. Obrázek získán ze zdroje [17]

Význačným bodům se přiřazují tzv. deskriptory. To jsou datové struktury popisující význačný bod jako vektor. Jsou invariantní k rotacím a změně měřítka. Visuálně podobné význačné body mají podobný deskriptor. Tím myslím to, že euklidovská vzdálenost těchto dvou vektorů bude blízká nule. Deskriptory jsou důležité při vyhledávání stejných význačných bodů na různých snímcích. Snímky se profiltrují filtrem, který pro daný snímek vrátí seznam význačných bodů. Tyto body se pak na základě jejich deskriptorů vzájemně párují. [13]

Význačné body lze uměle vyrobit a do scény přidat člověkem. V tomto případě mluvíme o uměle vyrobených značkách. Jedná se většinou o vysoce kontrastní nálepky se vzorem, který jednoznačně určuje jejich rotaci. Umělé značky mohou zvýšit efektivitu optického trackingu zejména v prostorech, kde se

vyskytují rozsáhlé jednolité plochy nebo mnoho blyštivého materiálu. [1] Rozlišujeme tedy tracking s uměle vyrobených značek a tracking přirozených bodů.

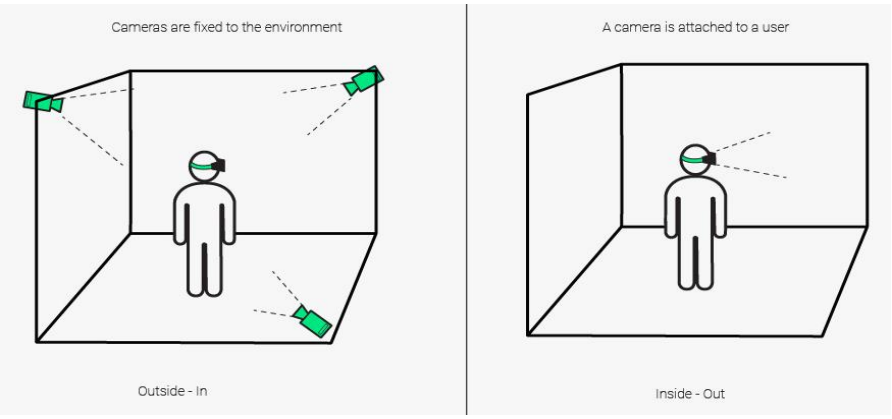
Tracking přirozených značek vyžaduje lepší kvalitu obrazu a více výpočetního výkonu. Proto tento typ optického tracking nabyl na popularitě teprve nedávno. [1]

Detekce těch správných význačných bodů je hned vedle přesného určení jejich prostorové pozice z obrazu ta největší výzva. Čím více význačných bodů kamera detekuje, tím více objektů bude schopna identifikovat. Vysoký počet význačných bodů však zvyšuje riziko jejich vzájemné záměny. Rozlišení obrazu v tomto případě hraje velkou roli.

Pro účely mé práce nebudu umělé značky využívat. To zejména z důvodu praktického a pohodlného využití. Zaměřím se tedy pouze na detekci značek přírodních.

Inside-out a Outside-in tracking

Systémy využívající optický tracking lze dělit do dvou kategorií podle toho, kde má sledované zařízení umístěné sensory. Pokud má zařízení sensory umístěné na pevno v prostoru, mluvíme o tzv. **outside-in** trackingu (obr. 4 vlevo). Sensory tak sledují body na zařízení a na základě svého rozmístění v prostoru a polohy pozorovaných bodů provádí výpočet polohy zařízení. Druhému případu, tedy kdy má zařízení sensory umístěné na sobě samém, se říká **inside-out** tracking (obr. 4 vpravo). Sensory pozorují význačné body umístěné v prostoru kolem. [1] Tyto význačné body mohou být dopředu do prostoru umístěné člověkem, však dnes je možné provádět tracking i v předem nijak nepřipraveném prostředí, viz minulá podkapitola.



Obr. 4 Získáno ze zdroje [18]

Inside-out tracking umožnuje zařízení větší volnost pohybu, neboť prostor není ohraničený sensory. Na druhou stranu je zařízení zatíženo tíhou sensorů nebo jejich energetickou spotřebou. V mé práci budu využívat inside-out tracking. To zejména kvůli prostorově omezujícímu charakteru outside-in trackingu.

Osvětlení

Na kvalitu a přesnost optického trackingu má velký vliv světlo. Obecně se dá říct, že při nedostatku světla optický tracking selhává, protože snímky jsou málo kontrastní, a tak se hůře detekují objekty. Jsou dvě možnosti, jak přistupovat k osvětlení ve scéně. **Pasivní osvětlování** se spoléhá pouze na přirozené osvětlení, které je již ve scéně přítomno. Tím se rozumí jak přírodní světelné zdroje, tak i člověkem vyrobené osvětlení, jako třeba domácí osvětlení. Oproti tomu **Aktivní osvětlování** není závislé na přirozeném osvětlení, neboť využívá vlastní zdroj světla. Viditelné světlo by měnilo vzhled okolí, a tak se

využívá světlo infračervené. Kamera s infračerveným filtrem zaznamená pouze infračervené světlo a získá tak obraz s vysokým kontrastem. [1] Ve této práci budu využívat pouze pasivního osvětlení. Kamery využívající stereovidění nejsou tolik náchylné na světelné podmínky.

Model-based a Model-free tracking

U optického trackingu rozlišujeme, zda máme možnost porovnávat pořizované snímky kamery s dopředu známým referenčním modelem, jakým je například model okolí. V takovém případě mluvíme o tzv. **model-based** trackingu. V případě druhém, kdy nám takový model není dopředu známý, mluvíme o tzv. **model-free** trackingu. Bez známého referenčního modelu kamera určuje svou polohu pouze relativně ke své startovní pozici. [1]

Tracking detekcí

Jedná se o model-based tracking, který spočívá v tom, že si kamera dopředu naskenuje okolí a takto vzniklý digitální model pak za běhu zarovnává s tím, co právě snímá.

Pozice kamery je vyhodnocována pro každý snímek zvlášť. To znamená, že pro logiku tohoto trackingu je pozice kamery v jednom snímku nezávislá na pozici kamery ve snímku předcházejícím. Nesprávné určení polohy v jednom snímku tedy nemá žádný vliv na výpočet polohy ve snímku dalším. Tento přístup je jednodušší na implementaci, neboť není třeba udržovat informaci o historii.

V každém snímku se nejprve detekují potencionálně význačné body. Ke každému takovému bodu se vytvoří deskriptor, který se dále páruje s deskriptory na předem naskenovaném referenčním modelu, čímž vznikají 2D - 3D korespondence. Na těchto datech je následně spuštěn robustní algoritmus pro určení pozice.

Inkrementální tracking

Tracking detekcí se dá využít v jednodušších případech, kde se páruje pouze menší počet význačných bodů, nejlépe umělých značek. Na tracking při velkém počtu přirozených značek tento přístup nestačí. U trackingu zařízení pro virtuální realitu lze předpokládat, že jejich poloha se mezi dvěma po sobě jdoucími snímky značně nezmění. To je informace, kterou tracking detekcí opomíjí. Provádí tak celý výpočet pozice pro každý snímek úplně od znova, a tím dělá celý problém mnohem těžším. Na rozdíl od trackingu detekcí, bere tento fakt v potaz inkrementální tracking.

Informace z minulého snímku výrazně usnadní výpočet polohy kamery. V každém novém snímku víme, kde zhruba se kamera bude nacházet. Hledání význačných bodů a jejich následné párování provádíme pouze v okolí jejich předešlé pozice. Přesnost predikce polohy kamery se dá dále zlepšovat například tím, že budeme předpokládat její souvislý pohyb. Na základě parametrů jako je rychlost kamery nebo její zrychlení, které lze získat z předešlých snímků nebo gyroskopu, lze poloha kamery predikovat s ještě větší přesností.

Pro spuštění inkrementálního trackingu je nutné znát přesnou pozici kamery vůči předem naskenovanému modelu. Tento problém se dá řešit tak, že při spuštění trackingu se nebude spoléhat na předem naskenovaný model, ale na model získaný z video záznamu kamery. Ve chvíli, kdy je model rozpoznán a pozice kamery určena, přejde se k inkrementálnímu trackingu. V případě ztráty modelu se vrací zpět do detekčního modu, kdy model extrahujeme z videozáznamu kamery.

Informace o trackingu detekcí a inkrementálním trackingu jsem získal ze zdroje [1].

3D rekonstrukce a mapování

Než přejdu k problematice souběžné lokalizace a mapování, vysvětlím, jak funguje 3D rekonstrukce, jež je důležitou součástí mé práce. 3D rekonstrukce v kontextu této práce je proces vytváření 3D modelu z dat získaných pomocí optických sensorů. Tento proces lze rozdělit do tří stěžejních kroků.

Prvním krokem je extrahování kolekce 3D bodů pozorovaných z jedné fixované pozice. V každém jednotlivém pořízeném snímku je potřeba identifikovat význačné body, viz výše. Následně pomocí principu stereovidění vypočítat jejich souřadnice. Této kolekci bodů se často říká *point cloud*. V druhém kroku se hledají shodné body na snímcích pořízených z různých pozic. To znamená, že se vzniklé kolekce bodů slučují. Krokem třetím je na základě sloučené kolekce vygenerovat geometrický model. Pokud je hustota bodů příliš vysoká, je nutné nějaké body odstranit. Stejně tak se musí odstranit body vychýlené od jejich opravdové pozice. Po tomto čištění se na základě pozůstalých bodů vytvoří plochy, ze kterých pak vzniká finální 3D síť. [4]

Simultaneous Localization and Mapping (SLAM)

Do teď jsem psal o způsobech model-based trackingu. Tyto způsoby předpokládaly přítomnost známého referenčního modelu před spuštěním trackingu. SLAM je označení pro problém současné lokalizace a mapování. Nejjednodušší formou model-free trackingu a základem pro visuální SLAM (V-SLAM) je visuální odometrie. [1] Stejně jako odometr v autě, který odhaduje počet najetých kilometrů na základě otáček kol, visuální odometrie odhaduje polohu kamery na základě změny mezi snímky pořízenými kamerou. Při trackingu je vytvářen 3D model okolí, který je využit k podpoře inkrementálního trackingu.

Tento přístup je sám o sobě velmi naivní, každá další pozice je určena na základě výpočtu pozice předešlé. To znamená, že se nepřesnost vznikající mezi dvěma snímky bude v průběhu času akumulovat. Následkem toho bude odklon odhadované trajektorie zařízení od její skutečné trajektorie. Tento problém se řeší pomocí techniky **bundle adjustment**, kdy se optimalizují pozice kamery vzhledem k doposud pozorovaným bodům. Problém lze převést na úlohu nejmenších čtverců, přičemž se hledají pozice kamery minimalizující rozdíly projekcí 3D bodů do snímků pořízených kamerou. Jelikož by složitost problému rostla s počtem snímků, **windowed bundle adjustment** uvažuje při optimalizaci pouze několik posledních. Pro další zefektivnění výpočtu se uvažují pouze body ve sledovaném regionu. [1]

Dalším problémem, se kterým se SLAM potýká, je dočasná ztráta trackingu. Ta může nastat například dočasným zastíněním kamery nebo jejím prudkým pohybem. Tracking se pak musí restartovat na stejných souřadnicích. Tomuto problému se říká **relokalizace**. Populární technikou pro řešení relokalizace je pro každý snímek vytvořit deskriptor, který vznikne snižováním jeho rozlišení. Kamera po ztrátě trackingu porovnává tyto deskriptory s aktuálním snímkem kamery. Při detekci dostatečně podobného snímku, kamera začne hledat známe význačné body v aktuálním snímku a restartuje tracking. [1]

Jedním z cílů SLAM je vytvořit konsistentní odhad trajektorie zařízení v prostoru. To vyžaduje udržování mapy okolí, aby se dalo poznat, kdy se zařízení vrátilo na již dříve navštívené místo. Navrácení zařízení do již navštíveného místa se říká **loop closure** (uzavření kruhu). Tuto informaci je možné využít pro redukci odchylky, jak ve tvořené mapě okolí, tak v odhadované trajektorii zařízení. Detekce toho, kdy loop closure nastane a efektivní integrování této nové podmínky do aktuální mapy okolí je jedním z hlavních problémů SLAM algoritmů. [12]

Od trackingu se vyžaduje operace v reálném čase. Pro mapování je tento požadavek příliš přísný. **Paralelní tracking a mapování** (PTAM) je přístup, při kterém běží tracking a mapování na samostatných vláknech.

