Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Pick by Vision navigace ve skladových prostorách

Plzeň 2017 Jakub Zíka

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 28. června 2017

Jakub Zíka

Abstract

This diploma thesis examines the areas of hardware and software whose features enable the implementation of applications that can handle navigation in warehouses. The result is a demo application that addresses the creation and management of navigation maps. The navigation map is represented by an unoriented graph. The path calculation is implemented by the Dijkstra algorithm. Navigation solves the problem of localization in the environment based on the previously created map. The application also provides the visualization of real-time motion in a navigable environment.

Abstrakt

Předkládaná diplomová práce se zabývá zkoumáním oblastí hardwaru a softwaru, jejichž vlastnosti umožní implementaci aplikace, která zvládne navigaci ve skladových prostorách. Výsledkem práce je aplikační demo, které řeší vytvoření a správu navigačních map. Navigační mapa je reprezentována neorientovaným grafem. Výpočet cesty je realizován Dijkstrovým algoritmem. Navigace řeší problém lokalizace v prostředí na základě předchozí vytvořené mapy. Aplikace také řeší vizualizaci real-time pohybu v navigovaném prostředí.

Obsah

1 Úvod

Ve skladovém prostředí se firmy snaží být co nejefektivnější, jak ve směru organizace položek skladu, tak ve směru skladových procesů. Organizaci položek ve skladu řeší WMS (*Warehouse Management system*). Řízení skladových procesů se ale optimalizuje o něco hůře. Jedním z těchto procesů je i navigace ve skladových prostorách na konkrétní místo či ke konkrétní skladové pozici. WMS systém sice skladníkovi řekne, na jaké pozici se položka nachází a skladník může vědět, jak se na dané místo dostat, ale již například neví, zda je zvolená trasa ta nejkratší.

Hlavním cílem práce je prozkoumání hardwarových a softwarových možností, které umožňují vytvoření a správu navigačních map vnitřního prostředí. Úkolem je navržení reprezentace a výpočtu trasy mezi dvěma body v 3D prostoru s ohledem na pohyb objektu nezanedbatelné velikosti. Ze získaných informací bude navržena vhodná datová struktura dat pro efektivní algoritmické řešení hledání nejkratší cesty navigace ve skladovém prostředí. Výsledkem práce tedy bude jednoduché demo, které zvládne skenování, zpracování a správu vnitřního prostředí. Bude umožněn i náhled z jiné perspektivy aby bylo možné oblast lépe zkontrolovat a případně upravit.

Samotný text bude členěn do sedmi kapitol. Druhá kapitola se zabývá průzkumem oblastí hardwaru a softwaru na získání a zpracování 3D modelu vnitřního prostředí. Ve třetí kapitole jsou uvedeny datové struktury reprezentace navigačního grafu a algoritmy na jeho procházení. Čtvrtá kapitola popisuje navrhované řešení a obsahuje všechny podstatné matematické vzorce pro práci s grafikou, které dosud nebyly v textu zmíněny. Pátá kapitola obsahuje popis implementace navrhovaného řešení. Šestá kapitola obsahuje výsledky testování aplikace za různých podmínek okolí (osvětlení, přemisťování různě velkých objektů, atd.). Poslední sedmá kapitola je zhodnocením celé práce.

2 Získání a zpracování 3D modelu vnitřního prostředí

V této kapitole bude popsán software a hardware pro získání a zpracování 3D modelu vnitřního prostředí.

2.1 Hardware

Pomocí hardwarových zařízení je možné získávat prostorové souřadnice, ty převést na mračno bodů a to následně na polygonální síť nebo jiný matematický model. Modely lze poté využít v inženýrských aplikacích (tzv. Rapid prototyping, reverse engineering), ve zdravotnictví (plastická chirurgie, ortopedie, výzkum) nebo v 3D vizualizačních softwarech (rendering, vizualizace, animace). Skenery se mezi sebou liší v metodách snímání dat a prací s nimi. Základní rozdělení skenerů je na kontaktní a bezkontaktní. Předmětem skenování budou skladové prostory, dále se tedy bude hovořit pouze o skenerech bezkontaktních, čili takových, které nepotřebují přímý kontakt se skenovanou plochou či předmětem. [27], [2]

2.1.1 Rozdělení podle technologie snímání

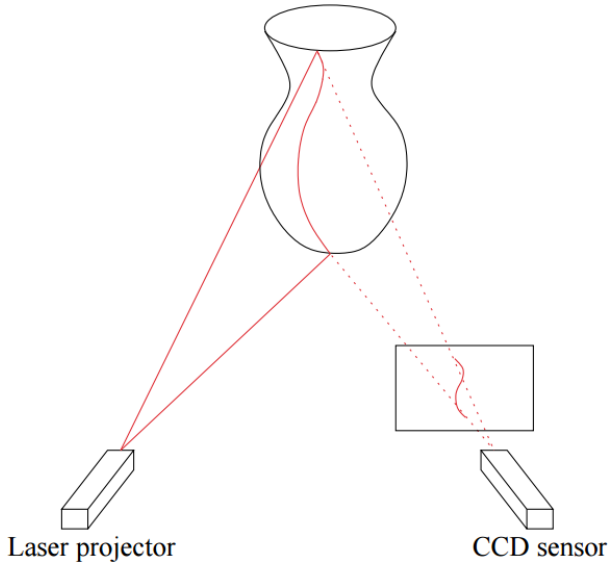
Bezkontaktní skenery vysílají určitý druh elektromagnetického vlnění a detekují jeho odraz nebo záření procházející skrze objekty v tom pořadí, jak jsou nacházeny v prostředí. Různé typy záření používají světlo, ultrazvuk nebo rentgen.

Metoda time-of-flight

Skenery využívající světlo ke skenování okolí mohou používat k výpočtu vzdálenosti předmětu například dobu letu světla. Tato zařízení vysílají krátké pulsy světla a ohodnocují pak vzdálenost na základě času navrácení odraženého světla. Tyto systémy byly vyvinuty pro *real-time* ohodnocení a mohou být využity pro velké vzdálenosti (až 100 metrů). Systémy *time-of-flight* vyžadují vysokou přesnost měření času odrazu. Tento typ skenu umožňuje ohodnotit 10 000 $\sim$ 100 000 vzdálenostních bodů za sekundu. [7], [28]

Metoda laserová triangulace

Alternativní třídou této oblasti je systém *triangulace*. Osvětlovací systém zobrazí rozkmitávaný vzor světla na předmětu, který má být skenován, viz obrázek 2.1 a zároveň je snímán přidruženou kamerou. Snímač, kterým je často CCD kamera, snímá odražené světlo od objektu. Kamera je umístěna ve známé poloze vůči skeneru a lze tedy pomocí trigonometrie vypočítat 3D prostorové (XYZ) souřadnice bodů povrchu. Kamera zaznamenává průměty laseru na povrch a digitalizuje všechny body z laserové čáry. Největší vliv



Obrázek 2.1: Ukázka systému triangulace.[7]

na výsledek skenování mají vlastnosti skenovaného materiálu. Triangulační skenery mohou podávat špatný výkon pro materiály, které jsou lesklé, mají nízké albedo[1] nebo mají významný podpovrchový rozptyl[2]. [7]

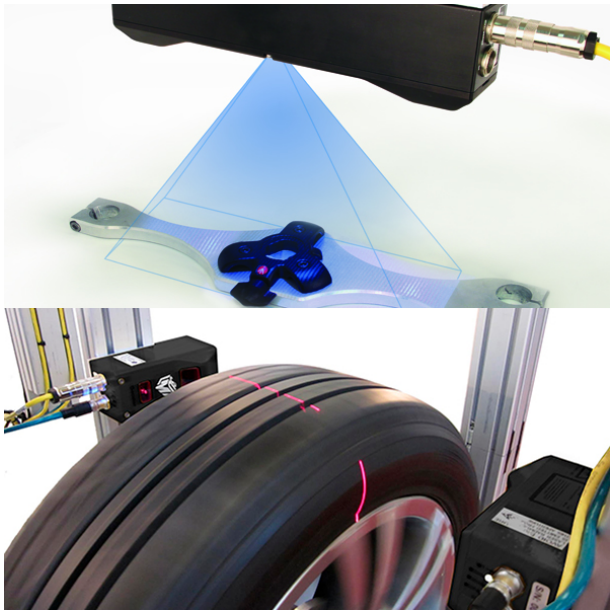
Metoda structured light

Posledním typem skenování je promítat vzor světla na subjekt a sledovat deformaci vzoru na subjektu. Tento typ skenování je technicky velmi podobný systému triangulace. Zde je ovšem laser nahrazen strukturovaným světlem a

[1] Schopnost odrážet záření.

[2] Tento pojem se používá hlavně ve hrách při programování grafické simulace kůže či rostlin. Paprsek se zde, po kontaktu s materiálem, neodrazí ihned, ale chvíli putuje pod povrchem a teprve pak ho opustí. Takové vlastnosti má například kůže, vosk, organické povrchy nebo nejrůznější tekutiny.

zvětšením pozorované plochy. Systém triangulace snímá totiž v jeden časový okamžik pouze vzor daný laserem (bod nebo linie). Oproti tomu při použití strukturovaného světla je snímána v jeden časový okamžik celá pozorovaná plocha, což nám vrací mnohonásobně větší skenovanou plochu za jeden časový okamžik. Technologie je založena na tom, že vzor světla je zobrazován na subjektu buď za pomoci LCD projektoru nebo jiným stabilním zdrojem světla. Kamera, která je umístěna mírně od projektoru, se dívá na tvar vzoru světla a vypočítává vzdálenost každého bodu v mřížce, která slouží v pozorovaném obraze jako oddělovač mezi jednotlivými body. Tato technologie skenování je stále v aktivním vývoji a používá se v mnoha výzkumech, ale i komerčních systémech.



Obrázek 2.2: Strukturované světlo vs laserová triangulace.[21]

Silné a slabé stránky

Při skenování jakýmkoliv systémem je důležité mít čistý pohled na scénu, která má být zaznamenána. Co senzor nevidí, je schované za překážkou, to nenaskenuje. Time-of-flight a triangulační dálkoměry mají oba své silné a slabé stránky, které je předurčují pro využití v rozdílných situacích. Výhodou time-of-flight skenerů je, že jsou schopné provozu na velmi dlouhé vzdálenosti v řádu kilometrů. Skenery jsou tak vhodné pro skenování velkých staveb jako jsou budovy nebo geografické prvky. Nevýhodou time-of-flight dálkoměrů je

jejich přesnost na velké vzdálenosti. Vzhledem k vysoké rychlosti světla je ovšem tato nepřesnost často zanedbatelná a pohybuje se v řádu milimetrů.

Triangulační dálkoměry jsou přesným opakem. Mají omezený dosah na několik metrů, ale jejich přesnost je poměrně vysoká. Přesnost triangulačních dálkoměrů se pohybuje v řádu desítek mikrometrů. Při skenování 10 000 bodů za sekundu mohou skeny s nízkým rozlišením trvat méně než sekundu, ale skeny s vysokým rozlišením, které vyžadují miliony vzorků, mohou přivodit problémy. Vzhledem k tomu, že zachycení odrazu trvá delší dobu, přináší tato metoda problém při skenování v kombinaci s pohybem. Zařízení prakticky musí zůstat stát nehybně na místě, aby se nenarušila skenovaná data.

Problémem obou předchozích metod je malá snímaná plocha při skenování. Pro triangulaci navíc platí, že skenování probíhá dlouho, je náchylné na otřesy a jakýkoliv větší pohyb. Oproti tomu výhodou strukturovaného světla je větší snímaná plocha. Většina mobilních zařízení využívající strukturované světlo, je vybavena gyroskopem a akcelerometrem, čímž dokáží částečně eliminovat zkreslení pohybem, a zároveň tak tvořit komplexnější mračna bodů.

2.1.2 Rozdělení podle mobility zařízení

Po rozdělení skenovacích zařízení do kategorií podle systému sběru dat lze skenery rozdělit také do kategorií podle jejich mobility.

Stacionární skenovací zařízení

Stacionární skenery jsou zařízení, která jsou pevně umístěna na své pozici. Mohou skenovat buď pouze prostor před sebou do určité šířky zorného pole nebo mají otočnou hlavu a dokáží zaznamenat celé okolí v 360°.

LIDAR. Jedním z takovýchto zařízení je LIDAR. Toto zařízení využívá technologii time-of-flight. Ve velkém měřítku se používá pro skenování území, kdy je senzor umístěný na spodku letadla či dronu.[8] LIDAR se ale vyrábí i jako pevný podstavec s otočnou hlavou. Konkrétním zařízením tohoto druhu je například *REO5* a *REO8* od firmy *Ocular Robotics*. Zařízení umožňují plné skenování horizontálně 360°a vertikálně 70°. Přístroj jako takový nedisponuje žádným výpočetním výkonem. Pro zobrazení dat je nutné připojit

REO k počítači přes Ethernet a příslušným softwarem zobrazit a zpracovat skenovaná data. [24]

Ocular REO5 3D LIDAR Scanner	
Dosah	30 m
Přesnost	± 50 nm
Rozsah pohledu	360°x 70°
Váha	2,8 / 3,0 kg

Tabulka 2.1: Specifikace Ocular REO5

LIDAR technologie se v této době také využívá v automobilovém průmyslu pro autonomní řízení. Například auto od firmy Google, které samo bez pomoci člověka jezdí po USA, využívá tuto technologii ke skenování okolí aby mohlo reagovat na jeho změny.

Microsoft Kinect	
Dosah	4 m
Rozsah pohledu	43°x 57°
Rozlišení IR camera	1280x960 px
Rozlišení Color camera	1280x960 px
Přesnost (prostorová)	3 mm
Přesnost (hloubková)	1 cm
Snímkování	30 FPS, 27 MHz
Raw data	color map, IR map, depth map
RAM	64 MB

Tabulka 2.2: Specifikace Microsoft Kinect

Microsoft Kinect. Stacionárním zařízením založeném na technologii strukturovaného světla je *Microsoft Kinect*. Přístroj využívá RGB kameru s rozlišením 1280x960. Dále má IR vysílač a IR hloubkový senzor. IR vysílač vysílá světelné signály a hloubkový senzor je čte. Zařízení také disponuje akcelerometrem pro 3 stupně volnosti konfigurovaným pro zatížení v rozsahu 2G. Stejně jako REO5 ani Kinect nedisponuje žádným výpočetním výkonem, proto musí být zařízení neustále připojeno k počítači. Připojení je přes USB 2.0. Jeho úhel skenování okolí je mnohem menší, je tu ovšem již zmíněná podpora akcelerometru, ale bohužel jen ve 3 stupních volnosti. To znamená,

že nelze využít rotace, tím se skenování prostředí za pohybu zařízení ztěžuje. Propojení mračen bodů by se muselo doprogramovat nebo by byl použit specializovaný software.[20]

Gocator 1300 Series. Skenovací zařízení založené na laserové triangulaci je například *Gocator* od firmy *LMI Technologies*. Senzor se například využívá k zachycení různých profilů povrchu. Dá se díky tomu například měřit kvalita výrobku podle velikosti odchylky. Zařízení se využívá v automobilovém průmyslu, elektronice nebo k profilování vozovky. Opět platí, že senzor musí být napevno umístěn. Nedisponuje žádnými pohybovými čidly. Senzor nemá ani otočnou část, aby dokázal snímat prostor kolem sebe v rozsahu 360°. Senzor musí být po celou dobu fungování připojený k počítači a ke zdroji napětí. Připojení k počítači je přes Gigabitový Ethernet. [9]

Gocator 1390 Series	
Rozlišení	0,025 mm
Rozsah skenu	2 m
Váha	0,7 kg
Snímkování	32 kHz

Tabulka 2.3: Specifikace Gocator 1390 Series

Všechny výše popsané systémy se shodují v tom, že jsou při použití vždy umístěné na pevném bodě, snímají okolí a reagují na jeho změny. To znamená, že při skenování místnosti by bylo nutné, například u Kinectu pro jeho malý rádius zachycení scény, vždy vytvořit snímek mračna bodů, ručně zařízení pootočit a udělat další snímek. Na skenované snímky bude tedy nutné použít algoritmy pro skládání mračen bodů do jedné komplexní scény. O tom bude ale více řečeno v kapitole 2.2.

Mobilní skenovací zařízení

Pro typ skenování, kdy je zařízením procházena celá scéna najednou, je lepší použít mobilní skenovací zařízení. Ta by měla být vybavena akcelerometrem a gyroskopem, aby bylo možné do jednotlivých mračen bodů zanášet i údaj o pohybu zařízení.

Google Yellowstone Tango. Takovým zařízením je například tablet od *Google ATAP* s technologií *Tango*, který dostal výrobní název Yellowstone. Zařízení obsahuje jak akcelerometr, tak i gyroskop, což zařízení umožňuje zaznamenávat pohyb v 6-ti stupních volnosti. Skener pracuje se strukturovaným světlem, takže dokáže oskenovat velkou plochu za jednotku času. Výhodou zařízení je, že nemusí být nijak připojené k žádné výpočetní technice. Dokáže mračna bodů zaznamenat, zpracovat i uložit do paměti, která má kapacitu 128 GB. Navíc je zařízení vybaveno displejem, na kterém lze pozorovat výsledky skenování, a dostat tak informaci, které části scény jsou nebo nejsou oskenované. Data je možné rovnou posílat do počítače přes Wifi nebo kabel. Tablet disponuje konektory Micro USB 2.0 a Micro HDMI. Pokud uživateli nestačí paměť tabletu, lze ji rozšířit paměťovými kartami microSD. Technologie je momentálně dostupná v komerční verzi pouze ve dvou zařízeních a těmi jsou *Lenovo Phab 2 Pro* a *Asus ZenFone AR*. Yellowstone je dostupný pouze v *Google Play* pro vývojáře. [12]

Google Yellowstone Tango	
Rozlišení	1920x1200 px
Váha	0,37 kg
Kamera	4MP (RGB-IR senzor)
Přesnost	2 µm
Úložiště	128 GB
RAM	4 GB
CPU	Nvidia Tegra K1
Velikost baterie	4960 mAh
Cena	16 000 Kč

Tabulka 2.4: Specifikace Yellowstone Tango

IIIDScan PrimeSense 1.08. Dalším zařízením využívajícím strukturované světlo je *IIIDScan PrimeSense 1.08*. Ten díky tripodu může sloužit stejně dobře jako stacionární, tak i jako mobilní skener. Jeho největší nevýhodou je konektivita. Zařízení musí být stále připojené k počítači přes USB 2.0 nebo 3.0. Na druhou stranu má výhodu v tom, že je cenově dostupný. Stejně jako Tango i IIIDScan PrimeSense obsahuje software pro zpracování vstupních dat. Dokáže tedy skládat mračna bodů dohromady. [13], [1]

IIIDScan PrimeSense 1.08	
Dosah	0,8 - 3,5 m
Rozsah pohledu	57,5°x 45°
Rozlišení 1	640x480 px
Přesnost	0,5 mm
Snímkování 1	30 FPS
Rozlišení 2	320x240 px
Snímkování 2	60 FPS
Cena	1 442 $

Tabulka 2.5: Specifikace IIIDScan PrimeSense 1.08

Artec Eva. Profesionálním nástrojem je *Artec Eva* skener. Ten je opět založený na strukturovaném světle a stejně jako PrimeSense vyžaduje při skenování přímé připojení k počítači přes USB 2.0 nebo 3.0. Artec ale nabízí možnost dokoupit externí baterii s kapacitou 16 000 mAh, která udrží zařízení v chodu až 6 hodin. Eva dokáže zachytit až 288 000 bodů za sekundu. Skener má výhodu oproti ostatním zařízením využívající strukturované světlo v tom, že dokáže velmi dobře skenovat tmavá a nebo lesklá místa. Takovou vlastnost levnější (neprofesionální) skenery postrádají. Skenery Artec také vynikají velice přesnými skeny s velkým množstvím detailů. [4] [1]

Artec Eva	
Dosah	0,4 - 1 m
Přesnost	0,1 mm
Snímkování	16 FPS
Váha	0,85 kg
Kamera	1,3 MP
Cena	13 700 euro

Tabulka 2.6: Specifikace Artec Eva

iSense Scanner. Zařízení vzhledově podobné Google Tango je *iSense Scanner*, který funguje jako doplněk k iPad Air, mini a 4th Generation. Ke skenování využívá stejně jako předešlé mobilní skenery strukturované světlo. Výhodou tohoto skeneru je přímé připojení na iPad. To znamená, že skener je iPadem napájen a zároveň mu poskytuje úložiště dat a výpočetní výkon. Navíc iPad Air obsahuje jak gyroskop, tak i akcelerometr. Bohužel cena sa-

motného skeneru bez iPadu je velmi podobná ceně Tanga. Cena iPadu se pohybuje okolo 10 000 Kč. [14], [15]

iSense Scanner	
Dosah	0,4 - 3,5 m
Přesnost	0,9 mm
Rozlišení	640x480
Snímkování	30 FPS
Rozsah pohledu	58°x 45°
Váha	99,2 g
Výdrž	3 - 4 hodiny
Cena	17 399 Kč

Tabulka 2.7: Specifikace iSense skeneru

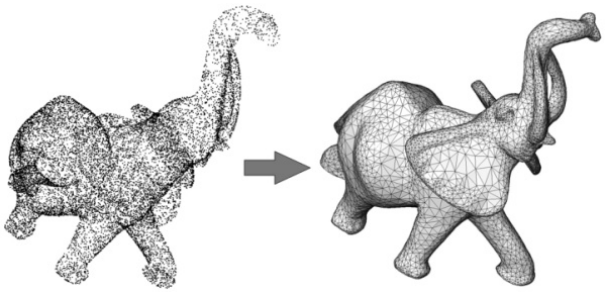
2.1.3 Zhodnocení HW

Pro toto zadání je důležitá mobilita a jednoduchost skenovacího zařízení. Proto bude zvolen typ skenování pomocí strukturovaného světla, protože je tento systém přesný, stále ve vývoji (zdokonaluje se), dokáže reagovat na pohyb a snímá velkou plochu za jednotku času. Ostatní metody ztrácí na využitelnosti buď tím, že nedokáží reagovat na pohyb (time-of-flight) nebo tím, že za jednotku času skenují malou plochu (laserová triangulace). Z vybraného hardwaru by nejvíce těmto kritériím vyhovovala zařízení Google Yellowstone Tango a iSense s iPadem. Přidanou hodnotou je možnost vidět sběr dat přímo na displeji zařízení. Tato dvě zařízení navíc vítězí i svou nízkou cenou.

2.2 Software

Po naskenování okolí se získá mračno bodů (*Point Cloud*), které je nutné dále zpracovat. Některé přístroje dokáží data zpracovávat během skenování, jiné potřebují předat data do počítače, kde následně probíhá převod na grafický (polygonová síť neboli *mesh*) nebo jiný popis okolí. Většina programů využívá rekonstrukci povrchu z mračna bodů, která vygeneruje mesh. Ten je pak možné uložit do formátu `OBJ` nebo `PLY`, které umožňují přidání dalších

informací, např. o barvě jednotlivých ploch. Jiný software z mračna bodů vyjme pouze specifické body nebo shluky bodů a vytvoří z nich matematický popis okolí, který lze pak uložit například do formátu ADF (*Area Description File*).



Obrázek 2.3: Ukázka převedeného mračna bodů na polygonovou síť.[3]

2.2.1 Rekonstrukce povrchu z mračna bodů

Při digitalizaci prostředí je naším cílem modelovat, rozpoznávat a analyzovat okolní svět. Metoda rekonstrukce povrchu slouží k získání informací o okolí, kde základem je získat mračno bodů popisující reálné prostředí a zrekonstruovat z něj co nejvíce informací, což znamená vymodelovat tvarem skutečnosti odpovídající mesh, viz obrázek 2.3. Metody a hardware pro získání mračna bodů již byly popsány v kapitole 2.1. Rekonstrukci lze popsat ve 4 částech, kterými jsou *Artefakty mračna bodů*, *Vstupní požadavky*, *Třída tvaru* a *Výstup z rekonstrukce*. [6]

Artefakty mračna bodů

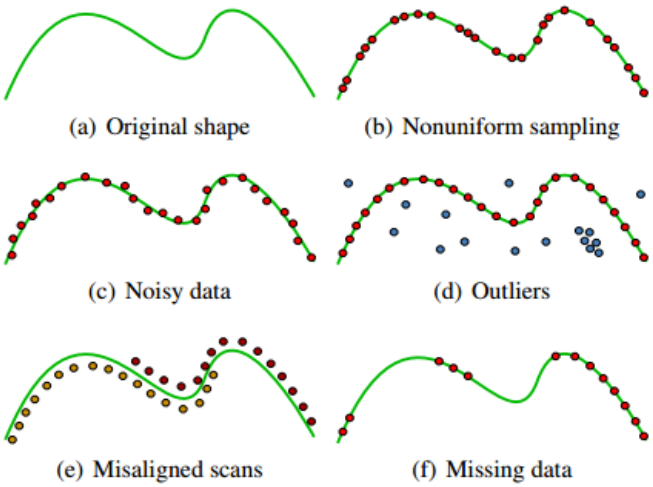
Vlastnosti mračna bodů jsou důležitým faktorem, který ovlivňuje chování metod rekonstrukce. Těmi nejdůležitějšími jsou: hustota vzorkování (*sampling density*), šum (*noise*), odlehlé hodnoty (*outliers*), chyba zarovnání (*misalignment*) a chybějící údaje (*missing data*) viz obrázek 2.4.

Vstupní požadavky

Různé metody se liší různými vstupními požadavky na mračna bodů. Minimum, které platí pro všechny algoritmy, je množina 3D bodů, které reprezentují rekonstruovaný povrch. Každá informace navíc pak vede k přesnější rekonstrukci. Dodatečnými informacemi jsou například data o normálách,

informace o skeneru a barevnosti (RGB) snímků, atd.

Povrchové normály jsou informace o kolmici k povrchu pro každý bod vstupního mračna bodů a významně ovlivňují výsledný povrch. Normála může být *orientovaná* nebo *neorientovaná*. Pokud informace o normále chybí, její výpočet může být mnohdy velice náročný.



Obrázek 2.4: Vlastnosti mračna bodů.[6]

Neorientované normály nemají informaci o směru. Vypočítají se obvykle přímo z mračna bodů. Pomocí informací o skeneru lze pak dopočítat jejich orientaci. Jednou z nejpopulárnějších metod pro výpočet normály je analýza hlavních komponent (PCA[3]). Neorientované normály mohou sloužit k určení rovinných oblastí v mračnu bodů, projekci bodu na aproximovaném povrchu, konstrukci neznaménkového pole vzdáleností nebo výpočtu kovariančí matice.

Orientované normály mají konzistentní orientaci. A to buď dovnitř nebo ven z povrchu. Při existenci orientace normál lze vytvořit znaménkové pole vzdáleností. Negativní hodnota bude indikovat interiér, pozitivní hodnota pak exteriér a nula bude reprezentovat povrch. Existuje mnoho způsobů, jak vypočítat orientaci normály. Jeden z možných postupů je popsán ve zdroji [23]. Jiné metody tento postup generalizují použitím implicitního pole a indikační funkce, viz obrázek 2.6. Základní myšlenka ale zůstává stejná, pokusit

[3]Slouží k dekorelaci dat. Často se používá ke snížení dimenze dat s co nejmenší ztrátou informace.

se rekonstruovat interiér a exteriér.

Informace o skeneru, ze kterého bylo mračno bodů získáno, může poskytnout další užitečné informace. Lze například získat 2D mřížku odpovídající rozsahu záběru viz obrázek 2.5. Tato mřížka snímku může být použita k určení orientace normál. Lze ji také použít k odhadu hustoty vzorkování. Může být také použita k detekci outlierů, jejichž sousedé na skenovací mřížce jsou v mnohem větší vzdálenosti, než je hustota vzorkování. Při odlišování outlierů je však nutné být opatrný, náročnost této úlohy není triviální.

Metoda	**Artefakty mračna bodů**					**Vstupní požadavky**		**Třída tvaru**
	Nerovnoměrné vzorkování	Šum	Outliers	Chyba zarovnání	Chybějící data	Neorientované normály	Orientované normály	
RBF	Ano	Ne	Ne	Ne	Ano	Ne	Ano	Obecná
MPU	Ano	Ano	Ne	Ne	Ano	Ne	Ano	Obecná
Poisson	Ano	Super ano	Ano	Ano	Ano	Ne	Ano	Obecná

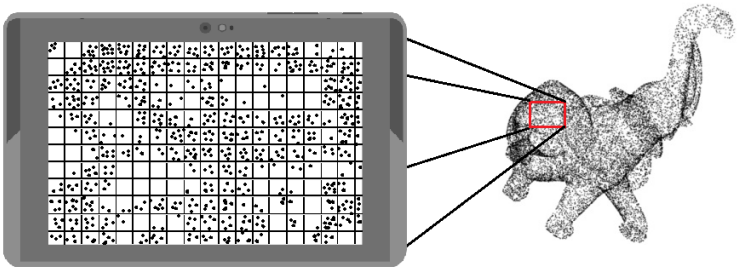
Tabulka 2.8: Srovnání metod rekonstrukce

Třída tvaru

Rekonstrukční algoritmy se mohou dále rozlišovat podle třídy tvarů. Tyto tvarové specifikace pomáhají k identifikaci jednotlivých artefaktů mračen bodů. Velmi často je rekonstrukce ze začátku řízena podle třídy tvarů. Základními třídami jsou: CAD modely, uměle vytvořené tvary, organické tvary, architektonické modely, městské prostředí a vnitřní prostředí.