To dovoluje operaci trackingu v reálném čase, zatímco mapovací vlákno může běžet mnohem pomaleji. To aktualizuje mapu vždy po až několika snímcích pořízených kamerou. [1]

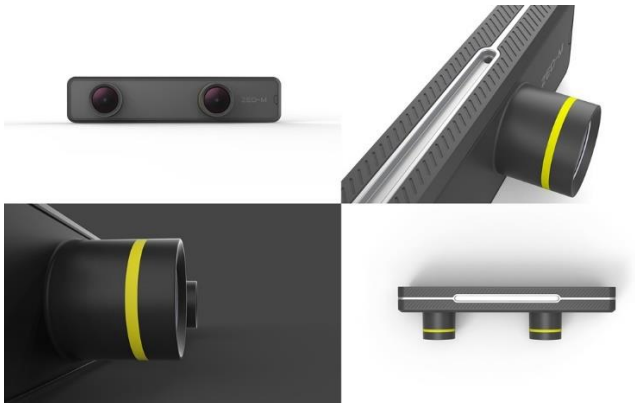
Algoritmy pro SLAM předpokládají, že se kamera pohybuje v neměnném prostředí. Proto jsou tyto algoritmy nepoužitelné například pro snímání lidských postav nebo ručních gest.

2.3 Vybraná technologie

V průběhu minulých kapitol jsem zúžil svůj výběr zařízení na stereo RGB kamery. Ty budu využívat k inside-out trackingu založeném rozpoznávání přirozených bodů v prostoru. V této kapitole představím konkrétní zařízení, která využiji pro realizaci mé demonstrační aplikace. Zařízení popíši a analyzuji možnosti, které mi tato zařízení poskytují.

2.3.1 ZED mini

Posledním výrobkem firmy Stereolabs je stereo kamera ZED mini (obr. 5) vybavená dvěma RGB kamerami. ZED mini je oproti svému předchůdci, kameře ZED, menší a dá se tak snadno připevnit na headset pro virtuální realitu. Snaží se napodobit způsob, jakým vidí člověk. RGB kamery jsou od sebe vzdálené 63 mm, což je velmi podobné, jako tomu je u člověka. Kameru je tak vhodné využít jako video see-through displej.



Obr. 5 Kamera ZED mini. Získáno ze zdroje [5]

Kamery dokáže měřit vzdálenost v rozsahu od 0,1 metru až do 12 metrů. Při přepnutí do *Ultra-Depth* módu se dá vzdálenost, na kterou dokáže kamera měřit hloubku, zvýšit na 20 metrů. Nejnovější SDK update tuto hranici prý posouvá až na 40 metrů. [5]

Zorné pole kamery (FOV) je 90° horizontálně a 60° vertikálně. Kamera je schopná pořizovat videozáznam v rozlišení 2.2K, však pro propojení s headsetem pro virtuální realitu je rozlišení třeba snížit na 720p (při zachování 60FPS), viz obr. 6.

Video Mode	Output Resolution (side by side)	Frame Rate (fps)	Field of View
2.2K	4416x1242	15	Wide
1080p	3840x1080	30, 15	Wide
720p	2560x720	60, 30, 15	Extra Wide
WVGA	1344x376	100, 60, 30, 15	Extra Wide

Obr. 6 Získáno ze zdroje [5]

ZED mini má vbudovaný gyroskop a akcelometr. Díky tomu a vlastnímu SLAM algoritmu není kamera závislá na jiném zařízení a dokáže se tak lokalizovat sama. Využít jí tak lze například i na headsetu, který nepodporuje tracking.

Všechny údaje o této kameře byly získány z [5].

2.3.2 Možnosti a vlastnosti kamery ZED mini

Kamera ZED mini disponuje třemi hlavními funkčnostmi, kterými jsou *Positional Tracking*, *Depth Sensing* a *3D Reconstruction.*

Informace o těchto funkčnostech jsem čerpal z [5].

Positional Tracking

Lokalizace kamery ZED mini je založená na principu SLAM, přičemž využívá pouze přirozeného světla. Disponuje šesti stupni volnosti (6DoF) a svou polohu aktualizuje až stokrát za vteřinu. Poloha zařízení je počítána pro levou kameru a je relativní vzhledem k referenčním souřadnicím (*reference frame*). Těmi můžou být buď světové souřadnice nebo souřadnice kamery.

Světové souřadnice popisují kameru ve fyzickém světě kolem. To vyžaduje referenční bod, který zůstává pevný, zatímco se zařízení pohybuje. Referenčním bodem je počátek souřadnicového systému. Pokud není specifikováno jinak, je umístěn na místě, kde kamera spustila tracking. Souřadný systém kamery má počátek umístěném za levou kamerou zařízení. Pohybuje se tedy společně s kamerou.

Během trackingu si kamera vytváří mapu význačných bodů. Po jeho ukončení si tuto mapu ukládá do tzv. areafiles, což jsou zakódované soubory s příponou area. Při spuštění trackingu lze specifikovat, jaký tento soubor má být pro tracking použit.

Depth Sensing

ZED mini díky svým dvou kamerám dokáže generovat hloubkovou mapu. Pro každý pixel se souřadnicemi (x, y) ukládá jeho z souřadnici v souřadnicovém systému kamery.

Hloubka pixelu je reprezentována jako 32bitové číslo. Zobrazit se dá 8bitová reprezentace ve stupni šedi, kde každý pixel má hodnotu mezi 0 a 255. Hodnota 255 představuje nejbližší možný pixel a 0 ten nejvíce vzdálený.

3D Rekonstrukce

3D Rekonstrukce je schopnost zařízení vytvářet *3D síť* prostředí. Mapa je aktualizována pohybem kamery a snímáním nových objektů kolem.

3D síť je ukládána ve světových souřadnicích. Po ukončení 3D rekonstrukce je možné 3D síť uložit do souboru[5], stejně tak i načíst. 3D síť dříve uložená pomocí 3D rekonstrukce se zobrazí na stejné místo ve světových souřadnicích, na kterých byla uložena.

Výsledná 3D síť je složena z trojúhelníkových ploch sestávajících se ze třech vrcholů, přičemž každý vrchol má svou normálu. Rozlišení (*resolution*) sítě a maximální vzdálenost (*range*) do které se 3D síť generuje se dají nastavit při inicializaci. Finální 3D síť lze filtrovat pomocí filtru (*mesh filtering*) a redukovat tak počet trojúhelníků, ze kterých je 3D síť složena. Dále je možné snímat i barvu prostředí a přidat tak na výslednou 3D síť odpovídající texturu.

[5] ZED plugin podporuje ukládání 3D sítě ve formátu wavefront (.obj), formátu polygon file format (.ply) a binárním formátu (.bin).

2.3.3 Náhlavní display Windows Mixed Reality

Acer Windows Mixed Reality (obr. 7) headset je náhlavní displej pro virtuální realitu. Taktéž využívá optický inside-out tracking společně s jednotkou IMU. Stejně jako kamera ZED mini rozlišuje šest stupňů volnosti (6DoF). Je vybavený dvěma černobílými kamerami pro rozpoznávání význačných bodů v okolí. Přístup k osvětlení je pasivní, spoléhá se tedy pouze na přirozené viditelné světlo. Význačné body v prostoru a informace s nimi spojené se ukládají na disk počítače. Při zapojení headsetu do počítače headset rozpozná místnost, ve které již byl použit.

Obr. 7 Acer Mixed Reality headset. Získáno ze zdroje [19]

2.3.4 Možnosti a vlastnosti náhlavního displeje Windows Mixed Reality

Headset Acer je běžný náhlavní displej pro virtuální realitu. Na rozdíl od hloubkové kamery ZED mini kromě vlastní lokalizace více možností nenabízí. Tracking je však to, na co se zaměřím. Bude mě zajímat, jak si povede jeho lokalizace ve srovnání s kamerou ZED mini. Má práce se tedy bude zaobírat i porovnáním těchto dvou zařízení, co se týče schopnosti jejich lokalizace. Obě tyto technologie pracují na podobném principu, jimž je optický inside-out tracking založený na přirozených bodů. Obě technologie si ukládají význačné body do souborů v počítači, které jsou pak při startu schopné načíst.

2.3.5 Interakce s virtuálním prostředím

Pro interakci s virtuálními objekty nám ani kamera ZED mini nestačí. ZED SDK nepodporuje snímání rukou, takovéto snímání bych tedy musel implementovat sám. Ovladače Windows Mixed Reality by se pro interakci s prostředím využít daly. Však pro scénáře, ve kterých uživatel interaguje se světem virtuálním i fyzickým zároveň, by ovladače v rukou byly nepraktické. Ovladače v rukou by omezily možnost interakce uživatele s reálným prostředím. Tento problém tedy vyřeším integrováním další hloubkové kamery Leap Motion Controlleru.

Leap Motion Controller je malá hloubková kamera založená na principu structured light. Je sestavená ze dvou IR kamer a tří IR LED diod. Kamery snímají IR záření s vlnovou délkou 850 nm. Kamera dokáže snímat hloubku do vzdálenosti 0,6 metrů. [10]

Toto zařízení je určeno ke snímání rukou na krátkou vzdálenost. Snímané ruce pak mohou být replikovány ve virtuálním světě. V mém případě připevním zařízení na headset pro virtuální realitu. Oprostím tak uživatelovi ruce od ovladačů a umožním mu hlubší interakci s prostředím. Tento přístup by mohl být vhodný i pro starší lidi, kteří si tolik nerozumí s novými technologiemi.

3 Praktická část

3.1 Testování charakteristik vybraných zařízení

V následujících kapitolách se zaměřím na charakteristiky vybraných zařízení, zejména na charakteristiky 3D rekonstrukce kamery ZED mini. Vytvořím základ pro mou demonstrační aplikaci, pomocí které budu charakteristiky testovat. Toto testování slouží zejména pro získání povědomí o tom, jak technologie funguje. Na základě toho budu dále postupovat ve vývoji demonstrační aplikace.

Demonstrační aplikaci, kterou budu později navrhovat, budou využívat lidé. Ti budou mít na hlavě headset pro virtuální realitu a neuvidí reálný svět kolem. Je tedy nutné zajistit to, aby nekolidovali s fyzickým prostředím. To budu realizovat pomocí umělých vodítek, které budou uživatelé následovat. Může se však stát, že se nějaký uživatel vzdálí od naplánované cesty. V tu chvíli hrozí jeho kolize s fyzickými objekty. Tento problém budu řešit tím, že ve chvíli, kdy se uživatel vzdálí od vyznačené cesty, mu zobrazím 3D model reálného fyzického prostředí, aby do něj nevrazil.