CAD modely se obvykle skládají z jednodušších primitiv jako jsou roviny, válce a koule. Detekce takových instancí v mračnu bodů může být použita pro dopočítání chybějících dat v neúplných oblastech pro identifikovaná primitiva.

Umělé (syntetické) tvary často obsahují určité kanonické a geometrické vlastnosti jako například koplanární plochy, ortogonální plochy a plochy svírající úhly, které se opakují, a definují tak různé části tvarů.



Obrázek 2.5: Ilustrace pomocné 2D mřížky skeneru.

Organické tvary mají obvykle volnou strukturu a jsou často složeny z křivočarých prvků. Například stromy mají silnou strukturu kostry, kde uložení určitých jednodušších prvků lze posléze využít k rekonstrukci ztracených dat při skenovacím procesu. Jedná se zejména o malé větve či listy.

Architektonické modely patří mezi uměle vytvořené tvary. Tyto tvary jsou obvykle pravidelné. Předpokládáme-li, že skenovaný tvar patří mezi architektonické modely, lze problém regularizovat vytvořením předpokladů pro rekonstrukci fasád, manhattanské geometrie ulic a strukturované pravidelnosti.

Městské prostředí se často skládá z omezeného počtu typů objektů. Pro městské prostředí jsou vhodné datově řízené metody rekonstrukce. Lze zde předpokládat přítomnost půdy, budov, vegetace a dalších městských objektů.

Vnitřní prostředí. Typy tvarů vnitřního prostředí jsou spíše kombinace umělých a organických tvarů. Typickým faktorem vnitřního prostředí je, podobně jako u městského, existence malého počtu typů objektů, kde ale každý typ může být reprezentován více modely. Například v kancelářském

prostředí máme typy jako židle, stůl, tabule, ale každý zaměstnanec pak může mít jiný model židle nebo stolu. [6]

Shrnutí

Jelikož byla jako hardware ke skenování vybrána dvojice zařízení iSence a Google Tango, tak pro software z toho plyne, že data budou obohacena o informace o skeneru, a mračno bodů bude obsahovat orientované normály. Metodě by mělo být jedno, jaká třída tvaru bude skenována. Z toho podle srovnávací tabulky 2.8 ze zdroje [6] plyne, že bychom mohli použít metody RBF (*Radial Basis Function*), MPU (*Multi-level Partition of Unity*) nebo Poissonovskou rekonstrukci. Podle dalších vlastností metod ve srovnávací tabulce se jako nejlepší metoda jeví Poissonovská rekonstrukce díky velké robustnosti proti šumu, schopnosti se vypořádat s outliery, chybným zarovnáním, chybějícími daty a nerovnoměrnou hustotou vzorkování v mračnu bodů.

2.2.2 Poissonovská rekonstrukce

Jejím cílem je rekonstruovat uzavřený[4] (*watertight*) povrch aproximovaný indikační funkcí modelu a extrakcí isopovrchu. Klíčovým prvkem je tedy co nejpřesněji spočítat indikační funkci modelu. Výsledkem indikační funkce jsou indikační ukazatele. Gradient indikační funkce je vektorové pole, které je téměř celé nulové (indikační funkce je téměř všude konstantní), kromě bodů, které jsou blízko povrchu, kde se rovnají vnitřní povrchové normále. Tudíž orientované body mohou být brány jako gradient indikační funkce modelu.

Problémem ale je, jak indikační funkci sestavit. Je nutné tedy najít skalární funkci χ jejíž gradient co nejlépe aproximuje vektorové pole $\vec{V}$, to jest $min_{\chi}|| \bigtriangledown \chi - \vec{V}||$. Použitím operátoru divergence se problém transformuje do standardního Poissonovského problému. Což znamená, že výpočet skalární funkce χ, kde Laplacian[5] se rovná rozdílu vektorového pole $\vec{V}$, je $\Delta\chi \equiv \bigtriangledown \cdot \bigtriangledown\chi = \bigtriangledown \cdot \vec{V}$. Definice s postupem jsou upřesněny v kapitolách 3 a 4 zdroje [16].

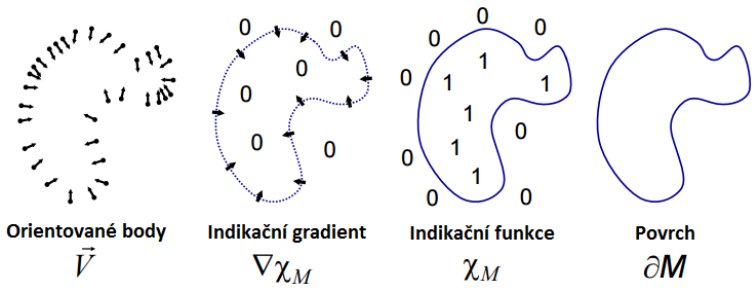
[4]Povrch je celistvý bez chybějících, nezrekonstruovaných částí.

[5]Odchylka gradientu.

Přeformulování rekonstrukce povrchu na Poissonovský problém nabídne mnoho výhod. Většina implicitních metod při rekonstrukci povrchu segmentuje data do regionů, které pak samostatně rekonstruuje. Opětovným spojením regionů vznikají ucelené zrekonstruované povrchy. Oproti tomu Poissonovská rekonstrukce pracuje se všemi daty najednou, aniž by používala jakékoliv rozdělení nebo následné spojování různých částí. Podobně jako RBF, tak i Poisson vytváří hladké povrchy i nad velmi zašumělými daty. Navíc mnoho implicitních metod zpracovává pro určitou hodnotu vždy jen její blízké okolí. Následná rekonstrukce pak obsahuje malé pospojované desky a povrch není hladký. U Poissonovské rekonstrukce tyto jevy nastávají jen zřídka, protože gradient implicitní funkce je vynucený na všech bodech v prostoru.

Postup rekonstrukce

Jsou dána vstupní data S jako množina prvků $s \in S$, kde každý prvek obsahuje bod $s.p$ a normálu $s.\vec{N}$. Indikační funkce je částečně konstantní, její explicitně počítané gradientní pole bude plné vektorů. Dále existuje těleso M



Obrázek 2.6: Ilustrace Poissonovi rekonstrukce ve 2D.[16]

s definovaným povrchem ∂M, kde χ_M je indikační funkce M. Dále $\vec{N}_{\partial M}(p)$ bude vnitřní povrchová normála pro $p \in \partial M$, $\tilde{F}(q)$ bude vyhlazovací filtr a $\tilde{F}_p(q) = \tilde{F}(q-p)$ je translace do bodu p. Gradient vyhlazené indikační funkce je roven vektorovému poli získanému vyhlazením pole s povrchovými normálami:

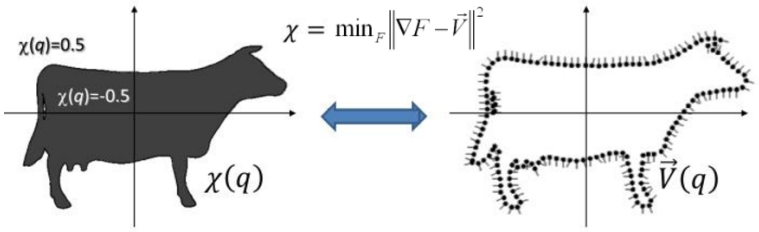
$$\nabla(\chi_M * \partial F)(q_0) = \int_{\partial M} \tilde{F}_p(q_0)\vec{N}_{\partial M}(p)dp. \tag{2.1}$$

Samozřejmě, že nelze ohodnotit povrch, dokud není znám tvar tělesa. Nicméně vstupní množina orientovaných bodů poskytuje dostatek informací pro aproximaci integrálu diskrétním součtem. Když se rozdělí množina bodů S na

části $P_S \subset \partial M$, lze aproximovat integrál přes část P_S hodnotou bodu $s.p$ škálovanou oblastí z části P_S:

$$\bigtriangledown(\chi_M * \tilde{F})(q) = \sum_{s \in S} \int_{P_S} \tilde{F}_p(q) \vec{N}_{\partial_M}(p) dp \approx \sum_{s \in S} |P_S| \tilde{F}_{s.p}(qPs.\vec{N} \equiv \vec{V}(q) \quad (2.2)$$

V praxi je potřeba dbát na výběr filtru. Měl by splňovat dvě pravidla. Na jednu stranu by měl být natolik přísný, aby data nevyhlazoval až příliš. Na druhou stranu by měl být natolik benevolentní, aby integrál přes P_S byl co nejlépe aproximován hodnotou $s.p$. Jakmile je k dispozici vektorové pole $\vec{V}$,



Obrázek 2.7: Dosednutí skalárního pole na gradientní.[18]

bude potřeba vyřešit funkci $\tilde{\chi}$ tak, že $\bigtriangledown\tilde{\chi} = \vec{V}$. Nicméně $\vec{V}$ není obecně integrovatelné, tím pádem exaktní řešení obecně neexistuje. Pro nalezení nejbližšího přibližného řešení metody nejmenších čtverců se použije operátor divergence pro vytvoření Poissonovy rovnice:

$$\triangle\tilde{\chi} = \bigtriangledown \cdot \vec{V}. \quad (2.3)$$

Více informací o implementaci v [16]. [16], [17], [18]

2.2.3 Popisy oblastí a lokalizace

Dalším druhem reprezentace prostředí je využití vlastností technologie Tango, kterými jsou nauka o okolí (*Area Learning*), sledování pohybu (*Motion Tracking*) a měření vzdáleností od objektů (*Depth Perception*). Konkrétně Area Learning využívá matematický popis prostředí, který lze uložit do souboru typu ADF.

Area Learning pracuje stejně jako rekonstrukce s mračny bodů. Z jednotlivých snímků, které jsou zpracovávány v reálném čase, vybírá klíčové body (*key-points*), které unikátním způsobem popisují skenované prostředí. Těmito prvky jsou například hrany zdí, rohy různých předmětů, atd. Díky tomuto popisu prostředí se zařízení dokáže po čase znovu lokalizovat. Jakmile

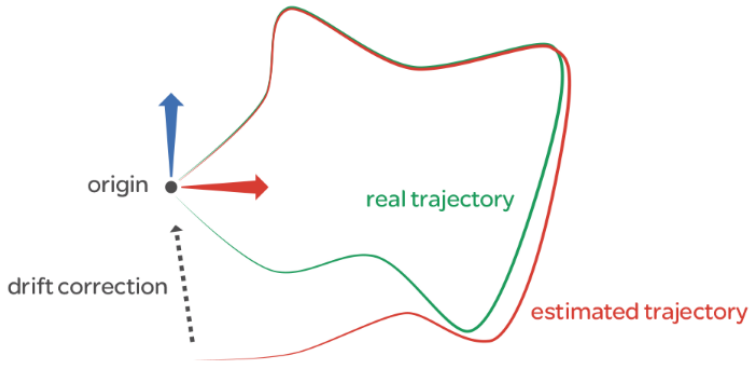
se Tango naučí nějakou oblast, má dvě možnosti, jak zdokonalovat stávající model pomocí informací získaných pouze z trekovacího senzoru.

- **Zlepšení přesnosti trajektorie pohybu** za použití korekce odchylky (*drift correction*).
- **Orientace a umístění v dříve naučené oblasti** za použití lokalizace.

Zlepšování trajektorie

Trekovací senzor v průběhu času kumuluje chybu, tím pádem se měření pohybu časem stává nepřesné. Zařízení opravuje některé chyby tak, že se orientuje na gravitaci. Na jiné chyby spojené s polohou samotný trekovací senzor nestačí.

Při zapnuté Area Learning si zařízení pamatuje vizuální vlastnosti oblasti, kterou navštívil, a zároveň ji používá k opravě chyb díky znalosti své pozice, orientace a pohybu. Tento model povoluje systému využít korekci odchylky (jindy také nazývanou jako *uzávěr smyčky*). Pokud zařízení vidí místo, které již bylo v relaci viděno, uvědomí si, že vykonalo uzavřenou smyčku a upraví svou cestu tak, aby byla více konzistentní vzhledem k předchozímu měření pohybu viz obrázek 2.8. Tyto korekce se používají k úpravě polohy a trajektorie v rámci aplikace. Když se zařízení začne pohybovat skenovanou oblastí,



Obrázek 2.8: Ukázka jak zařízení Tango provádí korekci odchylky.[10]

začnou se ve skutečnosti tvořit dvě rozdílné trajektorie. Jednou je skutečná cesta, kterou prochází uživatel se zařízením v ruce, a druhá je cesta odhadovaná zařízením, u které si zařízení myslí, že po ní uživatel kráčí. Na obrázku

2.8 zelená čára znázorňuje skutečnou cestu a červená čára znázorňuje cestu odhadovanou zařízením, na níž je vidět, jak se pomalu odklání od skutečné trajektorie. Jakmile se zařízení vrátí k původnímu místu odkud vyšlo, uvědomí si, že vidí známé místo a začne opravovat nakumulovanou chybu mezi oběma trajektoriemi tak, aby odhadovaná trajektorie co nejvíce odpovídala té skutečné.

Bez korekce odchylky se lze dostat do situace, kdy jsou pomocí AR (*Augmented Reality*) objektu vyznačené dveře, které jsou umístěné na pozici dveří reálných. Bez korekce se může stát, že virtuální dveře se mohou posunout z původní pozice tak, že jejich aktuální virtuální pozice nebude odpovídat dveřím v reálném světě.

Popisy oblastí a lokalizace

Poté, co je celá oblast naskenována, je možné data uložit do formátu ADF. Každá oblast v ADF má svůj jednoznačný identifikátor `UUID`. Uložení oblasti do ADF přináší mnoho výhod. Lze převzít souřadný systém z uložené oblasti a překrýt ho virtuálně vymodelovanou scénou. Při znovunačtení se pak virtuální scéna objeví vždy na stejném fyzickém místě.

Pro znovunačtení zmapované oblasti je ale nutné při spuštění aplikace provést lokalizaci. Jedná se o dvoufázový proces:

- Načíst uložený popis oblasti ve formátu ADF
- Projít část oblasti, která byla uložena do ADF

Jakmile zařízení uvidí, že je v oblasti, která je uložená v ADF, načte si všechna data originální pozice (tzn. bodu, ve kterém se původně při skenování nacházel) a tím se lokalizuje. Bez uložení popisu oblasti do ADF je vždy výsledná session s daty o skenování a startovací pozici ztracena. [10]

2.2.4 Zhodnocení SW

Po vypsání vlastností jednotlivých technik bude sestaven průřez výhodami a nevýhodami jednotlivých modelů reprezentující určité skenované oblasti.

Výhody a nevýhody rekonstrukce

Mezi velké výhody rekonstrukce lze zařadit grafickou reprezentaci celé skenované oblasti. Model oblasti je možné dále použít i k jiným účelům, jako je například rozpoznávání objektů ve scéně. Další výhodou grafického modelu je fakt, že malé změny, jako například přesun malých věcí, které z pohledu překážky v průchodu jsou zanedbatelné, modelu nevadí. Na druhou stranu při velké změně je nutné celý prostor přeskenovat. Další problém přichází s otázkou, jak exaktně rozpoznávat různá místa a pracovat s jejich polohou. To sebou nese i problém trekování pohybu po oblasti a určení kalibračního[6] bodu pro start aplikace, jelikož v tomto případě neexistuje nic jako lokalizace v prostředí.

Výhody a nevýhody ADF

Pokud se zařízení nachází v oblasti, která je málo rozmanitá, to znamená, že je místnost úplně holá a bez nábytku, je učení velmi obtížné, protože metoda nemá moc informací k zapamatování. Lokalizace je nejvíce úspěšná v době, kdy je prováděna v podmínkách, které se nejvíce podobají těm, ve kterých bylo prováděno skenování. To znamená, že přesuny nábytku nebo úplně jiné podmínky osvícení oblasti kombinované s pozorováním ze špatného úhlu mohou zapříčinit fakt, že lokalizace nebude úspěšná. Protože se prostředí může a bude měnit, je řešením vytvořit například více souborů ADF a uživatel si poté vždy načte ten, který nejvíce odpovídá jeho situaci. Je také možné připojit více naskenovaných sessions do jednoho ADF souboru, aby se získalo více popisů oblasti z více míst a úhlů za různých podmínek.

Asi největší nevýhodou metody používající Area Learning je fakt, že nikde není popsáno, co přesně za informace ADF uchovává. ADF je možné zpracovat a přistupovat k jeho datům pouze přes Tango API. Což přináší jistá omezení v oblasti použitelného hardwaru i softwaru.

[6] Výchozí pozice při startu aplikace. Z ní bude muset uživatel po každém startu aplikace vycházet.

2.3 Výběr HW a SW

Jako zařízení vhodná k získání a zpracování modelu vnitřního prostředí byla vybrána zařízení Google Tango a iSense v kombinaci s iPadem. Obě zařízení podporují snímání okolí pomocí mračen bodů. Je tedy možné na výstup ze zařízení, který popisuje scénu pomocí mračna bodů, aplikovat Poissonovskou rekonstrukci. U zařízení Tango je dokonce možné implementovat Poissonovskou rekonstrukci uvnitř zařízení. Lze využít knihovnu PCL[7], která podporuje vývoj v jazyce *C++* a poskytuje Poissonovu rekonstrukci jako nativní funkci, přes *JNI*[8].

Zvolením ADF jako vhodného formátu pro uchování reprezentace skenované oblasti zůstává z hardwaru pouze zařízení Google Tango, které jako jediné dokáže tento formát vytvořit, zpracovat a používat jej. Momentálně existují dvě alternativní zařízení disponující touto technologií. Jsou jimi již zmíněné Lenovo Phab 2 Pro a Asus Zenfone AR.

[7]Point Cloud Library - knihovna poskytující nástroje pro práci s mračny bodů.

[8]Java Native Interface - rozhraní přes které lze propojit kód jazyka *Java* s kódem psaném v C++.

3 Trasy mezi dvěma body v 3D modelu

Pro navigaci je také důležité vybrat správný algoritmus pro výpočet trasy mezi dvěma body. Nejprve bude navržena struktura reprezentace cest v prostředí, a určí se tak reprezentace jak výchozí pozice zařízení, tak pozice cílová, na kterou bude zařízení navigováno. Jelikož byla pro reprezentaci prostředí vybrána matematická struktura ADF, je nutné zvolit, jakým způsobem budou reprezentovány prvky navigace jako je uživatel, trasy a cílové navigační body.

3.1 Datové struktury

Jak již bylo zmíněno v kapitole 2.2.3, lze k souboru ADF připojit `XML` definici AR objektů s přidruženou informací o 3D pozici v matematickém modelu. Těmito objekty je následně možné přímo reprezentovat konkrétní pozice v 3D prostředí nebo jimi označit pro navigaci důležité elementy a jejich následným zpracováním nechat například vygenerovat jinou datovou strukturu, která umožní jednodušší reprezentaci prostředí i následnou navigaci.

3.1.1 AR navigační bod

Pokud AR objekty budou použity jako markery konkrétních pozic v 3D prostoru, je důležité stanovit vlastnosti těchto markerů a určit základní pravidla pro jejich umisťování. Na začátku tedy budou stanoveny základní typy markerů, které by se mohly v prostředí vyskytovat:

- **Křižovatka** - pozice reprezentující rozcestník
- **DB pozice** - pozice spojená s informacemi z databáze
- **Průchod** - pozice reprezentující zúžená místa jako dveře, vrata, atd.
- **Stoupání** - označení schodiště vedoucího vzhůru
- **Klesání** - označení schodiště vedoucího dolů

Křižovatka

Křižovatka reprezentuje místo, ze kterého lze odbočit do více směrů. Slouží jako univerzální spojnice grafu.

DB pozice

Databázová pozice reprezentuje místo, které lze propojit s databází, která o daném místě může obsahovat další informace. Markerem tohoto typu lze stanovit význam místnosti, data v databázi pak budou obsahovat podrobnější informace. Dále je možné markerem označit skladovou pozici, která bude mít v databázi například informace o uloženém zboží.

Průchod

Markerem typu průchod je možné označit zúžená místa, která mají vliv na průchodnost různě velkých objektů. Lze tedy označit dveře, vrata a další podobné průchody. Díky této informaci je například možné předat informaci o tom, že tímto místem neprojde velký náklad nebo neprojede vysokozdvižný vozík.

Stoupání/klesání

Stoupáním nebo klesáním bude označené místo, které představuje například schodiště či nájezdovou rampu. Opět touto informací je možné zjistit průchodnost daným místem. Dále je také možné získat informaci o tom, že se mění podlaží a že bude nutné změnit navigační 2D graf.

3.1.2 Navigační graf

Všechny typy markerů reprezentují vrcholy navigačního grafu. Ten může být *orientovaný* nebo *neorientovaný*. Graf je možné reprezentovat *spojovým seznamem* nebo *maticí sousednosti*. Spojový seznam je realizován polem vrcholů, kde každý prvek obsahuje spojový seznam vrcholů, do kterých z daného vrcholu vede hrana. Jednotlivé položky seznamu navíc obsahují cenu za průchod hranou. Matice sousednosti je vyplněna celá a význam hodnot je následující:

- $hodnota = 0$ - hrana mezi vrcholy neexistuje
- $hodnota > 0$ - cena za průchod danou hranou

Orientovaný graf

Pro orientovaný graf platí, že lze definovat vztah mezi dvěma vrcholy třemi způsoby. Hrana tedy může být neprůchozí, průchozí jedním směrem nebo průchozí oběma směry. V tomto typu grafu hrozí uvíznutí ve vrcholu, ze kterého nevede odchozí hrana.

Neorientovaný graf

Pro neorientovaný graf platí, že vztah mezi vrcholy lze definovat pouze dvěma způsoby. Hrana mezi uzly je průchozí oběma směry nebo hrana mezi uzly neexistuje. V tomto typu grafu nehrozí uvíznutí, protože pokud jednou cestou do vrcholu přijdu, mohu ho tou samou i opustit.

Implementace grafu

V případě, že je *graf úplný*[1] s V vrcholy, je lepší použít matici sousednosti, protože počet prvků bude stejně jako u seznamu sousednosti V^2. Ovšem složitost přístupu k jednotlivým hranám mezi vrcholy bude pro matici $O(1)$ a pro seznam $O(V)$.

Pokud ale bude *graf planární*[2], pak podle *Eulerovy věty* platí:

1. $V + S = H + 2$
2. $|H| \leq 3|V| - 6$ (s obsahem trojúhelníků)
3. $|H| \leq 2|V| - 4$ (bez obsahu trojúhelníků)

kde V je počet vrcholů, S je počet ploch a H je počet hran. Druhá a třetí část věty určuje maximální počet hran pro určitý počet vrcholů. U rozsáhlého grafu lze předpokládat, že nemůže být úplný, protože by se hrany musely křížit. Planární graf tedy bude implementován pomocí seznamu sousednosti. [25]

V tomto případě bude každý marker zároveň i vrchol grafu. Cena přechodu mezi hranami bude vzdálenost jednotlivých vrcholů. Graf bude neorientovaný, protože v reálném prostředí se bude obtížně hledat situace, kdy je

[1] Z každého vrcholu vede hrana do všech vrcholů grafu.

[2] Lze ho nakreslit tak, že ve 2D se žádná z jeho hran nekříží s jinou hranou. Jindy označován také jako rovinný graf.

hrana průchozí pouze jedním směrem. Zobrazení grafu na zařízení by mohlo být realizováno přepínáním mezi jednotlivými patry, kde by se pro každé patro vygeneroval 2D navigační graf. Algoritmus, který vypočte nejkratší cestu z bodu A do bodu B, bude popsán v kapitole 3.2.

3.1.3 AR identifikátor elementu reálného prostředí

Jiným typem identifikace je možnost použít AR marker ne jako navigační bod, ale jako identifikátor elementů v prostředí. Princip je podobný označování věcí pomocí různých druhů samolepek. Tím se jednotlivé elementy rozdělí do kategorií podle typu samolepek. Lze pak přeměřit vzdálenosti mezi jednotlivými elementy a vygenerovat například CAD model prostředí, který by se dále zpracoval. Typy markerů jsou zde téměř stejné jako v předchozí kapitole.

- **Zeď** - marker označující zeď
- **DB pozice** - pozice s pojená s informacemi z databáze
- **Průchod** - pozice reprezentující zúžená místa jako dveře, vrata, atd.
- **Stoupání** - označení schodiště vedoucího vzhůru
- **Klesání** - označení schodiště vedoucího dolů

Zde propojením jednotlivých značek nevznikne navigační graf, ale 2D CAD model. Díky jednotlivým markerům je v modelu zavedena informace o průchodech, schodištích a dalších prvcích, díky kterým se dá, podobně jako v předchozí kapitole, přesněji určovat trasa výsledné cesty.

Nevýhodou této metody je fakt, že nejsou přímo stanovené navigační body jako v metodě popsané v kapitole 3.1.1. To sebou nese problém jak reprezentovat cestu z bodu A do bodu B. Na druhou stranu má tato metoda výhodu v reprezentaci prostředí. Ve 2D modelu je vidět rozloha jednotlivých místností.

3.1.4 Navigační mřížka

Řešením navigace může být navigační mřížka, což je struktura složená z malých geometrických 2D tvarů. Nejčastěji se využívá čtvercová, hexagonová

nebo trojúhelníková mřížka. Jelikož CAD modelu nese informace o tvaru a velikosti dané oblasti, lze na něj přiložit mřížku, čímž bude oblast rozdělena na malé části. Bude použita například čtvercová mřížka tak, že čtverec bude představovat vrchol grafu. Jednotlivé vrcholy budou spojeny vždy s nejbližšími sousedy tak, aby žádná z hran neprotínala hrany objektů v CAD modelu (spojnice nesmí vést skrze stěnu). Všechny hrany mají hodnotu 1 a jsou neorientované. Vznikne tedy ohodnocený, neorientovaný graf, jehož vlastnosti byly popsány již v kapitole 3.1.2. [22]

3.2 Výpočet trasy

Obě metody pro tvorbu navigační struktury mají stejnou výchozí pozici a tím je ohodnocený, neorientovaný graf. Lze tedy přejít k výpočtu trasy mezi dvěma vrcholy. Na to se nejvíce hodí grafové algoritmy. Tím nejzákladnějším a nejznámějším je *Breadth First Search* (BFS) neboli prohledávání do šířky.

3.2.1 Breadth First Search

Tento algoritmus prohledává stavový prostor rovnoměrně ve všech směrech. Algoritmus pracuje na velmi jednoduchém principu. Na začátku je určen startovací vrchol. Dále jsou všichni jeho sousedé přidáni do fronty s informací, z kterého vrcholu byly nalezeni. Po nalezení všech sousedů se vybere první prvek z fronty a postup s jeho sousedy se opakuje. Tvoří se tak strom cest tak, jak bylo postupně dosaženo vrcholů spolu s informacemi o jejich předchůdcích. Algoritmus nepoužívá žádnou heuristiku či analýzu cest. Nejkratší cesta je brána vždy ta, která má nejmenší počet vrcholů.

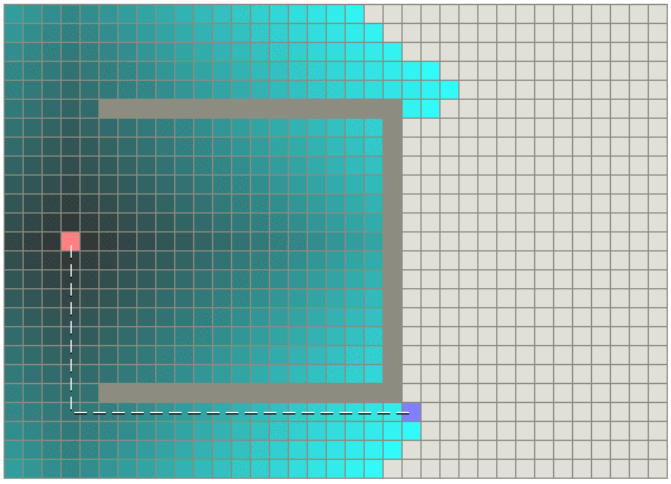
Pseudokód algoritmu

```
1 void BFS (Graph G, Node s) {
2   for (Node u in U(G)-s)
3     { stav[u] = FRESH; d[u] = nekonečno; p[u] = null; }
4   stav[s] = OPEN; d[s] = 0; p[s] = null;
5   Queue.Init(); Queue.Push(s);
6   while (!Queue.Empty()) {
7     u = Queue.Pop();
8     for (v in Adj[u]) {
```

```
 9        if (stav[v] == FRESH) {
10          stav[v] = OPEN; d[v] = d[u]+1;
11          p[v] = u; Queue.Push(v);
12        }
13      }
14      stav[u]=CLOSED;
15    }
16 }
```

Složitost

Složitost algoritmu je $O(|V|+|H|)$, kde V je množina vrcholů a H je množina hran grafu.