Nyní se mi nabízí několik způsobů, jak takovýto model získat. Jak lze tušit z obsahu předchozích kapitol, využiji možnosti 3D rekonstrukce kamery ZED mini. V následující kapitole vytvořím základ pro demonstrační aplikaci a následně se blíže zaměřím na charakteristiky 3D rekonstrukce poskytované kamerou ZED mini, přičemž se pokusím identifikovat její přednosti a nedostatky. V souvislosti s 3D rekonstrukcí mě bude zajímat to, jakou rychlostí se 3D síť generuje. Jak tato síť odpovídá realitě a jak velký prostor lze sítí pokrýt. Dále mě také bude zajímat vliv prostředí a osvětlení na tyto vlastnosti.

Další vlastností, která mě bude zajímat, je schopnost dynamické registrace obou vybraných zařízení. Efektivita dynamické registrace bude patrná z toho, jak správně bude naskenovaný 3D model zarovnán na fyzický svět. Tato vlastnost je z pohledu této práce klíčová a bude středem mé pozornosti při závěrečném testování demonstrační aplikace s lidmi.

V první řadě budu potřebovat sestavit a propojit vybraná zařízení. V případě headsetu s kamerou ZED mini použiji jako náhlavní displej Pimax 4K, který nepodporuje tracking. Tracking tedy zajistí kamera ZED mini, která bude zodpovědná i za 3D rekonstrukci. Posledním dílem bude Leap Motion Controller, který zajistí převod rukou do virtuálního světa. Na Acer headset pouze připevním další zařízení Leap Motion Controller.

Pro tvorbu demonstrační aplikace jsem zvolil herní engine Unity. Unity mi usnadní hodně práce, neboť nebudu muset psát vše od začátku. Mohu se tak více soustředit na vývoj aplikace jako takové. Ačkoliv mi Unity velmi urychlí práci, stále mi ponechává velký prostor pro mé vlastní představy a plány.

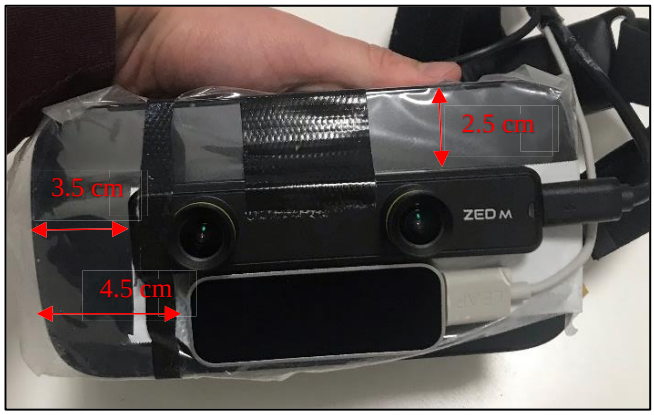
Testování charakteristik chci také ověřit celkovou funkčnost propojení vybraných zařízení. Po dokončení této kapitoly bude má aplikace schopna zapnout 3D rekonstrukci a také ji zastavit. Dále bude schopná naskenovaný model uložit a opět načíst.

3.2 Konfigurace zařízení a založení projektu

V této kapitole uvedu postup, jak jsem postupoval při propojování vybraných zařízení jak po stránce fyzické, tak po stránce softwarové. Při popisu postupu budu používat názvy *prefabů*[6], které jsou obsaženy ve zmíněných balíčcích. Názvy prefabů, proměných a *komponent*[7] budu psát kurzívou.

Ohledně záležitostí týkajících se Leap Motion Controlleru jsem se informoval na [11] a ohledně kamery ZED mini na [5].

Po stránce fyzické jsem pomocí oboustranné lepenky na Pimax 4K připevnil kameru ZED mini a pod ní Leap Motion Controller. Vše jsem zabezpečil kobercovou páskou, viz obr. 8. Rozměry jsou znázorněny červeně.



Obr. 8 Vlastní fotografie

V případě Acer headsetu jsem obdobně připevnil LeapMotion Controller, viz obr. 9.

Obr. 9 Vlastní fotografie

Pro správné fungování headsetu Pimax 4K jsem stáhl a nainstaloval aplikaci PiPlay ze stránek *pimaxvr.com*. K zajištění funkčnosti kamery LeapMotion Controller jsem si ze stránek *leapmotion.com* stáhl *Leap_Motion_Setup_4.0.0.exe*. Soubor jsem spustil a nainstaloval SDK. Dále jsem si ze stránek

[6] Prefab je název pro uložený herní objekt, ze kterého lze vytvářet ve scéně nové instance tohoto objektu.
[7] Unity je založeno na komponentách. Každý herní objekt je seskupením komponent, které definují jeho vlastnosti.

leapmotion.com stáhl plugin *Leap_Motion_Core_Assets_ 4.4.0.unitypackage*, který poskytuje základní rozhraní mezi Unity a Leap Motion Controllerem. Následně pak *Leap_Motion_Interaction_Engine_1.2.0.unitypackage*, který poskytuje interaktivní API pro interakci s virtuálními objekty. Ze stránek *stereolabs.com* jsem stáhl Unity plugin *ZEDUnityPlugin_v2.7.0_b2.unitypackage* pro kameru ZED mini. V Unity jsem vytvořil nový projekt a všechny tyto balíčky do projektu importoval. Posledním korkem ke zdárnému spuštění aplikace bylo stáhnout a importovat Steam VR Plugin, který je dostupný v Unity Assets Store.

Balíčky obsahují prefaby *ZED_Rig_Stereo* a *Leap Rig*. Při testování kamera ZED mini po přetažení těchto prefabů do scény nebyly mé reálné ruce synchronizované s virtuálníma. Zde uvádím svůj postup, který tento problém vyřešil.

1. Udělal jsem empty GameObject *HandsOrigin*, který jsem udělal potomkem *CameraEyes*
2. *HandsOrigin* jsem pak nastavil v *Leap Rig \ MainCamera* jako *Device Origin* pro *HandModels*
3. Aby správně fungoval script *SimpleFacingCameraCallbacks*, který rozpozná, kdy je dlaň natočená ke kameře, změnil jsem ve scriptu všechen výskyt Camera.main na kameru *LeftEye* ze *ZED_Camera_Rig_Stereo*. Jinak by bylo natočení rukou počítáno vzhledem ke kameře z *Leap Rig* prefabu, která teď není tam, kde má uživatel hlavu.

Na levou ruku jsem připevnil *HandAtachedUI*, které zobrazím, když je levá dlaň natočena k zapojenému zařízení. Toto jednoduché uživatelské rozhraní mi dovolí do scény vkládat objekty, které mi poslouží jako referenční body.

Spouštění 3D rekonstrukce i její ukončení se provádím pomocí skriptu *ZED Spatial Mapping Manager*, který je komponentou prefabu *ZED_Spatial_Mapping*. Při ukončení 3D rekonstrukce je možné 3D síť uložit do souboru specifikovaného v poli *Mesh Path.*

3.3 Analýza charakteristik

Tato kapitola je zaměřená zejména na analýzu charakteristik 3D rekonstrukce kamery ZED mini. V kapitole je i zhodnocena dynamická registrace obou zařízení a vliv osvětlení na jejich funkčnost.

3D rekonstrukci jsem testoval na třech různých místech. V prostorné podzemní garáži, kde se vyskytovaly velké holé plochy. Na chodbě ve škole, která byla o něco více zaplněna objekty. U sebe doma v bytě, kde byla koncentrace objektů nejvyšší a prostory nejmenší.

Nejhorší výsledky jsem dostával v garáži. Kamera nejspíše nenacházela záchytné body. 3D síť neodpovídala skutečnosti a kamera se občas v prostoru úplně ztratila. Podlaha byla trochu lesklá a odrážela se na ní světla. To vedlo k tomu, že se 3D síť na zemi negenerovala nebo se generovala velmi pomalu a nepřesně. Na školní chodbě a v bytě jsem dosáhl značně lepších výsledků. Výsledky následně porovnám v několika kategoriích.

3.3.1 Rychlost 3D rekonstrukce

Rychlost, jakou kamera ZED mini generuje 3D síť, je naprosto dostačující. Osobně jsem zkusil rychlou chůzi (s notebookem v ruce jsem nechtěl běhat), se kterou kamera neměla žádný problém. Navíc, při nastavení parametru *range* na *far*, se 3D síť generuje dopředu s velkým předstihem. Rychlé otáčení hlavou taktéž není pro kameru ZED mini problémem.

3.3.2 Rozsah 3D rekonstrukce

Rozsah je již větší problém. Zde velmi záleží na nastaveném parametru *resolution*. Při vysokém rozlišení lze zachytit jen velmi malou část okolního prostředí. Při nízkém rozlišení se tato oblast několikanásobně zvětší.

Zjistil jsem, že tato hranice nezpůsobuje kamera ZED mini ani její SDK, ale Unity. V Unity žádná 3D síť nesmí obsahovat více než 64k vrcholů. 3D síť se tedy přestává generovat z tohoto důvodu.

Tento problém se dá obejít tím, že zastavím 3D rekonstrukci a vygenerovanou 3D síť uložíme pod jiný *GameObject* (Původně je pod objektem *[ZED Mesh Holder]*). Následně 3D rekonstrukci opět spustím, a tak mohu rekonstruovat dále. Nevýhoda tohoto řešení je ta, že pokud se budeme pohybovat v místech, kde 3D síť již máme hotovou, budou se tyto sítě překrývat. To však není žádný fatální problém. Větším problémem podle mého bude ukládání a načítání takto vzniklých 3D sítí.

3.3.3 Registrace vybraných zařízení

Jak jsem zmiňoval v kapitole lokalizace, registrace v kontextu této práce znamená zarovnávání souřadných systémů fyzického světa a světa virtuálního, kterým je v tuto chvíli pouze generovaná 3D síť.

Zde hraje velkou roli, zda hodnotím registraci před ukončením 3D rekonstrukce, nebo až po ukončení.

Hodnotím-li registraci až po ukončení 3D rekonstrukce, dostávám výrazně lepší výsledky. 3D síť se již tolik nehýbe, někdy se dokonce lépe „usadí“ na reálný svět. Po delším chození v místech, kde je 3D síť namapována, se však občas začne zvedat směrem vzhůru (někdy i jiným směrem). To vyvolává pocit toho, že jste buď velmi malý nebo že se brodíte po kolena v zemi.

Registraci během 3D rekonstrukce nejvíce rozhazuje to, když se vracíme již zmapovaným prostředím. 3D síť se často lehce pootočí a není tak zarovnaná se skutečným světem. Orientovat se čistě podle takto vzniklé sítě mi přijde riskantní.

Poměrně dobrého výsledku jsem dosáhl při pokusu, kdy jsem prováděl 3D rekonstrukci během chůze z místa A do místa B. Na místo A jsem položil referenční bod (fialovou kouli). Po příchodu do místa B jsem 3D rekonstrukci vypnul, a následně jsem se vrátil zpět na místo A. Referenční se z místa pohnul jen o pár desítek centimetrů vzdálenost.

Již jsem zmínil, že 3D síť vytvořenou kamerou ZED mini lze uložit do souboru. Toho jsem využil a zkusil aplikaci spustit s náhlavním displejem Acer. Načetl jsem 3D síť do scény a manuálně od oka provedl kalibraci. Z důvodu nepřítomnosti kamery ZED mini jsem přišel o možnost vidět 3D síť společně s fyzickým světem skrz náhlavní displej, však pomocí hmatu a občasného sundání headsetu z očí jsem se vydal po bytě. K mému překvapení mi dynamická registrace pomocí Acer headsetu připadala mnohem lepší. 3D síť se téměř vůbec nevychylovala od fyzického světa. Při dotyku virtuální zdi jsem cítil zeď opravdovou. Toto jsou zatím pouze mé dojmy, mou finální aplikaci budu testovat na obou zařízení a podám konkrétní výsledky.

3.3.4 Přesnost 3D rekonstrukce

Přesností zde mám na mysli to, jak tvar 3D sítě odpovídá fyzickému prostředí. Zde opět velice záleží na zvoleném rozlišení. Při vysokém rozlišení je možné poměrně dobře zachytit menší část světa. Osamocené větší a jednodušší objekty lze identifikovat. Objekty menší a objekty geometricky složitější většinou identifikovat nelze. Objekty blízko u sebe splývají do jednoho.

Na přesnost před ukončením 3D rekonstrukce má negativní vliv nepřesnost registrace. Když se například vracím na místo, které jsem již zmapoval a 3D síť není správně zarovnaná s reálným světem, objekty se začnou znovu vykreslovat přes sebe. Tím dochází k zašumění objektu.

Nemyslím si, že by takto vzniklá 3D síť mohla představovat plnohodnotný virtuální prostor. Její využití vidím spíše jako přesnější ohraničení prostoru, kde se uživatel může volně pohybovat.

3.3.5 Vliv osvětlení

Osvětlení na fungování obou zařízení nehraje příliš roli. Zařízení si obecně vedou lépe při úplném osvětlením scény, však nemají problém ani s fungováním v temnějších prostorách. Ve dne v prostorách s okny za přítomnosti pouze denního světla nemají žádné neobvyklé potíže způsobené nedostatkem světla.

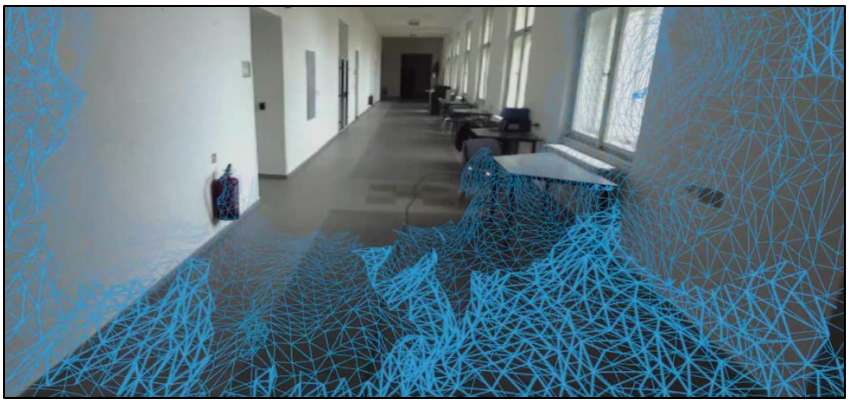
Až ve velmi tmavých prostorách, kde snímky přestanou být dostatečně kontrastní začnou mít zařízení problém. Kamera ZED mini při 3D rekonstrukci nenachází žádné význačné body a 3D síť se tak ve velmi tmavých místech netvoří anebo se tvoří nepřesně. Tracking obou zařízení občas vypadává, což se projevuje tím, že když se pohybuji, tak se naskenovaný 3D model hýbe semnou.

3.3.6 Pokus o realtime tvorbu virtuálního světa

Zkoušel jsem realizovat představu, že by se uživatel pohyboval ve virtuálním světě, do kterého by se v reálném čase pomocí 3D rekonstrukce replikoval svět reálný. Uživatel by spustil 3D rekonstrukci v prázdném prostoru, ve kterém by se před ním průběžně replikoval fyzický svět, jakožto 3D síť tvořená pomocí 3D rekonstrukce.

Zjistil jsem, že ZED plugin není napsán s představou, že by ho někdo chtěl používat tímto způsobem. Plugin poskytuje možnost vytvářet 3D rekonstrukci okolí v reálném čase. Nedává nám však možnost tuto 3D síť upravovat do doby, než je 3D rekonstrukce ukončena. Má představa tedy není za těchto podmínek zcela uskutečnitelná.

Základním nastavením kamery a 3D rekonstrukce je nastavení, kdy vidíme zároveň generující se 3D síť jakožto modrou síť hran (*wireframe*, viz obr. 10) spolu s okolním prostředím jakožto RGB obrazem poskytovaným kamerou ZED mini. RGB obraz lze vypnout a vidět je tak pouze wireframe 3D síť. Jak jsem již zmínil, 3D síť nelze upravovat, dokud není 3D rekonstrukce ukončena. Je tomu tak, protože po ukončení 3D rekonstrukce se nad sítí provádí optimalizační úpravy nebo například výpočet *mesh colideru*[8]. Tyto procesy by byly v reálném čase příliš početně drahé.



Obr. 10 Vlastní obrázek

Při spuštění aplikace se ve scéně vytvoří herní objekt *[ZED Mesh Holder]*. Ten po spuštění 3D rekonstrukce drží veškeré části právě vytvářené 3D sítě (*chunks*) jako své potomky.

Zkoušel jsem přistupovat k 3D síť a měnit její barvu za běhu 3D rekonstrukce. 3D síť jsem sice obarvil, ale takto obarvená 3D síť má na všech místech stejný odstín barvy. Není tedy možné usoudit, jaký má 3D síť tvar či jak je od nás daleko.

Zkoušel jsem také vždy po nějakém intervalu 3D rekonstrukci zastavit, přenést geometrii na jiný objekt a znovu spustit. Takto převedené geometrii je již možné změnit barvu a vypadá to správně. Bohužel výpočty prováděné při ukončení 3D rekonstrukce jsou velmi početně náročné a svět se tak na zhruba na jednu až dvě vteřiny zcela zastaví. Navíc takto vzniklá 3D síť má v místech, kde je zastíněna objekty z reálného světa, díry.

3.3.7 Závěr

Testování potvrdilo funkčnost propojených zařízení. V rámci vzájemného propojení bylo nutné udělat pouze několik menších úprav.

Co se týče testování charakteristik 3D rekonstrukce, tak jsem zjistil, že se kameře nejlépe vede v prostorách s vyšší koncentrací objektů. Rychlost tvorby 3D sítě mě neomezuje. Více mě omezuje její maximální

[8] Unity podporuje na herní objekty přidávat tzv. collidery. Collider definuje to, jakým způsobem se vypočítávají kolize objektu s jinými objekty ve scéně. Například pokud na kouli připevníte krychlový collider, bude koule na kolize reagovat tak, jako by to byla krychle. Mesh collider je typ collideru, který co nejpřesněji odpovídá geometrii objektu.

rozsah, který je dán maximálním počtem vrcholů pro jednu 3D síť, který Unity umožňuje. Zjistil jsem, že tento problém lze obejít ukončením 3D rekonstrukce a přemístěním naskenované 3D sítě na jiný herní objekt. Skenování většího prostoru tedy bude muset být provedeno v několika skenech.

Testování dále ukázalo, že 3D síť vzniklá pomocí 3D rekonstrukce neodpovídá fyzickému světu natolik, aby ji šlo použít jako plnohodnotnou náhradu virtuálního prostředí. Využít naskenovaný model jako vymezení prostoru pro uživatelův volný pohyb mi přijde jako vhodný způsob využití.

Dále jsem zjistil, že dynamická registrace kamery ZED mini má své nedostatky. Nepřesnost je občas natolik velká, že se podle naskenovaného modelu nedá orientovat. Tracking Acer headsetu mi přišel o mnoho lepší. Konkrétní výsledky však ukáže až testování s lidmi. Co se týče fungování obou zařízení, tak dobrě osvětlená scéna je výhodou, však obě zařízení zvládnou fungovat i v hůře osvětleném prostředí.

3.4 Návrh demonstrační aplikace

Demonstrační aplikací chci demonstrovat využití realtime lokalizace vybraných zařízení a 3D rekonstrukce kamery ZED mini pro lokalizaci a navigaci uživatele ve známém prostoru. Zajímat mě bude zejména sledování pozice uživatele v prostoru, jeho schopnost orientace ve fyzickém světě pomocí virtuálních vodítek a celková stabilita virtuálního světa.

Uživatel bude plnit scénář ve virtuálním světě, který bude registrován na svět fyzický. Plněním scénáře ve virtuálním prostředí bude uživatel interagovat i s fyzickým prostředím. Abych změřil správnost uživatelovi lokalizace, bude scénář obsahovat body, kterých se uživatel v průběhu plnění scénáře dotkne. Tyto body umístím na konkrétní body ve fyzickém světě a budu měřit vzdálenost mezi místem uživatelova dotyku a konkrétním bodem. Uživatel po dobu plnění scénáře nemusí mít žádné tušení o fyzickém světě kolem. Toto prohloubím tím, že uživatele umístím do vesmíru.

Aplikace bude zodpovědná za všechny kroky celého procesu, který je následující:

Uživatel zapne aplikaci a provede 3D rekonstrukci prostoru, do kterého chce scénář zasadit. Takto vzniklý model je možné uložit do souboru a opět načíst. Pokud je registrace načteného modelu s fyzickým světem chybná, pak bude mít uživatel možnost použít nástroje k opětovné registraci modelu s fyzickým světem. Dále uživatel pomocí editačního menu sestaví scénář, který umístí do scény vzhledem k naskenovanému modelu. Scénář je taktéž možné si uložit a opětovně načíst. Scénář se vždy načte na stejné místo vzhledem k naskenovanému modelu. Tedy pokud uživatel zarovná 3D model na fyzický svět, scénář se zarovná na 3D model, a tedy na fyzický svět. Ve chvíli, kdy má uživatel vytvořený scénář zarovnaný na fyzický svět, může přejít do exekučního modu. V tomto modu zmizí pomocné editační značky a uživatel se přenese do vesmíru, kde může začít plnit vytvořený scénář.

Aplikaci vytvořím pro dvě různá zařízení, hloubkovou RGB stereo kameru ZED mini a náhlavní displej Acer, přičemž v aplikaci pro Acer nebude možnost 3D rekonstrukce, neboť to neumí. Měřením, které jsem výše popsal následně porovnám, které z těchto dvou zařízení si vedlo lépe.

Při implementaci budu postupovat podle jednotlivých kroků procesu tvorby scénáře, který jsem popsal výše. Proces jsem rozdělil do několika následujících kroků:

1. 3D rekonstrukce
2. Manuální kalibrace pomocí Leap Motion
3. Kalibrace pomocí Acer ovladače
4. Tvorba a editace scénáře

Pro každý tento krok vytvořím samostatné GUI[9] podporující potřebné funkčnosti k dosažení cíle v každém kroku.

[9] Grafické uživatelské rozhraní, anglicky Graphical User Interface

3.5 Implementace demonstrační aplikace

3.5.1 Návrh GUI

Prvním krokem při implementaci aplikace je zajistit její ovládání. Začal jsem proto návrhem GUI systému. Při návrhu jsem čerpal nejvíce ze zdroje [3] převážně z kapitoly *Design Principles and Guidelines*.

Komplexní GUI vyžaduje vyšší kognitivní úsilí a uživateli zabere obecně více času při realizaci jeho cíle než jednoduché GUI s málo komponentami. [3] Snažil jsem se tedy o co nejjednodušší GUI s nízkým počtem komponent. Jediným typem komponent jsou tlačítka.

GUI by mělo být organizováno ve smysluplném a užitečném stylu [3]. Proto jsem se rozhodl GUI rozdělit do několika samostatných GUI. Každému kroku celého procesu, které zmiňuji v přechozí kapitole, patří alespoň jedno speciální GUI. Tato GUI vždy zaštiťují funkce spolu související. Navíc tak každé GUI obsahuje pouze několik tlačítek. Podle této struktury GUI jsem také postupoval při implementaci a stejnou strukturu má i tato kapitola.