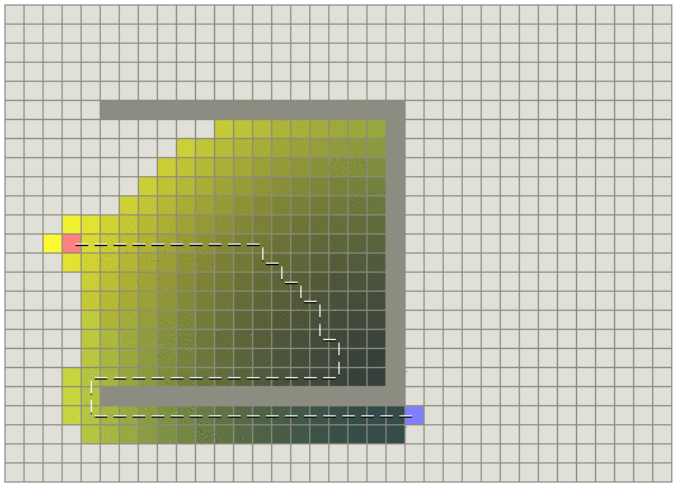
Obrázek 3.1: Dijkstrův algoritmus při prohledávání prostoru čtvercové mřížky.[22]

3.2.2 Dijkstrův algoritmus

Lehkou modifikací BFS je *Dijkstrův algoritmus*, který je nejrychlejší známý algoritmus pro nalezení všech nejkratších cest ze zadaného uzlu do ostatních uzlů grafu. Graf nesmí obsahovat hrany záporné délky. Vlna se zde oproti BFS nešíří na základě počtu hran, ale na základě vzdálenosti od zdroje. Tato vlna proto zpracovává jen ty uzly, k nimž byla nalezena nejkratší cesta.

Popis algoritmu

Je zadán graf G, v němž je potřeba najít nejkratší cestu. Existuje množina všech vrcholů V grafu G. Dále existuje množina H, která obsahuje všechny hrany grafu G. Algoritmus pracuje tak, že si pro každý vrchol v z V pamatuje délku nejkratší cesty, kterou se k němu dá dostat. Tato hodnota bude označena jako $d[v]$. Na začátku mají všechny vrcholy v hodnotu $d[v] = \infty$, kromě počátečního vrcholu s, který má $d[s] = 0$. Nekonečno symbolizuje, že neznáme cestu k vrcholu.



Obrázek 3.2: Greedy algoritmus při prohledávání prostoru čtvercové mřížky.[22]

Dále si algoritmus udržuje množiny Z a N, kde Z obsahuje už navštívené vrcholy a N dosud nenavštívené. Algoritmus pracuje v cyklu tak dlouho, dokud N není prázdná. V každém průchodu cyklu se přidá jeden vrchol v_{min} z N do Z, a to takový, který má nejmenší hodnotu $d[v]$ ze všech vrcholů v z N.

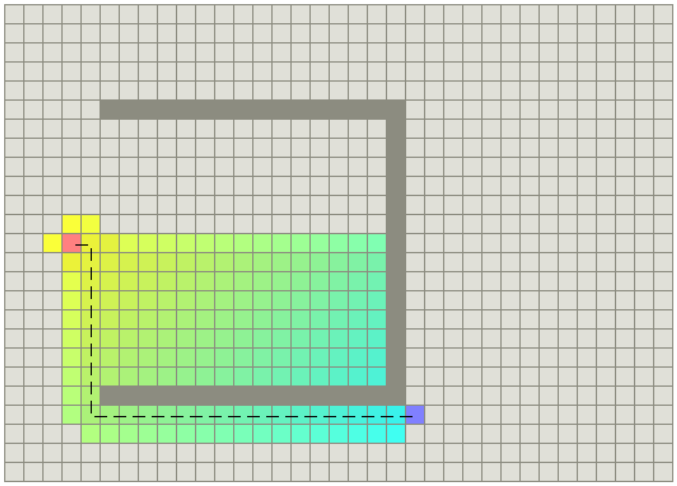
Pro každý vrchol u, do kterého vede hrana (označme její délku jako $l(v_{min}, u)$) z v_{min}, se provede následující operace: pokud $(d[v_{min}] + l(v_{min}, u)) < d[u]$, pak do $d[u]$ přiřaď hodnotu $d[v_{min}] + l(v_{min}, u)$, jinak neprováděj nic. Algoritmus končí ve chvíli, kdy pro hledaný vrchol v z V je délka jeho nejkratší cesty od počátečního vrcholu s uložena v $d[v]$.

Složitost

Složitost toho algoritmu záleží na implementaci prioritní fronty. V případě implementace pomocí sekvenčního vyhledávání bude složitost $O(|V^2|)$. V případě použití binární haldy to bude $O(|H| \log_2 |V|)$.

3.2.3 Best First Search algoritmus

Algoritmem založeném na podobném principu jako Dijkstra je *greedy BFS algoritmus*. Ten stejným způsobem objevuje nové vrcholy, které pak přidává do prioritní fronty, ve které prvky netřídí podle vzdálenosti, ale podle jejich slibnosti (podle odhadu do cíle). Metoda slibnosti je heuristická metoda, takže se řídí určitým druhem náhody. To znamená, že nám může GBFS najít stejnou cestu jako Dijkstra, ale rychleji. Na druhou stranu se také může stát, že sice najde cestu rychleji, ale za to delší. U Dijkstrova algoritmu je jistota, že vždy nalezne nejkratší cestu. Metoda slibnosti může být například implementována tak, že vrací velikost vzdušné čáry mezi aktuálním a cílovým vrcholem.



Obrázek 3.3: A* algoritmus při prohledávání prostoru čtvercové mřížky.[22]

3.2.4 A* algoritmus

Dijkstrův algoritmus funguje dobře pro nalezení nejkratší cesty, ale ztrácí čas prohledávání prostoru, který není slibný a nevede ke správnému výsledku. Greedy BFS naopak zkoumá slibné cesty, ale hrozí mu nenalezení

nejkratší cesty. Spojením těchto dvou výše popsaných algoritmů vznikne *A* algoritmus*. Ten využívá jak aktuální vzdálenost od počátečního vrcholu, tak odhadovanou vzdálenost k cíli.

Popis algoritmu

A* používá funkci $f(x)$, která ohodnocuje jednotlivé uzly pro určení pořadí v prioritní frontě. Funkce má tvar

$$f(x) = g(x) + h(x), \tag{3.1}$$

kde funkce $g(x)$ je funkce představující vzdálenost mezi počátečním a daným uzlem (Dijkstra), $h(x)$ pak představuje heuristickou funkci (greedy BFS). Ta odhaduje správnost postupu při vyhledávání optimální cesty za pomoci vzdálenosti z aktuálního uzlu do uzlu konečného. Zároveň musí být přípustná, tzn. nesmí nadhodnocovat vzdálenost k cíli. Například v navigaci může být použita jako heuristika vzdálenost vzdušnou čarou, jelikož je to fyzicky nejkratší možná cesta. Pokud heuristika h navíc splňuje podmínku

$$h(x) \leq d(x, y) + h(y) \tag{3.2}$$

pro každou hranu x, y grafu (d je délka této hrany), potom h je monotónní (někdy též označována jako konzistentní). V tomto případě algoritmus navštíví každý uzel maximálně jednou.

Samotný algoritmus pak probíhá následovně. Je vytvořena a udržována prioritní fronta otevřených, tj. ještě nenavštívených uzlů. Čím menší je hodnota $f(x)$ pro daný uzel x, tím vyšší má prioritu. V každém kroku algoritmu je uzel s nejvyšší prioritou odebrán z prioritní fronty a jsou spočítány hodnoty f a h pro jeho sousední uzly. Tyto uzly jsou pak přidány do prioritní fronty anebo jsou sníženy jejich hodnoty, pokud ve frontě už jsou a nové hodnoty jsou nižší. Algoritmus pokračuje, dokud nemá konečný uzel menší hodnotu f, než libovolný jiný uzel z fronty, nebo dokud není tato fronta prázdná. Hodnota f koncového uzlu je poté délkou nejkratší cesty grafem. Pokud je potřeba znát i konkrétní cestu, je nutné udržovat si i seznam uzlů na této cestě. Pro udržování stačí pamatovat si v každém uzlu jeho (libovolného) předchůdce na nejkratší cestě.

Složitost

Složitost algoritmu závisí na použité heuristice. V nejhorším případě je počet prozkoumaných uzlů exponenciální vzhledem k délce řešení. V optimálním případě je složitost polynomiální. Optimálním případem se rozumí stav, kdy je prohledávaný prostor stromem, existuje pouze jeden optimální stav a heuristická funkce h splňuje následující podmínku:

$$|h(x) - h^*(x)| = O(\log h^*(x)), \tag{3.3}$$

kde h^* je optimální heuristika, tj. přesná vzdálenost z x do koncového uzlu. Jinými slovy podmínka říká, že chyba heuristiky h neporoste rychleji, než logaritmus optimální heuristiky. [22], [5], [19], [26]

3.3 Výběr datové struktury a grafového algoritmu

V této kapitole budou vyjmenovány všechny výhody a nevýhody popsaných datových struktur a grafových algoritmů. Bude zde popsáno, které struktury a algoritmy se hodí pro jaké typy případů. Konkrétní výběr ale bude uveden a vysvětlen v kapitole 4.

3.3.1 Zhodnocení datových struktur

Při použití AR objektů jako navigačních bodů se získají přesné pozice, na které lze navigovat. Je možné si jakkoli rozšířit základní sadu markerů a označit si tak libovolnou věc a dát jí specifický význam. Navíc se lze ke každému markeru navigovat přímo a opakovaně s nejvyšší možnou přesností konkrétní pozice v prostoru. Na druhou stranu se nelze navigovat na místo, kde není umístěný žádný marker.

Tuto situaci řeší druhý způsob využití markerů, kdy pouhým označením zdí a dveří pomocí markerů vznikne CAD model. Rozdělením plochy pomocí mřížky tak vznikne možnost se nechat navigovat na jakékoliv políčko dané mřížky. Problém zde nastává v případě požadavku mít konkrétně rozmístěné pozice, na které se uživatel chce nechat navigovat. Částečným řešením může být jemnost mřížky, to problém ale zcela nevyřeší.

3.3.2 Zhodnocení grafových algoritmů

V tomto zadání se hledá algoritmus, který uživatele dovede do cíle po co nejkratší trase. Tím pádem lze z výběru odebrat BFS a GBFS. Oba dva algoritmy totiž nenajdou vždy tu nejkratší cestu. BFS za nejkratší cestu považuje tu, která má nejméně vrcholů, což neplatí v případě, že ohodnocení hran není konstantní. GBFS na druhou stranu díky heuristice sice najde cestu rychleji, ale v případě prostředí s velkým množstvím překážek nebude skoro nikdy cesta nejkratší.

Zůstávají tedy algoritmy Dijkstra a A*. V tomto případě záleží na zvolení datové struktury navigačního grafu. Při použití grafu s nekonstantním ohodnocením hran nebude v grafu tolik vrcholů, aby časová složitost hrála velkou roli, a použity by mohly být oba algoritmy. V případě použití velmi jemné navigační mřížky pro velkou plochu by bylo více vhodné použít A* algoritmus pro rychlejší nalezení nejkratší cesty.

4 Navrhované řešení

Po prozkoumání všech důležitých oblastí byl získán dostatek podkladů pro návrh výsledné aplikace. Ta by měla sloužit k navigaci ve skladových prostorách za použití technologie Tango. Technologie je relativně nová a nepříliš rozšířená. Výsledkem implementace tedy bude aplikace, která zvládne jednoduchou ukázku navigace a přesvědčí uživatele, že danou problematiku tato technologie nejen zvládne, ale že je zároveň tím správným směrem pro řešení tohoto typu problému.

4.1 Použité technologie

V předchozích kapitolách byl vybrán hardware, software, struktura grafu a algoritmus nejkratší cesty. V této kapitole bude výběr doplněn o jazyk a vývojové prostředí.

4.1.1 Vývojové prostředí a jazyk

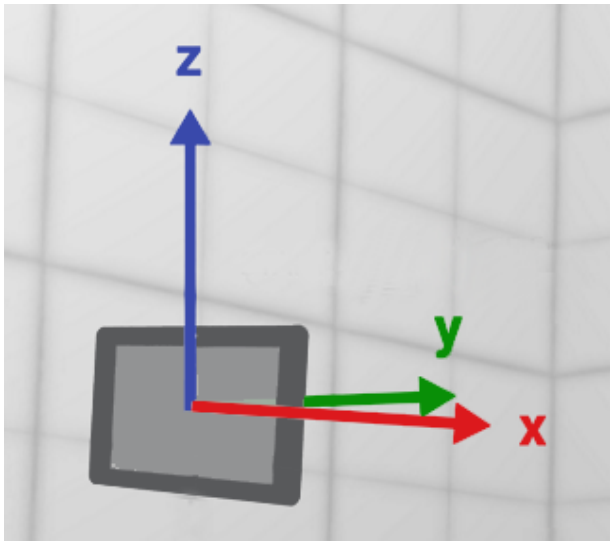
Aplikace bude implementována v prostředí Unity. Toto vývojové prostředí se používá pro tvorbu her. Lze v něm vytvářet scény z grafických objektů *Prefabů*. Těmto objektům je možné přidělit pomocí `C#` skriptů vlastnosti, chování, animace a další. Pokud aplikace bude tvořit pouze ukázkové demo, které nebude obsahovat tisíce vrcholů a hran, nebude tedy záležet tolik na optimalizaci, navrhuji využít Unity Tango SDK pro jeho schopnost pracovat s grafickými objekty. Unity IDE podporuje vývoj s Tango API. Ukázky použití Unity Tango SDK přímo od společnosti Google jsou k nalezení zde: `https://github.com/googlesamples/tango-examples-unity`

4.1.2 Alternativy

Hlavním vývojovým jazykem tedy bude C#. Alternativním řešení je jazyk Java nebo C++. Jako grafická knihovna se nabízí *OpenGL*, která by mohla spolupracovat buď přímo s jazykem Java nebo přes JNI s C++.

4.2 Souřadné systémy

Každá hardwarová i softwarová komponenta má definované vlastní prostředí pro práci s 3D souřadnicemi. Jednotlivá prostředí mají definovaný souřadný systém, který se může od souřadného systému jiných komponent lišit. Rozdílné souřadné systémy se nerovnají například v prostředí jednotky *IMU*[1], grafické knihovny OpenGL a vývojového prostředí Unity. Pro práci s 3D souřadnicemi je tedy nutné si definovat transformační matice pro převod souřadnic mezi jednotlivými prostředími.



Obrázek 4.1: Pravidlo pravé ruky: Right Hand Local Level.[11]

4.2.1 Tango souřadný systém

Tango používá dva typy souřadného systému, které jsou oba založené na *pravotočivém systému souřadnic*. První systém se používá v prostředí Tango API a jmenuje se *Right Hand Local Level*, druhý systém se jmenuje *Right Hand Android* a používá se v prostředí systému Android.