Podle designerského principu *viditelnosti* by mělo být jasné, k čemu daný GUI prvek slouží. Aby uživatel mohl zformulovat plán pro dosažení svého cíle, musí rozumět tomu, k čemu dané komponenty slouží. Jedním z možností, jak přispět k tomuto pochopení, je využít visuální ikony a symboly. [3] Příliš neovládám programy pro tvorbu 2D grafiky, rozhodl jsem se tedy ikony vyhledat na internetu. Na stránce *flaticon.com* jsem našel balík ikon vyrobených společností *Google*. Použil jsem tedy vybrané ikony z tohoto balíku.

Podle mého by GUI mělo být vždy po ruce. V mé aplikaci jsem to vzal doslova. Uživateli zobrazím GUI při natočení jeho levé dlaně k obličeji a následně pohybuje se společně s ní, dokud uživatel ruku neodvrátí. Obličejem je po technické stránce kamera, kterou má uživatel na hlavě. Interakci s GUI uživatel provádí pomocí pravé ruky.

Ústředním skriptem GUI systému je skript *GUI Manager*. *GUI Manager* vlastní reference na všechna ostatní GUI. Je zodpovědný za přepínání mezi jednotlivými GUI a za to, že je uživateli zobrazeno vždy pouze jedno GUI v jednu chvíli. Při pozdějším testování funkčnosti GUI jsem zjistil, že při stisknutí tlačítka pro přepnutí GUI na jiné se často stiskne i tlačítko, které je v nově otevřeném GUI umístěné na stejném místě. Proto je ve skriptu *GUI Manager* možné nastavit čas, po který budou při změně GUI tlačítka deaktivována.

Třídou, od které všechna GUI dědí, je abstraktní třída *GeneralGUI*. Tato třída je zodpovědná za obstarání reference na *GUI Manager* a obsahuje metodu DisableForSeconds(), která deaktivuje tlačítka daného GUI.

Jako výchozí GUI je nastaveno *MainMenuGUI* obsahující čtyři tlačítka. Každé vede do jednoho z GUI patřícího určitému kroku z minulé kapitoly. Jak jsem již zmínil, tato kapitola bude mít podobnou strukturu jako mé GUI samo. Proto nyní budu postupovat od jednoho GUI k dalšímu, přičemž budu blíže popisovat funkčnosti, které tato GUI obstarávají.

3.5.2 Spatial Mapping GUI

Prvním krokem celého procesu je pořízení 3D modelu pomocí 3D rekonstrukce kamery ZED mini. Proto *Spatial Mapping GUI* obstarává čtyři funkce potřebné k dosažení tohoto cíle. Funkcemi jsou spuštění 3D rekonstrukce, zastavení 3D rekonstrukce, smazání aktuální 3D sítě a načtení dříve naskenované 3D sítě. *Spatial Mapping GUI* deleguje veškeré požadavky na skript *Scan Manager*. Ten vlastní referenci na *Spatial Mapping Manager*, pomocí kterého ovládá 3D rekonstrukci.

V kapitole Analýza 3D rekonstrukce jsem zmínil, že Unity nepovoluje 3D síť s více než 64k vrcholy. Problém jsem obešel tak, že jsem vytvořil nový herní objekt *Total Mesh Holder*. Po ukončení 3D rekonstrukce uložím 3D síť do .obj souboru s názvem ve tvaru *mesh + pořadové číslo skenu*. Vytvořím prázdný herní objekt, kterému nastavím herní objekt *Total Mesh Holder* jako rodiče a následně veškeré 3D síti nastavím tento prázdný herní objekt jako rodiče. To, že každý sken má svého vlastního rodiče, odůvodním v následující podkapitole. Celý tento proces chvíli trvá, to indikuje červená barva tlačítka pro spuštění 3D rekonstrukce. Po převedení 3D sítě na *Total Mesh Holder* je možné 3D rekonstrukci opět spustit. Počet aktuálně naskenovaných 3D sítí ukládám do souboru *MeshCount.txt.* Tento počet je resetován na 0 stisknutím tlačítka pro odstranění 3D sítě. Při stisknutí tohoto tlačítka se odstraní veškerá 3D síť připojená na *Total Mesh Holder*.

3.5.3 Mesh Adjustment GUI

Načtený model se ne vždy správně registruje s fyzickým světem. V tom případě se musím zarovnat manuálně. Toto GUI obstarává osm funkcí, kterými jsou vytvoření instance *Handle*, vytvoření instance *Rotator*, zničení těchto objektů, připnutí herního objektu *Spawn Point*, přepnutí mezi globální a lokální kalibrací, přepnutí mezi původní registrací a upravenou, uložení manuální registrace a načtení dříve uložené registrace.

Na stejném herním objektu jako je umístěno *Mesh Adjustment GUI* je umístěn skript *Object Spawner*, pomocí kterého se instancuje *Handle*. Po pravé straně GUI je malý model Země. Ten znázorňuje místo, kde se *Handle* instancuje. Tento malý objekt *Spawn Point*, který je potomkem GUI, lze rukou přemístit na libovolné místo. Pomocí tlačítka pro připnutí objektu *Spawn Point* ho lze odepnout z GUI nastavením rodiče na *null*. Pomocí stejného tlačítka *Spawn Point* připneme zpátky nastavením *Mesh Adjustment GUI* jako jeho rodiče a resetováním jeho lokální pozice. *Spawn Point* jsem zakomponoval do GUI, neboť vyloženě usnadňuje přesouvání 3D sítě na větší vzdálenost.

Handle je objekt, kterému po instancování vložím *Total Mesh Holder* jako argument funkce *SetAttachedObject(GameObject obj)*. Pohybem *Handle* se v měřítku 1:1 hýbe i připojený objekt, kterým je v mém případě *Total Mesh Holder*. Objektu *Rotator* je při instancování nutné vložit existující *Handle*. Otáčením objektu *Rotator* kolem *Handle* se otáčí i objekt připojený k *Handle*, kterým je tedy v mém případě *Total Mesh Holder*. Otáčením objektu *Rotator* kolem *Handle* se tak v měřítku 3:1 otáčí i *Total Mesh Holder*. To znamená, že tři otočení objekt *Rotator* kolem *Handle* znamená jedno otočení *Total Mesh Holderu* o 360°.

Obr. 11 Objekty Rotator (vlevo) a Handle (vpravo). Vlastní obrázek.

Dále je možné si vybrat rozsah změny, kterou způsobují *Handle* a *Rotator*. Změna může být buď lokální nebo globální. Při nastavení změny na lokální se při instancování *Handle* provede vertikální *raycast* z

objektu *Spawn Point* směrem *Vector3.down* a pokud paprsek neprotne 3D síť, pak je vyslán znovu ve směru *Vector3.up.* Pokud paprsek protne 3D síť, *Handle* i *Rotator* dále upravují pouze *Transform* rodiče protnuté sítě, který je pro každý sken jiný, viz předchozí podkapitola.

Lokální transformace jednotlivých 3D sítí lze uložit pomocí skriptu *Object Saver*. Pokud je 3D síť načtená, lze tyto transformace opět načíst.

3.5.4 Calibration GUI

Dalším způsobem, jak lze registrovat 3D síť s fyzickým světem, je využít *Calibration GUI*. To poskytuje pouze dvě funkce. Nutné je mít připojený Acer ovladač. První funkcí, kterou *Calibration GUI* obstarává je translace objektu *Total Mesh Holderu* k Acer ovladači. Druhou funkcí je nastavení Acer ovladače jako rodiče objektu *Total Mesh Holder.* Pohybem ovladače tak hýbeme i s veškerou 3D sítí.

3.5.5 Edit GUI

Dalším GUI je *Edit GUI,* to slouží pro tvorbu a editaci scénáře. Z *Edit GUI* se dále dá dostat do *Catalog GUI*, pomocí kterého lze do scény přidávat objekty, které budu nazývat katalogovými objekty. Tyto objekty se dělí do čtyř druhů a z těchto objektů se skládá scénář. Skripty definující chování jednotlivých katalogových objektů dědí od třídy *Interactive Object.* Tato třída obstarává referenci na *Object Manager* a definuje virtuální metodu *OnTouch(),* která je volána při styku uživatelovi ruky a objektu. Nyní blíže popíši jednotlivé katalogové objekty včetně jejich GUI.

Path Point

Path Point je stavební prvek cesty, která má uživateli dát najevo, kudy se má vydat. Je reprezentovaný bílým světlem, který jsem na tento prefab přidal pomocí komponenty *Light.* Cesta je reprezentovaná jako prázdný herní objekt se skriptem *Start Path. Star Path* si drží reference na jednotlivé objekty *Path Point,* ze kterých se cesta skládá. Každá *Star Path* má své unikátní identifikační číslo *pathIdx.* Cesta může být buď deaktivovaná, což je její defaultní stav, nebo aktivovaná. Deaktivovaná cesta nijak neupoutává svou pozornost, její objekty *Path Point* svítí bíle. O bjekt*Path Point* aktivované cesty postupně problikávají ve směru, kterým má uživatel cestu následovat. Problikávání jsem dosáhl využitím animace *Light Up,* která mění *radius* a *color* komponenty *Light.*

Path Point GUI obstarává čtyři funkce, přidání objektu *Path Point*, smazání posledního objektu *Path Point,* zrušení cesty a možnost levitace objektu *Path Point.* Funkce přidání a odebrání objektu *Path Point* buď přidají nebo odeberou *Path Point* skrze *Object Manager,* o kterém budu psát později. Zrušení cesty také probíhá skrze *Object Manager*. Pokud je levitace zapnuta, při vytvoření nového objektu *Path Point* se provede *raycast* ve směru *Vector3.down.* Pokud paprsek protne 3D síť, *Path Point* se následně snese dolu nad 3D síť, kde začne levitovat. *Path Point* snáším ke 3D síti pomocí korutiny *TravelToGroundPoint(PathPoint pathPoint, Vector3 groudPoint),* které předám jako argument daný *Path Point* a určím místo nad 3D sítí, na které se má *Path Point* snést. K tomu využívám lineární interpolace funkce *Vector3.Lerp().*

Target

Target je prefab, pomocí kterého lze aktivovat cestu. Obsahuje skript *Target Visualizer*. Při instancování objektu *Target* se otevře velmi jednoduché GUI, která nám umožňuje *Target* smazat nebo se vrátit zpět do *Catalog GUI*. Ve chvíli, kdy máme *Target GUI* otevřené, lze *Target* přiřadit libovolné již vytvořené cestě. Přiřazení cesty se realizuje dotykem uživatele a objektu *Path Point* patřícího vybrané cestě. *Path Point* při doteku zavolá funkci *OnTouch()* a zjistí si, zda je *Targer GUI* otevřené. Pokud je tomu tak, pak zavolá na

Object Managerovi funkci *AssignPathToCurrentTarget(int pathIdx),* které přidá identifikátor *pathIdx* cesty pod kterou spadá. Propojení objektu *Target* a *StartPath* je znázorněno pomocí komponenty *Line Renderer* spojující první *Path Point* dané cesty a *Target.*

Danger Point

Danger Point je jediným katalog objektem, který nepotřebuje své GUI. Jeho role je jednoduchá. Slouží jako indikátor kritického místa, kde hrozí kolize uživatele s fyzickým světem. V momentě, kdy se uživatel vzdálí od cesty, kterou má následovat, zobrazím mu naskenovaný 3D model. Však v některých místech, kde hrozí větší riziko srážky uživatele s fyzickým světem, jako je tomu například při průchodu dveřmi, je nutné být více opatrný. *Danger Point* v tomto ohledu funguje vlastně opačně než *Path Point.* Když uživatel bude nebezpečně blízko tomuto objektu, zobrazím mu naskenovaný 3D model, aby nekolidoval s objektem, který *Danger Point* zastupuje.

Obstacle

Nejjednodušším katalogovým objektem je *Obstacle. Obstacle* slouží k vizualizaci objektů z fyzického světa, které nebyly přítomny v době 3D rekonstrukce. Jedná se většinou o mobilní nábytek. Aby se nemusela celá 3D rekonstrukce dělat znovu, je možné tyto objekty z fyzického světa vizualizovat ve světě virtuálním pomocí objektu *Obstacle.* Po vložení *Obstacle* do scény se zobrazí *Obstacle GUI,* pomocí kterého lze měnit *scale* u *Obstacle*. To realizuji instancováním prefabu *ResizeruEarth,* který má na sobě připnutý *Resizer* skript. *ResizerEarth* se objeví vedle *Obstacle.* Vzdálenost objektu *Resizer* od středu *Obstacle* je úměrná poloměru *Obstacle.* Tuto vzdálenost uložím do proměnné *baseDistance.* Posouváním objektu *ResizerEarth* měním *scale Obstacle* podle vzorce *obstacle.transform.localScale = resizerDistance / baseDistance * originalScale,* kde *resizerDistance* je aktuální vzdálenost objektu *ResizerEarth* od středu *Obstacle* a *originalScale* je defaultní *scale* prefabu *Obstacle.*

Ústředním skriptem pro správu veškerých katalogových objektů je *Object Manager.* Za jejich ukládání a načítání zodpovídá *Object Saver.*

Object Manager vlastní reference na veškeré katalogové objekty. Katalogový objekt lze do scény přidat pouze přes *Object Manager,* který si tak přidá referenci na nově vzniklý katalogový objekt do listu. V případě objektu *Path Point* si *Object Manager* nedrží referenci na každý *Path Point,* ale pouze na cestu, pod kterou daný *Path Point* spadá. Při přidání nového objektu *Path Point,* který je prvním objektem *Path Point* cesty, *Object Manager* tuto cestu vytvoří a přidá si referenci na ni do listu. Odstranění objektu ze scény opět probíhá přes *Object Manager.* Pro uložení katalogových objektů do souboru je volána metoda SaveObjects(), která tento požadavek deleguje na *Object Saver,* jenž je ve scéně připojen na stejný herní objekt jako *Object Manager*.

Object Saver je skript zodpovídající za ukládání a načítání katalogových objektů. Vlastní reference na prefaby katalogových objektů, aby je mohl instancovat. Dále má *Object Saver* specifikované cesty k soborům pro ukládání dat. Každý druh katalogových objektů ukládá do speciálního textového souboru pro daný druh. Ukládám vždy jen pouze data nutné ke správnému instancování daného katalogového objektu do scény. Obsah daných textových souborů pro jednotlivé druhy objektů je následující:

FileForPaths.txt: Ukládám počet *Start Path.* Pro každou *Start Path* pak její pozice, *pathIdx* a počet objektů *Path Point.* Pro každý *Path Point* jeho pozici.
FileForTargets.txt: Ukládám počet objektů *Target.* Pro každý *Target* jeho pozici a rotaci.
FileForObstacles.txt: Ukládám počet objektů *Obstacle.* Pro každý *Obstacle* jeho pozici a rotaci.
FileForDangerPoints.txt: Ukládám počet objektů *Danger Point.* Pro každý *Danger Point* jeho pozici a rotaci.

Přechod mezi editačním krokem a exekučním krokem realizuje skript *Scene Manager*. Obsahuje metody *EnterSpace()* a *ExitSpace()*. Volání metody *EnterSpace()* způsobí přechod do exekučního kroku. Prakticky to znamená, že nastavím *skybox*[10] na vesmírný. Dále přes *Object Manager* deaktivuji všechny cesty. Přestanu vykreslovat veškeré pomocné značení, jako je například znázornění toho, jaký *Target* patří k jaké cestě. Dále spustím korutinu *SpawnAsteroid()* skriptu *Asteroid Spawner* a nakonec veškeré 3D síti nastavím difusní RGBa materiál[11] a aktivuji výpočet alfa složky ve skriptu *Boundary Checker*.

Asteroid Spawner zodpovídá za instancování asteroidů. To provádím tak, že pomocí funkce *Random.insideUnitCircle()* získám náhodný bod ležící uvnitř kruhu s poloměrem 1. Tento bod vynásobím se *spawnRadius*, což mi dá náhodný bod uvnitř kruhu s poloměrem *spawnRadius*. Tento kruh posouvám o *userDistance* ve směru *Vector3.right* od *userHead*, což je herní objekt reprezentující zařízení, které má uživatel na hlavě. V získaném bodě instancuji *Asteroid*, který posílám směrem k uživateli náhodnou rychlostí a náhodnou rotací pomocí funkce *Random.Range(float min, float max)*. Obdobně mu přidám náhodnou velikost na všech třech osách. Tuto náhodnou velikost následně násobím *scaleMultiplier*, což je float, který se zvětšuje se vzdáleností od středu kružnice. Díky *scaleMultiplier* zamezuji srážce *Asteroidu* s uživatelem.

Boundary Checker je skript zodpovídající za zobrazení naskenovaného modelu uživateli v momentě, kdy se vzdálí od cesty nebo je příliš blízko nějakému objektu *Danger Point*. Pomocí funkce *Physics.OverlapSphere(Vector3 posiiton, int radius)* získám všechny *Path Points* a *Danger Points*, ze kterých vyberu jeden nejbližší *Path Point* a jeden nejbližší *Danger Point*. Na základě vzdálenosti uživatele od těchto herních objektů vypočítávám výslednou alfa složku pro materiál přiřazený naskenovanému 3D modelu.

Ve své demonstrační aplikaci jsem použil z Unity Asset Store:

Vesmírný skybox od *NIGHTSOUNDGAMES*.
Model planety Země od *Digital Ruby*.
Modely asteroidů od *Mark Dion*.

Ve své demonstrační aplikaci jsem použil ze stránky *free3d.com*:

Model Měsíce od *gerhald3d*.
Model Slunce od *jmbosch*.

[10] Skybox je textura znázorňující prostředí v nekonečnu. Při pohybu uživatele ve scéně se skybox hýbe spolu s ním. Není tedy možné se k této textuře přiblížit.
[11] rgba materiálem mám na mysli materiál, který podporuje kromě hodnot červené, zelené a modré barvy i alfa složku. Alfa složka udává neprůhlednost, tedy pro alfa = 0 je materiál průhledný a pro alfa = 1 je neprůhledný.

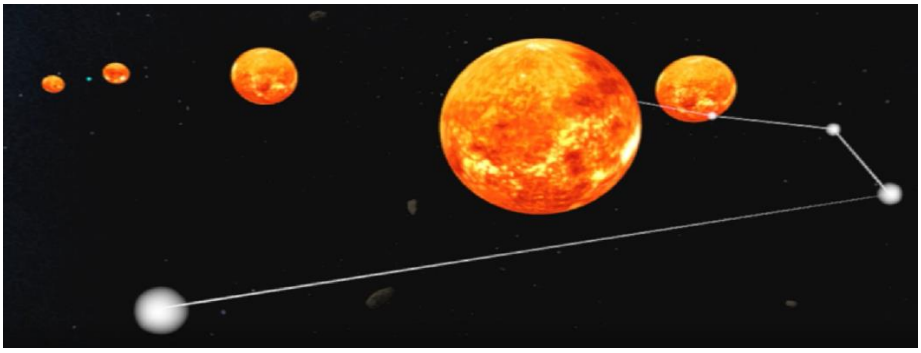
3.6 Testování aplikace s lidmi

Demonstrační aplikaci jsem se rozhodl testovat na chodbě ve škole. Školní chodba je prostorné prostředí a umožňuje tak pohyb na území desítek metrů čtverečních. Chodba, kterou jsem vybral byla podle mého dostatečně zaplněna objekty k tomu, aby tracking zařízení fungoval správně. Jednalo se zejména o nástěnné obrazy a nástěnky, veliké rostliny v květináčích nebo židle a skříňky. Při testování jsem byl omezen napájecím kabelem dlouhým 15 metrů. Ten jsem zasunul do zásuvky v polovině cesty a získal jsem tak úsek dlouhý 30 metrů. Zhruba v polovině této cesty chodba vedla do zatáčky pod úhlem 90°.

Myslím si, že charakteristika školní chodby je typická pro místa, kde by se tato technologie dala potencionálně využívat. Tím myslím spíše rozsáhlejší prostředí, ve kterém technologie bude plnit účel navigace v prostoru pro uživatele neznámém. Můj scénář jsem tedy zasadil do tohoto prostředí.

Před pokusem jsem pomocí 3D rekonstrukce kamery ZED mini prostor naskenoval a skeny uložil. Bohužel se mi modely nepodařilo ze souboru načíst. Při spouštění sestavené aplikace se 3D síť neobjevovala. Při spouštění z Unity jsem dostával errory, se kterými jsem se do té doby nesetkal, přičemž Unity přestávalo odpovídat. Ani po několika hodinách se mi nepovedlo naskenovaný model načíst tak, jak jsem byl zvyklý. Nezbývalo mi již mnoho času do příchodu prvního z testovacích subjektů. Byl jsem proto nucen se obejít bez 3D modelu.

Scénář jsem chtěl vytvořit takový, aby dokázal ověřit efektivitu lokalizace testovacích zařízení, a pomocí kterého bych dokázal získat přesná číselná data. Zároveň jsem chtěl, aby měl alespoň nějaký dopad na reálný fyzický svět. Rozhodl jsem se vytvořit scénář, který uživatele dovede ze startovního místa konec chodby. Po dokončení každého úseku se uživatel musel rukou dotknout *Target* objektu, aby aktivoval další cestu. Objekty *Target* jsem ve virtuálním světě umístil na místo, na kterém se ve fyzickém světě vyskytoval vypínač. Uživatel měl tedy plněním scénáře postupně pozhasínat světla na chodbě. Vypínač byl vždy umístěn až na konci sekce, kterou zhasínal. Osvětlení tedy zhasnutím nebylo ovlivněno. Po cestě jsem rozmístil několik objektů typu *Obstacle*, aby byla cesta pestřejší.



Obr. 12 Scénář po aktivaci první cesty. Vlastní obrázek.

Uživatelům jsem sdělil, aby se pokusili do objektu *Target* píchnout prstem, jelikož jsem chtěl znát přesný bod dotyku. Na toto místo jsem následně nalepil pomocný štítek, jehož vzdálenost od středu vypínače jsem po absolvování celé cesty změřil.

Většině subjektů jsem oblékl nejprve Acer headset a absolvoval s ním cestu. Zvolil jsem subjekt nejprve poslat s headsetem Acer, neboť jsem tušil, že bude fungovat lépe. Přišlo mi tedy vhodné, aby si subjekt nejprve vyzkoušel stabilnější variantu.

3.6.1 Analýza výsledků

Tabulka1 znázorňuje vzdálenost mezi bodem dotyku a středem vypínače v závislosti na vzdálenosti od startovní pozice. Musím zmínit fakt, že bez přítomnosti modelu pro mě bylo mnohem obtížnější správně registrovat virtuální objekty s reálným světem. Nějaká nepřesnost tedy může být zapříčiněná právě touto kalibrací.