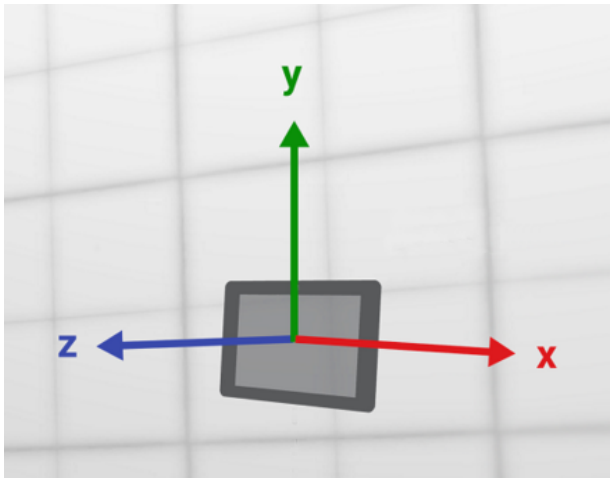
Right Hand Local Level. Je-li zadán střed souřadné soustavy umístěný na naší hrudi, tak osa X je ve směru pravé ruky, osa Y je směr od hrudi dopředu a osa Z směřuje k bradě, viz 4.1.

[1]Inertial measurement unit - je jednotka kombinující akcelerometr, gyroskop a magnetometr pro měření zrychlení, rotace a magnetického pole zařízení, ke kterému je jednotka připojená.

Right Hand Android. Je-li střed soustavy zadaný stejně jako v předchozím případě, pak osa X opět míří k pravé ruce, osa Y tentokrát míří k bradě a osa Z směřuje za naše záda, viz 4.2.

4.2.2 OpenGL souřadný systém

V prostředí OpenGL se používá pro definici světa (**OpenGL World Frame**) stejný systém jako pro kameru (**Camera Frame**) a tím je pravotočivý souřadný systém **Right Hand Android**. Je-li Tango aplikace psána pro platformu Android a bude-li nastavení kamery defaultní, není nutné počítat žádné převody mezi prostředími.



Obrázek 4.2: Pravotočivý souřadný systém: Right Hand Android.[11]

4.2.3 Unity souřadný systém

V prostředí *Unity* se používá *levotočivý souřadný systém* pro kameru (**Camera Frame**) i pro svět (**Unity World Frame**). Je-li střed soustavy souřadnic zadán stejně jako v předchozích případech, tak osa X míří k pravé ruce, osa Y míří k bradě a osa Z míří dopředu, viz 4.3.

V tomto případě prostředí Unity používá odlišný souřadný systém, než Tango. Je tedy nutné si definovat transformační matice, kterými se sjednotí 3D souřadnice z prostředí Tango API se souřadnicemi v prostředí Unity. Označení T^{Base}_{Target} definuje průběh transformace, kde $Base$ je výchozí a $Target$

je cílový souřadný systém prováděné transformace. Pro souřadnice z prostředí Tango API platí označení T_D^{SS}, kde SS znamená nastavení souřadného systému při startu zařízení **Start of service** a D značí souřadný systém zařízení **Device**.

4.2.4 Transformace Tango API na OpenGL

Při transformaci souřadnic z Tango API na OpenGL je zavedena transformační matice s označením T_{SS}^{OW}, kde OW znamená souřadný systém světa (**OpenGL World**) prostředí OpenGL. Je to převod mezi pravotočivým systémem Tango API a pravotočivým systémem Android. Vznikne tak matice

$$T_{SS}^{OW} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

a druhou transformací bude převod mezi kamerou v prostředí OpenGL (**OpenGL Camera**) a prostředím Tango **Device**. Protože obě prostředí používají stejný souřadný systém, bude tato matice identity.

$$T_{OC}^{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Vynásobením matic se získá transformační matice, která dokáže převádět souřadnice z prostředí Tango API do prostředí OpenGL. Výsledná transformace vypadá následovně:

$$T_{OC}^{OW} = T_{SS}^{OW} \cdot T_{D}^{SS} \cdot T_{OC}^{D} \tag{4.3}$$

Jelikož je T_{OC}^{D} matice identity, lze ji v rovnici vynechat, na výsledek to nemá žádný vliv.

4.2.5 Transformace Tango API na Unity

Transformace mezi prostředími Tango API a Unity se provede podobným způsobem jako v předchozím případě. První transformace bude mezi systémem **Start of service** a systémem světa (**Unity World**) v prostředí Unity.

$$T_{SS}^{UW} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

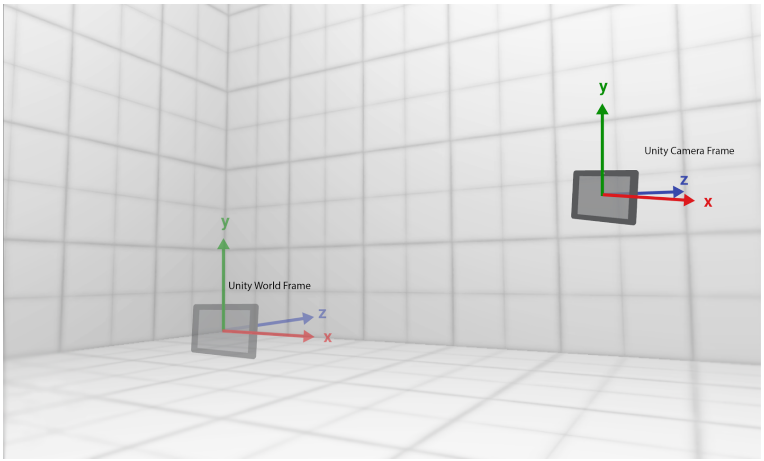
Druhá transformace bude mezi kamerou (**Unity Camera**) v prostředí Unity a zařízením (**Device**) v prostředí Tango. Zde nevznikne matice identity jako v případě převodu na OpenGL, protože Unity používá levotočivý systém, je tedy potřeba na ose Z změnit polaritu a otočit ji na druhou stranu.

$$T_{UC}^{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

Vzorec výsledné transformace vypadá následovně:

$$T_{UC}^{UW} = T_{SS}^{UW} \cdot T_{D}^{SS} \cdot T_{UC}^{D} \tag{4.6}$$

Nyní jsou definované všechny potřebné transformace pro obě možnosti implementace rozšířené reality s využitím Tango API. [11]

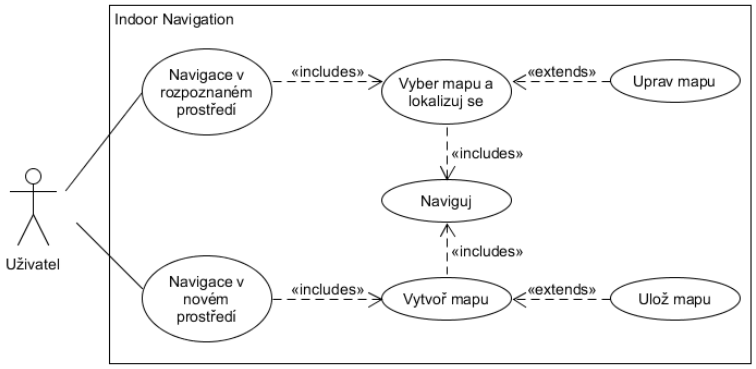


Obrázek 4.3: Pravidlo levé ruky.[11]

4.3 Průběh aplikace

Výsledná aplikace bude pro zařízení Google Yellowstone Tango. Ke skenování oblasti se použije Area Learning a výsledný model bude uložen do souboru

typu ADF. Navigace bude realizována pomocí navigačního grafu, který bude tvořen markery podle kapitoly 3.1.1. Markery budou reprezentovány AR objekty symbolizujícími konkrétní místo v 3D prostoru, na které se bude možné navigovat. Protože aplikace pro správnou funkčnost potřebuje přítomnost Tango API, je dobré, aby při spuštění nebo při instalaci aplikace zjistila, zda ji vůbec bude možné používat v plném rozsahu. Přítomnost Tango API v zařízení lze vynutit i při instalaci přes *Google Play*. Zde obchod bez splnění požadavků na zařízení nepovoluje instalaci nekompatibilních aplikací.



Obrázek 4.4: Graf případů užití (UseCase).

Na začátku budou stanoveny základní funkce, které bude aplikace poskytovat. Aplikace bude rozlišovat navigaci v *rozpoznané* a *nerozpoznané* oblasti viz *UseCase* 4.4. Základní výčet funkcí aplikace:

1. Zobrazení a výběr dostupných map
2. Lokalizace
   (a) Ve vlastní oblasti
   (b) V oblastech naskenovaných aplikacemi třetích stran
3. Připojení a zobrazení AR objektů v ADF oblasti
4. Úprava AR objektů
5. Uložení oblasti
6. Navigace

V případě navigace v rozpoznané oblasti je nutné zajistit funkce 1-6, pro případ navigace v nerozpoznané oblasti je nutné zajistit pouze funkce 4-6.

4.3.1 Práce s oblastmi

Po kontrole přítomnosti Tango API si uživatel zvolí buď navigaci v rozpoznané nebo nerozpoznané oblasti. Pro rozpoznanou oblast bude nutné nejprve vybrat mapu z nabídky možností. Tato nabídka bude obsahovat všechny dosud skenované oblasti, tzn. jak oblasti skenované touto aplikací, tak oblasti skenované aplikacemi třetích stran. Po výběru oblasti se spustí lokalizace, která rozpozná dané prostředí a řekne zařízení, kde se v prostoru nachází tak, že mu předá 3D souřadnice aktuální pozice. Princip lokalizace je podrobněji popsán v kapitole 2.2.3.

Je zapnutý learning mode?	**Je načtena oblast z ADF?**	**Lze uložit oblast do ADF?**
Ne	Ne	Nelze uložit skenovanou oblast.
Ano	Ne	Oblast bude uložena s novým UUID.
Ne	Ano	Nelze uložit skenovanou oblast.
Ano	Ano	Nelze uložit skenovanou oblast dokud se neprovede lokalizace vůči oblasti z ADF. Po uložení vznikne nová oblast s novým UUID.

Tabulka 4.1: Vliv nastavení Tanga na ukládání oblastí.[10]

Dále by měl mít uživatel možnost aktivovat či deaktivovat učící mód, tzv. *learning mode*. Když je učící mód aktivní, ukládají se do `RAM` paměti zařízení důležité body scény, jenž jsou použity při pozdější lokalizaci. To znamená, že při vytváření nové mapy musí být učící mód aktivní vždy. Při požadavku pouze se lokalizovat v oblasti je možné použít lokalizaci bez aktivního učícího se módu. Chce-li uživatel oblast rozšířit o další části scény, je nutné spustit lokalizaci s aktivním učícím módem, lokalizovat se v rozpoznané části a následně oskenovat novou část scény. Po uložení se vylepšené oblasti

vygeneruje nové `UUID`. Vliv učícího se módu a lokalizace na výsledné ADF viz tabulka 4.1. Vytvořená a uložená oblast do ADF musí jít také smazat. Odstranění oblasti by tedy mělo být součástí aplikace.

4.3.2 3D perspektiva

Po lokalizaci do oblasti či volbě vytvořit oblast novou bude moci uživatel spravovat navigační graf ve 3D perspektivě. Aplikace bude poskytovat možnost vytvářet a rušit jak markery, které reprezentují vrcholy grafu, tak hrany grafu. Užívání by mělo být co nejvíce uživatelsky přívětivé a intuitivní tak, aby uživatel nemusel nad každým krokem zdlouhavě přemýšlet.

4.3.3 Získání pozice v 3D prostoru

Umístění vrcholu bude probíhat přes dotyk obrazovky. Uvidí-li uživatel na obrazovce místo, kde by chtěl mít vrchol, označí ho dotykem obrazovky. Získá se tak souřadnice $T(x, y)$. Zařízení zároveň vidí před sebou mračno bodů skenované scény. Souřadnicí T a technikou zvanou *raycast* se označí v mračnu místo, kde má být vrchol umístěn. Získá se bod $S(x, y, z)$, pomocí kterého lze spočítat vzdálenost od zařízení a škálovat tak velikost markeru, který označuje vrchol. Pro správné umístění je potřeba spočítat normálu N bodu S. Vezme se tedy okolí bodu S do určitého rádiusu a metodou nejmenších čtverců se spočítá rovina. Ta definuje normálu N bodu S. Bod S tedy říká, kde bude AR marker umístěn, normála N určuje směr natočení AR objektu. Odebrání vrcholu bude dvojitým dotykem. Po vytvoření grafu musí mít uživatel možnost si graf uložit.

4.3.4 Uložení AR objektů

Při ukládání oblasti nelze přidávat AR objekty přímo do ADF souboru. Ke skenované oblasti tak bude připojen `XML` serializovatelný soubor s definicemi jednotlivých markerů. Když se tedy uživatel lokalizuje, načte se navigační graf z `XML` souboru, který je přidružený k dané oblasti. O každém markeru budou uchovány následující informace:

- **Type** - typ
- **Position** - 3D souřadnice

- **Orientation** - orientace
- **Neighbours** - seznam sousedů
- **Id** - jednoznačný identifikátor

Tento výčet informací o jednotlivých markerech poskytuje dostatek informací, aby bylo možné zpětně zrekonstruovat plnohodnotný navigační graf se všemi vrcholy a hranami. Jelikož je každý ADF soubor označen jedinečným identifikátorem `UUID`, použije se tento identifikátor jako jméno `XML` souboru. Tím se zajistí jednoznačné přiřazení definice grafu ke skenované oblasti. Aplikace musí poskytovat možnost vytvářet a rušit jak vrcholy (markery), tak hrany grafu. Aplikace bude také nabízet pohled na graf z ptačí perspektivy, tzn. ve 2D.

4.3.5 2D perspektiva

Při pohledu na graf ve 2D uživatel může zjistit, že mu některé hrany v grafu chybí nebo naopak přebývají. V tomto náhledu tedy bude také existovat možnost graf upravit. Aby uživatel věděl, kde se nachází, bude v náhledu umístěný indikátor jeho aktuální pozice vůči grafu. Bude zde také barevně odlišený ten vrchol grafu, ke kterému je uživatel nejblíže. Jelikož se ale uživatel může vůči grafu nacházet kdekoliv a nemusí se pokaždé nacházet na jedné z hran grafu, bude tato vzdálenost určena pouze vzdušnou čarou. Ohodnocení vzdálenosti se získá zjednodušeným vzorcem pro výpočet velikosti úsečky (bez odmocniny) mezi body A a B ve 3D prostoru

$$|AB| = (x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2. \tag{4.7}$$

Bude-li se počítat vzdálenost od aktuální pozice zařízení ke všem vrcholům grafu, výsledná složitost výpočtu bude $O(N)$. Pokud se ale použije pro ukládání pozic vrcholů datová struktura *KD-Strom*, sníží se složitost případů v průměru na $O(\log N)$, nejhorší případ zůstává na složitosti $O(N)$. Pro případ, kdy N se pohybuje v řádech tisíců, je možné dosáhnout velkého urychlení.