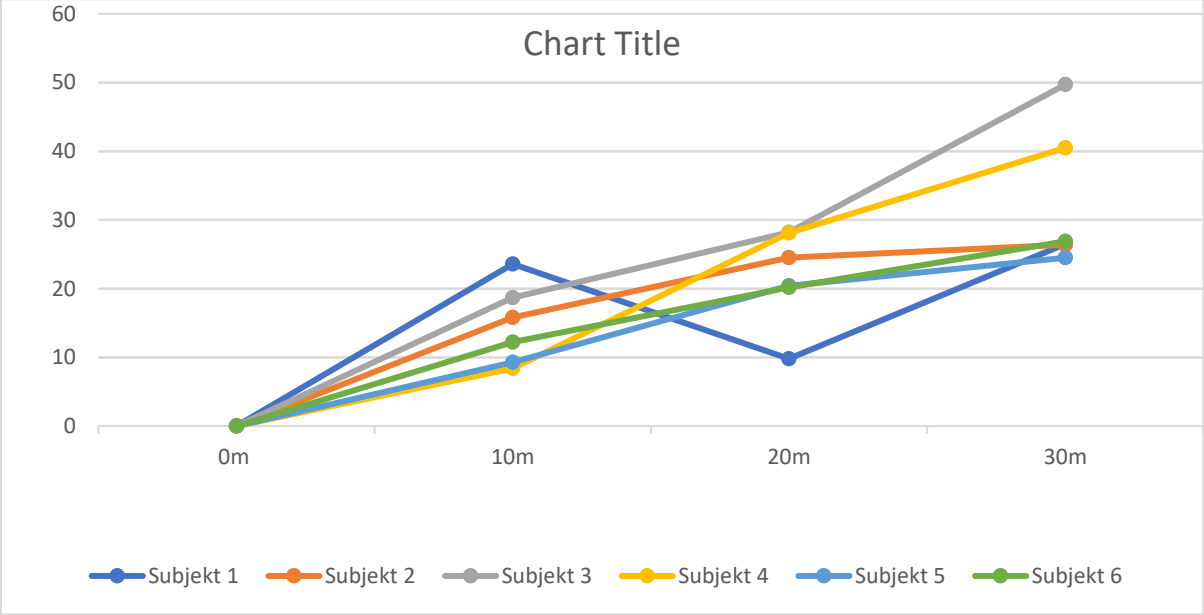
Při testování výsledků jsem dospěl k názoru, že by bylo vhodné rozlišit i způsob nepřesnosti. Měřit vzdálenost mezi bodem, kde se subjekt dotkl zdi a středem vypínače vždy nešlo. To z toho důvodu, že ne vždy se subjekt při dotyku objektu *Target* dotkl zároveň i zdi (B). Někdy se zase stávalo, že se subjekt nemohl dotknout objektu *Target*, neboť mu v tom bránila zeď. *Target* byl tak pro subjekt „ve zdi". Subjekt se tedy dotkl zdi, ale ne objektu *Target* (C). Poslední možností je ta, kdy se subjekt nemohl do místa, kde byl *Target* umístěn vůbec dostat (D). Vzdálenosti větší než metr značím jako 1+.

A = Dotek zdi i objektu *Target*
B = Dotek zdi, ale ne objektu *Target*
C = Dotek objektu *Target*, ale ne zdi
D = Nepovedlo se k objektu *Target* přiblížit
X = Neprovedeno

Tabulka 1

Subjekt	Zařízení	11,5 m	21,5 m	30 m
1	Acer	23,6 A	9,8 A	26,5 A
	ZED mini	X	X	X
2	Acer	15,8 A	24,5 A	26,4 A
	ZED mini	95,0 C	1+ B	D
3	Acer	18,7 A	28,2 A	49,7 A
	ZED mini	80,0 C	1+ B	D
4	Acer	8,4 A	28,1 B	40,5 A
	ZED mini	1+ C	1+ B	D
5	Acer	9,3 A	20,4 A	24,5 A
	ZED mini	80,0 C	1+ B	D
6	Acer	12,2 A	20,2 A	26,9 A
	ZED mini	85,0 C	1+ B	D

Graf 1



S využitím lokalizace pomocí Acer headsetu se všem subjektům podařilo dojít až na konec chodby. Pouze jednomu subjektu se stalo, že by měl *Target* „ve zdi“. Vzdálenost objektu *Target* „za zdí“ však byla nepatrná a po chvíli se subjektu podařilo *Target* aktivovat. Největší nepřesnost byla naměřena subjektu s číslem 3, která dosáhla ve vzdálenosti 30 metrů od startovní pozice necelého půl metru. Z grafu 1 lze vypozorovat, že závislost nepřesnosti na vzdálenosti od začátku je ve většině případů téměř lineární.

Kamera ZED mini si vedla o poznání hůře. Již na prvním stanovišti naměřené vzdálenosti dosahovali až metru. Na druhém stanovišti se žádnému subjektu nepodařilo aktivovat *Target*, který byl pro subjekty „ve zdi“. To znemožnilo postup v cestě směrem k poslednímu stanovišti.

3.7 Shrnutí

Při výběru zařízení pro realtime lokalizaci uživatele ve známém prostoru jsem se zaměřil na hloubkové kamery, které poskytují hloubková data, na základě kterých je možné pomocí algoritmů počítačového vidění tato zařízení lokalizovat v prostoru. Dále jsem blíže popsal problematiku lokalizace a uvedl různé lokalizační přístupy. Na základě analýzy hloubkových kamer a lokalizačních přístupů vhodných pro dosažení cíle této práce jsem vybral dvě konkrétní zařízení, a to Acer Windows Mixed Reality headset a kameru ZED mini.

V rámci praktické části jsem nejdříve popsal, jak jsem postupoval při propojení jednotlivých zařízení, jak po fyzické stránce, tak po stránce softwarové. Dále jsem se zabýval charakteristikami fungování jednotlivých zařízení, zejména 3D rekonstrukce kamery ZED mini. Testováním jsem následně odhalil několik nedostatků. Hlavním nedostatkem se stala nedostatečná maximální velikost 3D sítě, což bylo způsobeno omezením Unity. Druhým hlavním nedostatkem se stala špatná dynamická registrace kamery ZED mini, jež způsobovala odchýlení 3D sítě od fyzického světa. Ta však byla hodnotícím aspektem zkoumaných zařízení. Toto testování také ukázalo, že naskenovaný 3D model se podobá fyzickému světu velmi málo na to, aby šel sám o sobě využít ve virtuální realitě jako model prostředí.

Na základě nedostatků identifikovaných při testování charakteristik vybraných zařízení jsem navrhl a implementoval aplikaci. Ta mi umožnila podrobněji testovat jejich schopnost lokalizace v závislosti na vzdálenosti od startovní pozice plněním vytvořeného scénáře. Mimo pouhého plnění scénáře aplikace umožňuje jeho tvorbu i následnou editaci, což umožňuje aplikaci využít v libovolném prostoru.

Pomocí mé aplikace jsem podrobil vybraná zařízení testování s uživateli, které jsem prováděl na školní chodbě. Uživatelé měli za úkol plnit scénář, který je navigoval po trase dlouhé 30 metrů. Při testování jsem se zaměřil zejména na správnost trackingu daných dvou zařízení, který je zodpovědný za dynamickou registraci. To v kontextu této práce znamená zarovnání souřadných systému virtuálního a reálného světa. Registraci jsem měřil na třech stanovištích v průběhu cesty, na kterých měl uživatel stisknout vypínač, který byl ve virtuálním světě reprezentován hvězdou (herním objektem *Target*). Na každém stanovišti jsem měřil vzdálenost, o kterou uživatel vypínač svým dotekem minul. Hodnotil jsem i zda byl uživatel schopen celý scénář vykonat.

3.8 Závěr

Nepřesnost trackingu kamery ZED mini byla natolik velká, že se žádný uživatel nebyl schopný dostat na konec chodby pomocí vodítek z virtuálního světa. Již na první zastávce se nepřesnosti pohybovali ve vzdálenostech dosahujících až metr. Na základě toho tvrdím, že stabilita virtuálního světa s využitím lokalizační schopnosti kamery ZED mini je velmi nestálá. Využití této kamery pro lokalizaci v prostoru mi nepřijde vhodné, a proto nedoporučuji tuto kameru tímto způsobem využívat.

Ačkoliv tracking kamery ZED mini při testování neobstál, bylo toto zařízení přínosné pro svou schopnost 3D rekonstrukce. Tu jsem ve své demonstrační aplikaci využil jako vymezení prostoru pro volný pohyb uživatele. Naskenovaný 3D model jsem nechával neviditelný a zobrazil ho ve chvíli, když se uživatel odklonil od cesty. Přítomností tohoto modelu jsem chtěl zamezit pocitu nejistoty, který byl na subjektech patrný při jejich pohybu prostorem. Očekával jsem, že když si uživatel bude moci ověřit, že se 3D model zobrazí ve chvíli, kdy se vzdaluje od cesty, a že při dotyku tohoto modelu se dotýká fyzické zdi, zbaví je to pochybnosti o bezpečnosti prostoru, ve kterém se právě nacházejí. Bohužel z technických důvodů jsem nebyl schopen naskenovaný model při testování načíst, a tak jsem tuto myšlenku nemohl ověřit.

Z výsledků vyplývá, že tracking headsetu Acer byl výrazně lepší. Zjistil jsem, že Acer headset, ačkoliv povoluje uživateli při běžném používání nastavit herní prostor maximálně o velikosti 4 x 4 metry, je schopný poměrně dobré lokalizace i na vzdálenost mnohem větší. V jeho případě žádný uživatel nekolidoval s fyzickým světem a každý se dokázal dostat až na konec chodby. Nepřesnost se na konci chodby ve vzdálenosti 30 metrů od startovní pozice pohybovala v rozsahu do půl metru. Závislost mezi vzdáleností a nepřesností byla zhruba lineární. Na základě toho, že se každý uživatel bezpečně dostal až na konec chodby si dovoluji si tvrdit, že Acer headset je možné využít pro realtime lokalizaci uživatele ve virtuální realitě a jeho navigaci ve známém prostoru s ohledem na lineární nárůst nepřesnosti v závislosti na vzdálenosti od startovní pozice.

Z těchto závěrů je nutné vyčíst to, že ne každá hloubková stereo RGB kamera je schopna dostatečně správné lokalizace pro navigaci lidí v prostoru. Stabilita virtuálního světa za využití lokalizace různých stereo RGB kamer se může výrazně lišit. Proto je nutné podrobit konkrétní vybrané zařízení testování dříve, než se začne využívat.

Z výsledků dále vyplývá, že při navrhování virtuálního světa podporujícího chůzi jakožto způsob přemisťování by měl designer brát v potaz jeho rozlohu. Rozloha virtuálního světa má v tomto případě vliv na jeho stabilitu.

3.9 Zdroje

[1] SCHMALSTIEG, Dieter a Tobias HÖLLERER. *Augmented Reality: Principles and Practice*. Addison Wesley, 2016.

[2] AUKSTAKALNIS, Steve. *Practical Augmented Reality*. Addison Wesley, 2017.

[3] LAVIOLA JR., Joseph J. a spol. *3D User Interfaces: Theory and Practice (2nd Edition)*. Addison Wesley, 2017.

[4] LAVALLE, Steven M. *VIRTUAL REALITY*. University of Oulu: Cambridge University Press, 2019.

[5] *Stereolabs* [online]. [cit. 2019-01-16]. Dostupné z: https://www.stereolabs.com

[6] *Bastler Time-of-Flight Camera* [online]. [cit. 2019-05-21]. Dostupné z: https://www.baslerweb.com/en/products/cameras/3d-cameras/time-of-flight-camera/

[7] *Kinect Structured light 3D Imaging Depth Camera* [online]. [cit. 2019-05-21]. Dostupné z: https://www.alibaba.com/product-detail/Kinect-Structured-light-3D-Imaging-Depth_60813076160.html?spm=a2700.7724857.normalList.17.25317cce38eBWC

[8] KÁLOVÁ, Ilona a Karel HORÁK. *Optické metody měření 3D objektů* [online]. 2005 [cit. 2019-01-16]. Dostupné z: http://www.elektrorevue.cz/clanky/05023/

[9] Fotogrammetrie. *Wikipedia* [online]. 2018 [cit. 2019-01-18]. Dostupné z: https://cs.wikipedia.org/wiki/Fotogrammetrie

[10] *Https://www.leapmotion.com* [online]. [cit. 2019-01-16]. Dostupné z: https://www.leapmotion.com

[11] Leap Motion Unity Modules [online]. 2018 [cit. 2019-01-18]. Dostupné z: https://leapmotion.github.io/UnityModules/index.html

[12] SCARAMUZZA, Davide a Friedrich FRAUNDORFER. *Visual Odometry* [online]. 2011 [cit. 2019-05-21].

[13] RUI, Xianfeng. *A System of Image Matching and 3D Reconstruction* [online]. [cit. 2019-05-21]. Dostupné z: https://web.stanford.edu/class/cs231a/prev_projects_2016/CS231A_finalreport.pdf

[14] MANLEY, Eric D., Huzaifa Al NAHAS a Jitender S. DEOGUN. *Localization and Tracking in Sensor Systems* [online]. [cit. 2019-05-21]. Dostupné z: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33304.pdf

[15] HORAUD, Radu a spol. *An Overview of Depth Cameras and Range Scanners Based on Time-of-Flight Technologies* [online]. 2016 [cit. 2019-05-21]. Dostupné z: https://hal.inria.fr/hal-01325045/document

[16] GENG, Jason. *Structured-light 3D surface imaging: a tutoria* [online]. 2011 [cit. 2019-05-21]. Dostupné z: http://www.rtbasics.com/Downloads/IEEE_structured_light.pdf

[17] *CSC5280 Project 2: Feature Detection and Matching* [online]. [cit. 2019-05-21]. Dostupné z: http://mmlab.ie.cuhk.edu.hk/archive/gbq/csc5280_project_2.htm

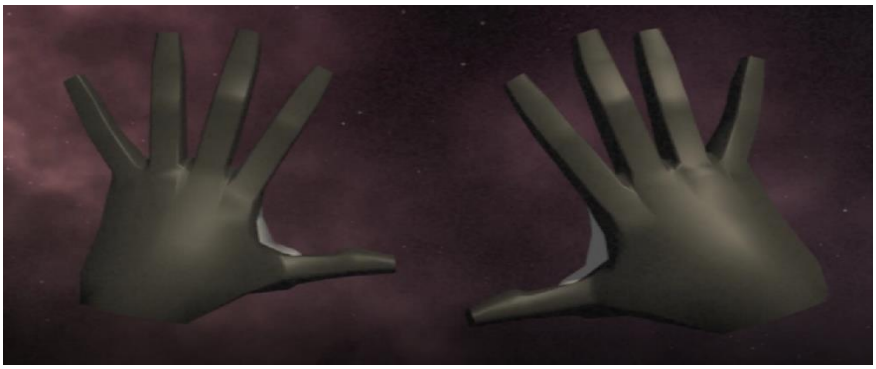
[18] *XR Glossary* [online]. [cit. 2019-05-22]. Dostupné z: https://delight-vr.com/xr-glossary/

[19] *Buy Acer Windows Mixed Reality* [online]. [cit. 2019-05-22]. Dostupné z: https://www.microsoft.com/en-ca/p/acer-windows-mixed-reality-headset-developer-edition/8pb4twx13m2n?activetab=pivot%3aoverviewtab

Příloha A
Manuál

Základní ovládání:

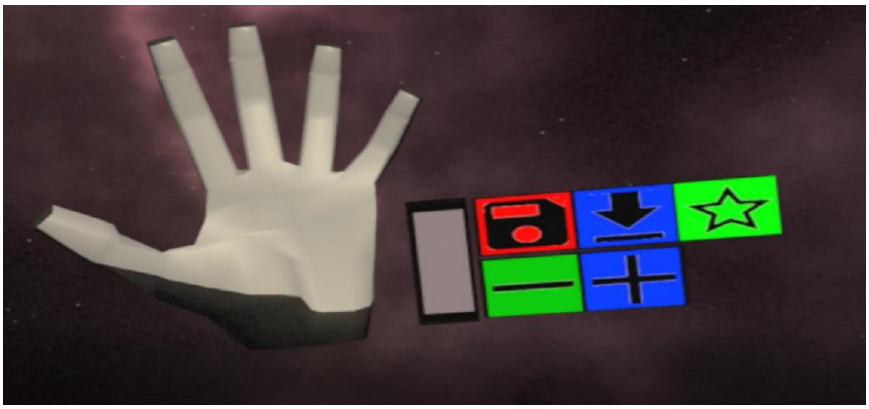
Uživatel se ve virtuálním prostředí pohybuje chůzí. Pro pohyb tedy není třeba žádného ovladače. S virtuálními objekty uživatel interaguje rukama. LeapMotion Controller se stará o přenos fyzických rukou do virtuálního světa. Modely rukou jsou vidět na obr. 13.



Obr. 13 Vlastní obrázek

Menu:

Menu uživatel zobrazí natočením levé dlaně k obličeji, viz obrázek 14. Menu se zobrazí na pravé straně této ruky a hýbe se společně s ní, dokud uživatel ruku neodvrátí nebo ruka nezmizí ze zorného pole LeapMotion Controlleru. Pravou rukou uživatel s tímto menu interaguje. Menu je sestaveno z tlačítek, která uživatel aktivuje jejich stisknutím pomocí pravé ruky.



Obr. 14 Vlastní obrázek

Nyní budu postupovat podle jednotlivých sekcí menu a popisovat jednotlivé funkce.

Main Menu

Main Menu je výchozí menu, pomocí tlačítek Main Menu se lze dostat do jiných menu.

Zobrazí Spatial Mapping Menu

Zobrazí Mesh Adjustment Menu

Zobrazí Calibration Menu

Zobrazí Edit Menu

Spatial Mapping Menu

Spatial Mapping Menu slouží k ovládání 3D rekonstrukce.

Spustí 3D rekonstrukci.
Zelená barva indikuje to, že je aplikace připravena na další rekonstrukci. Červená barva značí, že se zpracovává právě ukončená 3D rekonstrukce.

Ukončí 3D rekonstrukci.

Načte posledně naskenovaný prostor.

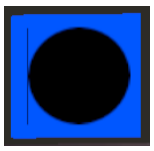
Červená barva indikuje načítání. V tom případě vyčkejte, než se barva tlačítka změní zpět na modrou.

Smaže aktuálně načtenou 3D síť.

Mesh Adjustment Menu

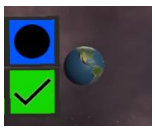
Mesh Adjustment Menu slouží k manuální úpravě načtené 3D sítě.

Změní oblast působnosti kalibrace. Pokud svítí zeleně, kalibrace se provádí v globálním měřítku. Pokud svítí červeně, kalibrace se provádí v lokálním měřítku.



Instancuje model Země, jehož pohybem se zároveň pohybuje i 3D síť podle nastavené oblasti působnosti.

Instancuje model Měsíce. Obíháním Měsíce kolem Země se otáčí 3D síť kolem své osy podle nastavené oblasti působnosti.



Tento malý model Země určuje místo, kde se objeví model Země. Lze ho vzít rukou a umístit na jiné místo.

Odepne malý model Země od menu. Nebude se tak pohybovat společně s uživatelovou levou rukou.
Opětovným zmáčknutím se malý model Země opět připne na původní místo.

Smaže model Země i Měsíce a ponechá změny.

Přepíná mezi aktuální ruční kalibrací a původní kalibrací.

Uloží lokální kalibrační změny jednotlivých skenů.

Načte lokální kalibrační změny jednotlivých skenů.

Calibration Menu

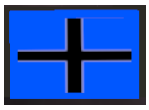
Calibration Menu je druhým způsobem, jak manuálně upravit polohu 3D sítě.

Posune 3D síť k Acer ovladači.

Připne 3D síť na Acer ovladač.

Edit Menu

Edit Menu slouží k tvorbě a editaci scénáře.



Otevře Catalog.

Při aktivaci (značí červená barva) uživatel dotykem maže objekty.

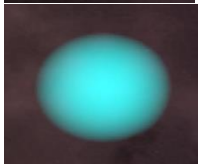
Uloží aktuální scénář.

Načte aktuálně uložený scénář.

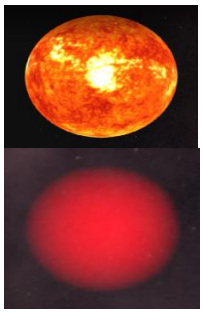
Přenese uživatele do vesmíru, kde může začít plnit vytvořený scénář.

Catalog

Instancuje první body cesty a otevře menu pro tvorbu cesty.



Instancuje vypínač a otevře menu pro vypínač.



Instancuje překážku a otevře menu pro překážku.

Instancuje kritický bod.
Kritický bod vyznačuje místo, kde hrozí srážka uživatele s fyzickým světem. V blízkosti tohoto bodu je uživateli zobrazen naskenovaný model prostředí.

Path Making GUI

Path Making GUI je menu pro tvorbu cesty. Cesta se skládá bodů, které jsou propojeny tenkou linií.

Přidá bod cesty.

Odebere poslední bod cesty.

Zruší cestu.

Výchozí chování bodů je to, že se snesou k zemi,kde začnou levitovat. Tímto tlačítkem se toto chování dá přepnout. Body se pak nebudou snášet k zemi.

Target GUI

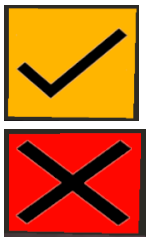
Target GUI slouží zejména k přiřazení vypínače k cestě. Společně s otevřením tohoto menu se ve scéně objeví vypínač v podobě modře blikající hvězdy. Vypínač slouží k aktivaci cesty. Aktivovaná cesta problikává ve směru, ve kterém ji má uživatel následovat. Uživatel přiřadí vypínač k cestě dotykem bodu patřícího pod zvolenou cestu.

Zruší vypínač.

Obstacle GUI

Obstacle GUI slouží zejména ke změně velikosti aktuální překážky.

Instancuje malý model Země. Jeho posouváním se mění velikost překážky.



Smaže malý model Země, a tak je možné překážku přesunout bez změny její velikosti.

Zruší překážku.

Veškeré ikony jsem získal stránce *flaticon.com* z balíku ikon od společnosti *Google*.

Příloha B

Postup pro sestavení aplikace

Pro sestavení aplikace (build) je nutné projekt spustit v Unity. Následně do Unity projektu importovat následující balíčky:

Leap_Motion_Core_Assets_ 4.4.0.unitypackage (dostupné na stránkách *leapmotion.com*)
Leap_Motion_Interaction_Engine_1.2.0.unitypack (dostupné na stránkách *leapmotion.com*)
ZEDUnityPlugin_v2.7.0_b2.unitypackage (dostupné na stránkách *stereolabs.com*)

Balíčky do Unity naimportujete: Assets – Import Package – Custom Package

Ze stránek *leapmotion.com* je nutné si stáhnout *Leap_Motion_Setup_4.0.0.exe* a nainstalovat SDK. Dále je nutné z Unity Assets Store stáhnout Steam VR Plugin a ten importovat do unity.

Pro sestavení aplikace otevřete Build Settings: File – Build Settings. Pro sestavení aplikace využívající kameru ZED mini zvolte v sekci Scenes In Build scénu ZED_FINAL, pro sestavení aplikace využívající Acer Windows Mixed Reality headset zvolte ACER_FINAL_STEAM. Následně potvrďte stisknutím tlačítka Build.

Příloha C

Postup při spuštění aplikace

Pro spuštění obou aplikací je nutné ze stránek *leapmotion.com* stáhnout *Leap_Motion_Setup_4.0.0.exe* a nainstalovat SDK. Následně nainstalovat aplikaci PiPlay ze stránek *pimaxvr.com*. Dále je nutné mít na počítači nainstalovanou službu Steam a pomocí služby Steam si stáhnout aplikaci Windows Mixed Reality for SteamVR.