Pokud bude graf příliš rozsáhlý, mohlo by se stát, že při zobrazení celého grafu na obrazovce budou vrcholy tak malé, že jakákoliv manipulace s grafem bude obtížná. Aplikace tedy bude poskytovat možnost přizpůsobovat si vzhled mapy pomocí gest.

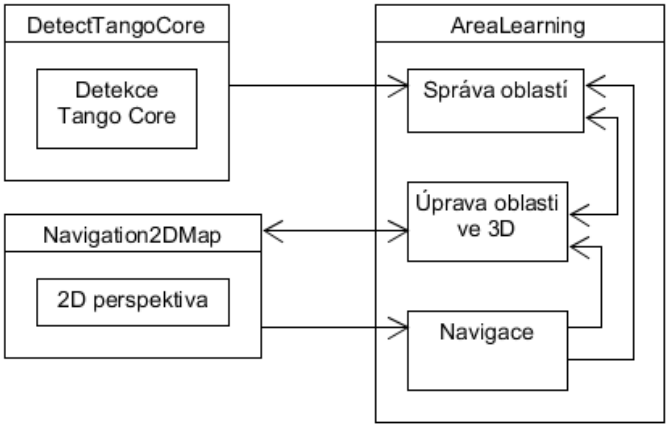
4.3.6 Navigace

Konečná navigace bude realizována pomocí AR. Po výběru cíle navigace se spočítá nejkratší cesta grafem Dijkstrovým algoritmem. Tato cesta bude tvořena AR hranami grafu, kterými vypočtená cesta vede. Ze zobrazené cesty musí být jednoznačné, kterým směrem se má uživatel k cíli vydat. Hrany tak budou tvořeny šipkami tak, že budou uspořádané ve směru navigace k cíli. Startovní pozice bude zvolena podle nejbližšího vrcholu vůči zařízení a cílovou pozici si uživatel určí sám.

Aplikace musí být znovupoužitelná, tzn. že ve fázi navigace musí mít uživatel možnost navigaci přerušit, zadat nový cíl nebo se lokalizovat v jiné mapě. Důvodem lokalizace do jiné mapy nemusí být změna prostředí, uživatel může chtít mít pro jednu oblast více variant navigačních grafů rozdělených například podle rolí zaměstnanců.

5 Implementace řešení

Aplikace se skládá ze 3 hlavních scén viz 5.1. První scéna `DetectTangoCore` řeší přítomnost Tango API, druhá scéna `AreaLearning` řeší lokalizaci s výběrem a vytvořením oblasti a poslední scéna `Navigation2DMap` řeší pohled na oblast z ptačí perspektivy.



Obrázek 5.1: UML diagram scén s přechody mezi obrazovkami.

V diagramu tříd, který je uveden v příloze C, jsou uvedeny veškeré implementované třídy. Pokud dále v textu bude zmíněna třída, která není v diagramu uvedena, znamená to, že je součástí Tango API.

5.1 Detekce Tango Core

Jelikož jde aplikace nainstalovat na jakékoliv zařízení s Androidem, je při spuštění provedena kontrola dostupnosti *Tango Core*. Toto jádro jde stáhnout pouze z *Google Play* a pouze na zařízení, která jsou schopna používat Tango API. Detekce může nabývat 3 stavů:

1. **Not Present** - Tango Core není nainstalováno,
2. **Out Of Date** - Tango Core je zastaralé,
3. **Present** - Tango Core je nainstalováno a je aktuální.

V případech (1) a (2) se uživateli zobrazí tlačítko `Download Tango Core`, které aplikaci přesměruje do Google Play. Zde je možnost potřebné jádro stáhnout. Pokud je zařízení s jádrem nekompatibilní, může uživatel aplikaci pouze odinstalovat. V případě číslo (3) se na obrazovce objeví tlačítko `Start` a uživatel může pokračovat do druhé scény.

5.2 Výběr oblasti

Druhá scéna se skládá ze tří částí. V té první probíhá správa skenovaných oblastí, ve druhé lze jednotlivé oblasti upravovat ve 3D prostředí a třetí slouží k zobrazení navigace.

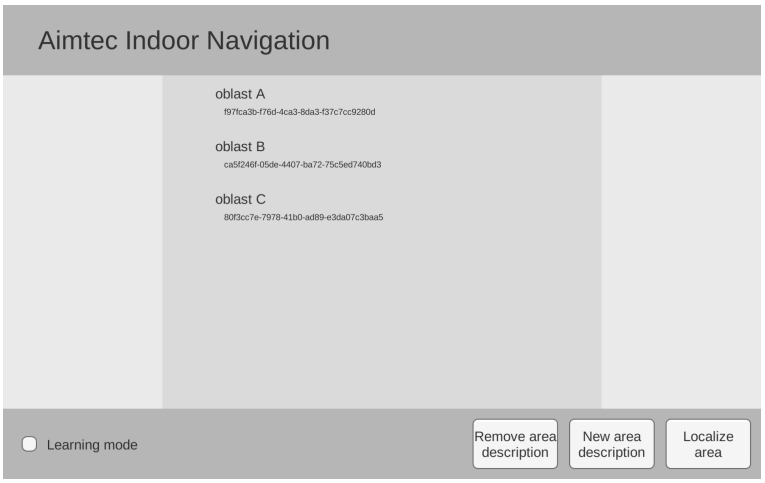
5.2.1 Správa oblastí

Správa oblastí obsahuje seznam již skenovaných a uložených oblastí do souboru ADF. Ten se načítá z centrálního úložiště, do kterého mají přístup všechny aplikace. Lze tak použít i skenované oblasti aplikací třetích stran, pokud používají centrální úložiště. Seznam všech oblastí se získá příkazem `AreaDescription.GetList()`, který je součástí Tango API. Tato část scény dále obsahuje trojici tlačítek `New Area Description`, `Localize Area` a `Remove area description`. První tlačítko slouží k vytvoření nové oblasti. Přepne se tedy rovnou do 3D prostředí pro úpravu grafu. Druhé tlačítko provede lokalizaci v prostředí proti zvolené uložené oblasti. Třetí tlačítko odstraní popis oblasti z ADF a `XML` definici grafu z lokálního úložiště aplikace. O definici grafu bude více informací v kapitole 5.3.2.

Lokalizace oblasti

Programové spuštění lokalizace se provede zavoláním metody `Startup` ze třídy `TangoApplication`. Vstupním parametrem metody je instance popisu oblasti `AreaDescription`, která se získá z ADF podle jejího `UUID`. Lokalizace probíhá automaticky a zařízení se při ní snaží inicializovat pohybový senzor. To se provádí tak, že se hledá shoda mezi právě viděnou scénou a uloženými daty z popisu oblasti. Jakmile je nalezena shoda, zařízení přiřadí spočítané prostorové umístění pohybovému senzoru na základě této shody dat pozorované scény.

Programátor pouze kontroluje moment, kdy je zařízení lokalizováno. To je zajištěno překrytím metody `OnTangoPoseAvailable`, která v časových intervalech poskytuje vstupní parametr typu `TangoPoseData`. Ten obsahuje informace o statusu, zda je aktuální souřadný systém `TANGO_POSE_VALID` nebo `TANGO_POSE_INVALID`. Jakmile je systém validní, zařízení začne z pohybového senzoru vracet souřadnice lokalizované oblasti. Po lokalizaci dojde k přepnutí do druhé části této scény. Posledním elementem této části scény je přepínač `Learning Mode`, který slouží k aktivaci učícího se módu (*learning mode*).

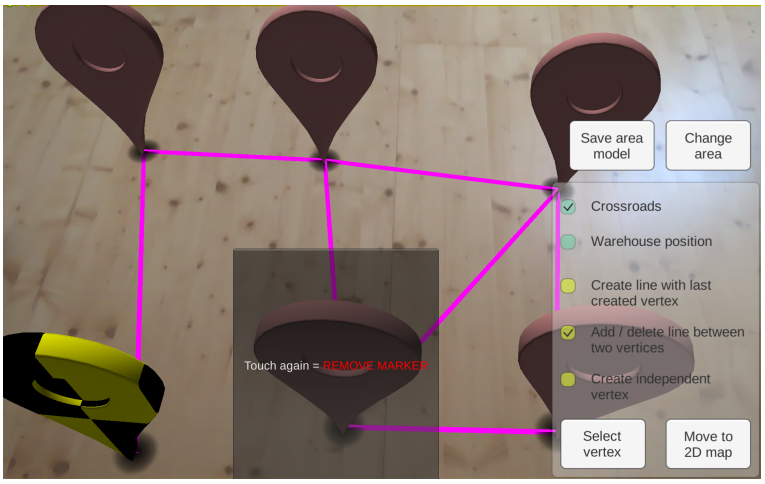


Obrázek 5.2: Seznam uložených oblastí.

Ten ve spojení s lokalizovanou oblastí z ADF hraje významnou roli ohledně ukládání procházené oblasti, viz tabulka 4.1. V případě, že uživatel zvolí tlačítkem `New Area Description` možnost vytvořit novou oblast a přitom nezaškrtne `Learning Mode`, tak podle tabulky nepůjde oblast uložit. V aplikaci tato situace nikdy nenastane, protože v případě volby vytvořit novou oblast je vždy automaticky spuštěn učící mód bez ohledu na stav přepínače. V případě aktivovaného učícího módu s lokalizovanou oblastí Tango přidává další data při opětovném procházení scény. Rozšiřuje a zdokonaluje tak původní data skenované oblasti a dělá tak popis univerzálnějším. Aktivního učícího módu lze využít i k přidání nových částí oblasti. Stačí se lokalizovat ve známé oblasti a z ní přejít do nové části, která bude ke stávajícímu modelu přidána. Pro uchování změn je nutné oblast znovu uložit. Po uložení oblasti je vygenerováno nové `UUID`.

5.3 3D prostředí

V druhé části této scény je možné upravit nebo vytvořit úplně nový navigační graf viz obrázek 5.3. Na pravé straně obrazovky je umístěné menu pro práci s vrcholy (markery). Nad menu se nachází dvě tlačítka `Change Area` a `Save Area Model`. První tlačítko umožňuje vrátit se zpět na výběr skenovaných oblastí, druhé uloží vytvořený model oblasti.



Obrázek 5.3: Pohled na navigační graf ve 3D s ukázkou označeného vrcholu (s popiskem) a vybraného vrcholu (žluto-černý).

5.3.1 Umisťování vrcholů

Menu obsahuje dvě skupiny přepínačů. V každé skupině je označena vždy jen jedna možnost. První skupina slouží k vybrání typu vrcholu

1. **Crossroad,**
2. **Warehouse position.**

Druhá skupina slouží k nastavení procesu umisťování vrcholů

1. **Create line with last created vertex,**
2. **Add/delete line between two vertices,**
3. **Create independent vertex.**

V první skupině momentálně mezi přepínači není rozdíl a oba fungují jako rozcestník. Do budoucna se počítá s rozšířením aplikace, kdy typ `Crossroad` bude značit rozcestník a typ `Warehouse position` bude označovat skladovou pozici, která bude svázána s položkou v databázi. Přepínače (1-3) ve druhé skupině slouží k nastavení umisťování a propojování vrcholů grafu. Možnost (1) automaticky propojuje hranou nový vrchol s předchozím vytvořeným. Možnost (2) propojuje/rozděluje dva vrcholy. Poslední možnost (3) povoluje umisťování samostatných vrcholů bez jakýchkoliv automaticky doplněných hran. Ty je tedy nutné doplnit dodatečně pomocí nastavení číslo (2). Pro výpočet umístění jednotlivých vrcholů využívá Tango API objekty `Tango Point Cloud` a `Tango AR Camera`. Nejdůležitějšími skripty objektu `Tango AR Camera` jsou:

1. `Tango AR Pose Controller` - distribuuje data z pohybového senzoru,
2. `Tango AR Screen` - zobrazuje AR objekty na obrazovce,
3. `Tango Environmental Lighting` - pracuje se světlem.

Tango Point Cloud je Unity objekt, který má na starosti zpracování a distribuci bodových mračen získaných z hloubkové (*depth*) kamery. Práci s mračny zajišťuje skript `TangoPointCloud`. Ten transformuje body z prostředí kamery do prostředí Unity světa. Obsahuje pomocné metody jako například nalezení roviny (`FindPlane`), nalezení podlahy (`FindFloor`) a nalezení nejbližšího bodu z mračna k hledané 3D souřadnici (`FindClosestPoint`).

Tango AR Pose Controller je Unity objekt, který zpracovává a distribuuje data z pohybového (*motion tracking*) senzoru. Zpracování dat řeší skript `TangoARPoseController`. Ten transformuje souřadnice ze senzoru do prostředí Unity světa. Skript neposkytuje žádné pomocné metody. Pouze distribuuje důležité hodnoty přes globální proměnné, jako jsou aktuální pozice `m_tangoPosition` a rotace zařízení `m_tangoRotation`.

Nalezení pozice ve 3D

Jak již bylo zmíněno, vrcholy se umisťují dotykem obrazovky. Zpracování dotyku obrazovky zajišťuje skript `AreaLearningInGameController` metodou `reactionOnTouch`. Nejprve se zjistí, zda dotyk neprochází jedním z AR objektů scény. To se provede za pomoci souřadnice (x, y) získané z dotyku

obrazovky a vysláním ray-castu do mračna bodů.

Ray-cast propojuje souřadnici obrazovky (x, y) a reálnou souřadnici (x, y, z) nalezenou v právě snímaném mračnu bodů. K realizaci ray-castu slouží metoda `Physics.Raycast`, jejíž vstupními parametry jsou `Vector2` (souřadnice dotyku obrazovky) a `RaycastHit`. Druhý vstupní parametr bude vyplněn uvnitř metody údaji o průchodu ray-castu scénou. Lze z něj pak zjistit, zda vyslaný ray-cast prošel AR objektem nebo dopadl čistě na mračno bodů. Pokud nedošlo k protnutí AR objektu, umístí se nový AR objekt do scény na souřadnice dotyku v mračnu bodů. Zjednodušený kód nalezení pozice a určení normály pro umístění nového markeru, kde `pointCloud` je reference na objekt `Tango Point Cloud`:

```
private Vector3 FindPlane(Vector2 touchPosition)
{
    Camera cam = Camera.main;
    Vector3 planeCenter;
    Plane plane;

    if (!pointCloud.FindPlane(cam, touchPosition,
                              out planeCenter, out plane))
    { return null; }

    return plane.normal;
}
```

Proměnná `touchPosition` je souřadnice dotyku obrazovky. V mračnu bude nalezena plocha, jejíž vlastnosti budou zapsány do proměnné `plane`. Získání normály už je dále triviální přes tečkovou notaci z `plane.normal`.

Instance vrcholu

Každý vrchol je marker datového typu `GameObject`. Tento objekt je spojený se skriptem `ARMarker`, který popisuje jeho chování. Každý vrchol si drží seznam sousedních vrcholů, ke kterým má hranu. V tomto seznamu nejsou všechny hrany, ale pouze ty, u kterých byl při vytváření hrany tento vrchol uveden jako první. Jednotlivé hrany jsou tvořeny pomocí datového typu `LineRenderer` z prostředí Unity. Po výpočtu 3D souřadnice a normály lze

vytvořit instanci vrcholu. Zjednodušený postup vytvoření a nastavení nového vrcholu:

```
GameObject marker = Instantiate(m_markPrefabs[m_currentMarkType],
planeCenter,Quaternion.LookRotation(forward, up)) as GameObject;

ARMarker markerScript = marker.GetComponent<ARMarker>();

Matrix4x4 uwTDevice = Matrix4x4.TRS(
                        poseController.m_tangoPosition,
                        poseController.m_tangoRotation,
                        Vector3.one);
Matrix4x4 uwTMarker = Matrix4x4.TRS(
                        marker.transform.position,
                        marker.transform.rotation,
                        Vector3.one);

markerScript.m_deviceTMarker =
                          Matrix4x4.Inverse(uwTDevice) * uwTMarker;
```

Do proměnné `marker` typu `GameObject` se připraví defaultní instance vrcholu. Přes `GetComponent` se získá skript typu `ARMarker`, který popisuje vlastnosti a chování vrcholu. Připraví se transformační matice, která se uloží do vrcholu právě přes vytažený skript. Tato matice slouží k přepočítání pozice a natočení vrcholu.

Odstranění vrcholu

Pokud při dotyku obrazovky došlo k protnutí AR objektu zobrazí se na vrcholu okno s hláškou `Touch again = REMOVE MARKER`, to znamená, že pokud dojde k dalšímu dotyku, daný objekt bude ze scény odstraněn. Odstranění se provede zavoláním metody `Destroy(GameObject)`, která ovlivňuje životní cyklus Unity objektů. Okno vyvolané prvním dotykem AR objektu zároveň slouží jako označení vrcholu k vytvoření/odstranění hrany mezi vrcholy při nastavení druhé skupiny přepínačů v menu na volbu číslo (2).

Při odstranění vrcholu je nutné zajistit, aby nezůstaly zobrazeny žádné hrany, které původně vedly do odstraněného vrcholu. Hrany ze seznamu

sousedů, který je součástí instance vrcholu, se automaticky odstraní. Naopak hrany, které jsou spojnicí mezi odstraněným vrcholem a jeho sousedem, ale jejich instance patří sousedovi, musí být odstraněny dodatečně. Podle `ID` odstraněného vrcholu jsou procházeny všechny vrcholy a pokud některý z nich v seznamu sousedů nalezne hranu vedoucí k odstraněnému vrcholu, je hrana smazána. Toto řešení je v pořádku pro malé množství vrcholů. V aplikaci, u které bude počet vrcholů v řádu tisíců, je lepší zvolit implementaci pomocí seznamu sousednosti a při mazání vrcholů tak procházet jen seznam sousedů daného vrcholu.

Práce s hranami

Při nastavení menu na `Add/delete line between two vertices` přibude v menu tlačítko `Select vertex`, kterým se potvrzuje výběr vrcholu. Algoritmus přidání či odebrání hrany funguje následovně:

1. Výběr prvního vrcholu dotykem obrazovky
2. Potvrzení výběru tlačítkem `Select vertex`
3. Dojde k barevnému odlišení vrcholu
4. Výběr druhého vrcholu dotykem obrazovky
5. Potvrzení výběru tlačítkem `Select vertex`
6. Aktualizace hrany mezi vybranými vrcholy
   (a) hrana existuje - bude zrušena
   (b) hrana neexistuje - bude vytvořena

5.3.2 Ukládání oblastí

Ukládání oblastí probíhá v `AreaLearningInGameController` v metodě `Save`. Samotné uložení se provede zavoláním metody `SaveCurrent` nad třídou `AreaDescription`. Metoda vrátí instanci skenované oblasti. Nad instancí se zavolá `GetMetadata`, což vrátí instanci s metadaty uložené oblasti. Nové jméno oblasti se uloží zavoláním metody `SaveMetadata` nad instancí oblasti. Metoda má vstupní parametr metadata s upraveným názvem oblasti. Ukázka přiřazení nového jména oblasti:

```
AreaDescription newDesc = AreaDescription.SaveCurrent();
AreaDescription.Metadata metadata = newDesc.GetMetadata();
metadata.m_name = "area_name";
newDesc.SaveMetadata(metadata);
```

Každá oblast je uložena defaultně do ADF a zároveň je označena jedinečným identifikátorem `UUID`. Protože ale neexistuje přímý přístup do ADF souboru, veškerá dodatečná data se ukládají do `XML`. Důležitá data o vrcholech jsou popsána v kapitole 4.3.2. `XML` soubor je uložen ve složce aplikace (neexistuje přístup zvenčí) a jeho jméno je `UUID` skenované oblasti s koncovkou `XML`. Pro každou oblast popsanou jedinečným identifikátorem `UUID` může existovat nejvýše jeden `XML` soubor.

5.3.3 Uzavírání smyček

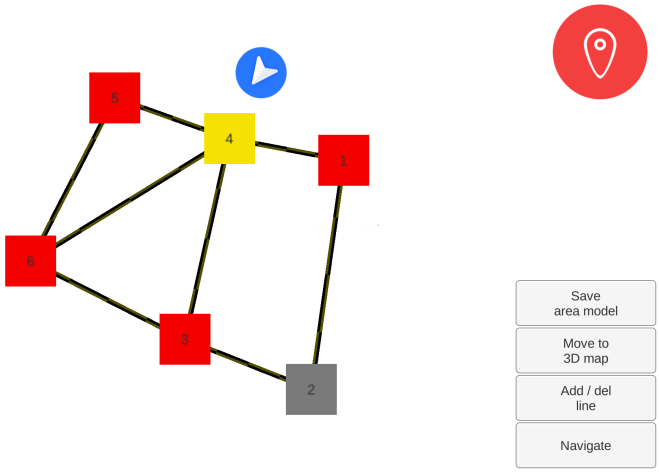
V kapitole 2.2.3 je popsán princip uzavírání smyček. Z tohoto důvodu je po uložení grafu nutné aktualizovat pozici rozmístěných vrcholů. Aktualizace se provádí v metodě `Update`, která řídí životní cyklus Unity objektů a probíhá přibližně 4x za sekundu. Občas se stane, že celý graf na obrazovce lehce změní pozici, to je důsledek uzavření smyčky a aktualizace pozic vrcholů. Toto je vliv úpravy souřadnic z pohybového senzoru. Při ukládání grafu je použito v tu chvíli nejlepší nastavení souřadného systému a aktualizují se souřadnice všech vrcholů. Při aktualizaci je použita transformace souřadnic popsaná v kapitole 4.2.5 a transformační matice `m_deviceTMarker` z popisu vytvoření instance vrcholu `marker`. Zjednodušená ukázka transformace a přiřazení souřadnic vrcholu:

```
Matrix4x4 uwTDevice = m_poseController.m_uwTss
                        * relocalizedPose.ToMatrix4x4()
                        * m_poseController.m_dTuc;

Matrix4x4 uwTMarker = uwTDevice * marker.m_deviceTMarker;

obj.Value.transform.position = uwTMarker.GetColumn(3);
obj.Value.transform.rotation = Quaternion.LookRotation(
                        uwTMarker.GetColumn(2),
                        uwTMarker.GetColumn(1));
```

První matice slouží k převodu souřadnic mezi systémem Unity světa a zařízením. Druhá matice slouží k převodu souřadnic mezi předchozí vytvořenou transformací a souřadnicemi vrcholu z pohledu kamery, která je zde reprezentována proměnnou `uwTMarker`.

5.4 2D prostředí

Tlačítkem `Move to 2D map` se aplikace přepne do třetí scény, do pohledu na graf z ptačí perspektivy. Při přechodu mezi scénami se na pozadí spočítá matice sousednosti. Podle teoretické části je lepší použít implementaci seznamu sousednosti, ale v tomto případě, kdy aplikace slouží jako ukázkové demo, nemá použití matice sousednosti žádný vliv na výsledný výkon. Při přechodu se předchozí scéna zachová aktivní na pozadí. Je tak učiněno z důvodu zachování aktivního senzoru pohybu, který neustále aktualizuje pozici v 3D prostředí. Kdyby se aktivoval ve 2D prostředí nový pohybový senzor, nestačilo by pouze předat poslední známou pozici a pokračovat dále. Aplikace by se musela znovu lokalizovat. To by s sebou neslo další komplikace, jako je například nutnost uložení oblasti vytvořené ve 3D, kterou ale uživatel vůbec nemusí chtít ukládat, protože pouze demonstruje funkčnost a uloženou oblast by dále nevyužil.



Obrázek 5.4: Pohled na graf z ptačí perspektivy.

Tato scéna obsahuje graf s indikátorem pozice zařízení vůči grafu, který je znázorněn modrým kruhem s bílou šipkou. Indikátor je aktualizován daty získanými z pohybového senzoru `Tango AR Pose Controller`. Vrcholy grafu

mohou být různě zbarvené:

1. **červená** - defaultní barva vrcholu
2. **šedá** - vrchol označený jako cíl navigace
3. **žlutá** - vrchol nejblíže k zařízení vzdušnou čarou
4. **zelená** - vrchol vybraný pro vytvoření či odstranění hrany

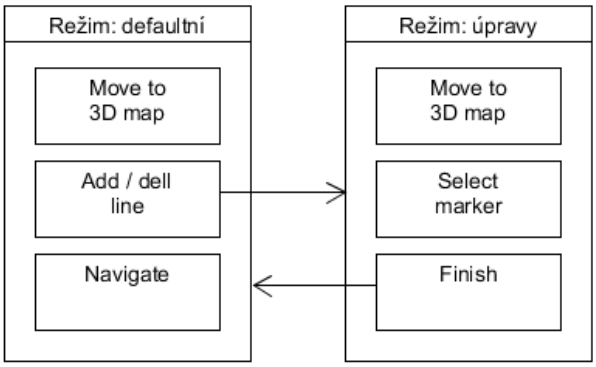
V pravém horním rohu je centrovací tlačítko. Je-li obarvené černě, znamená to, že indikátor pozice zařízení je statický a umístěný na středu scény, kterou kamera zabírá. Přenáší se pouze rotace zařízení. Kamera je v tomto případě dynamická a pohybuje se na základě souřadnic z pohybového senzoru. V opačném případě je tlačítko červené, kamera má statické souřadnice a indikátor se pohybuje po scéně na základě souřadnic pohybového senzoru.

Ve výchozím nastavení je centrovací tlačítko vždy černé. Ve 2D perspektivě aplikace povoluje i gesta pro manipulaci se scénou. Přibližováním či oddalováním dvou prstů na obrazovce se mění velikost grafu. Tahem jedním prstem po obrazovce se mění pozice kamery a tedy i část sledované scény. Navrácení polohy kamery do defaultního stavu se provede stiskem centrovacího tlačítka. Centrování a gesta zajišťuje skript `PinchZoom` připojený ke kameře 2D scény. V pravém dolním rohu se nachází menu s konfiguračními tlačítky. Seznam zobrazitelných tlačítek:

1. **Save area model** - uložení oblasti
2. **Move to 3D map** - přesun zpět do 3D perspektivy
3. **Add/del line** - přidat/odebrat hranu mezi vrcholy
4. **Select vertex** - vybrat označený vrchol
5. **Finish** - dokončení úprav grafu
6. **Navigate** - spuštění navigace

Tlačítkem (1) je možné uložit upravený graf. Tlačítkem (2) se provede přepnutí do předchozí 3D scény. Při každé změně grafu ve 2D se vše okamžitě propisuje i do vzhledu ve 3D. Tlačítkem (3) se scéna přepne do režimu úprav

viz 5.5. Zde je možné přidávat a odebírat hrany mezi mezi vrcholy. Z uživatelského pohledu je postup modifikace hran stejný jako ve 3D. To znamená, že se dotykem provede výběr prvního vrcholu a ten se potvrdí tlačítkem (4). Stejným způsobem se vybere i druhý vrchol. Potvrzením výběru druhého vrcholu se aktualizuje vzhled grafu. Tlačítkem (5) se opustí režim úprav a aplikace se vrátí do defaultního režimu. Poslední možnost (6) spouští navigaci na vybraný vrchol grafu, který musí být označen šedivou barvou. Provede se návrat do předchozí scény a začne se provádět její třetí část, navigace.



Obrázek 5.5: Vizualizace viditelnosti tlačítek v závislosti na zvoleném režimu 2D scény.

5.4.1 Navigační 2D graf

Graf je ve 2D tvořen maticí sousednosti, která se spočítala při přechodu z předchozí scény. Vrcholy tvoří objekty typu `Button`, jejichž souřadnice rozmístění jsou škálované (x, z) z původních souřadnic vrcholů 3D scény. Tyto souřadnice jsou zvoleny podle souřadného systému Unity viz obrázek 4.3. Všechny hrany jsou uloženy v mapě typu `Dictionary` v párech klíč-hodnota typu `<KeyPair, GameObject>`. Třída `KeyPair` uchovává dvě hodnoty typu `Integer`. Touto dvojicí je reprezentován identifikátor jednotlivých hran. Při odstranění hrany tedy stačí projít danou mapu a najít ten správný klíč. Podle klíče se získá `GameObject` reprezentující hranu. Ta se odstraní metodou `Destroy(GameObject)`. Odstranění hrany je nutné zanést i do matice sousednosti, aby její stav neustále odpovídal vzhledu grafu.

5.5 Navigace

Pokud graf obsahuje více než jednu komponentu, může se stát, že cesta do cílového vrcholu nebude existovat. V tom případě při pokusu navigovat se na takový vrchol dojde k zobrazení upozornění, že daná cesta neexistuje. Před každým spuštěním navigace probíhá kontrola, zda je možné se do cílového vrcholu dostat. Dijkstrův algoritmus vrací nejkratší cestu ve formě pole identifikátorů vrcholů. Index pole je identifikátor vrcholu a hodnota pod tímto indexem je identifikátor příchozího vrcholu. Není-li v poli na indexu s hodnotou identifikátoru cílového vrcholu hodnota různá od nuly, cesta do cíle neexistuje.

Obrázek 5.6: Navigace do cílového vrcholu.

Jako startovní pozice je vybrán ten vrchol, ke kterému je zařízení vzdušnou čarou nejblíže. Ve 2D scéně je to vrchol, který je obarven žlutě. Navigovaná cesta je tvořena původními hranami grafu, kterým byl změněn `Material` na skupinu šipek tak, aby směřovaly k cíli, viz obrázek 5.6. V navigačním módu je vypnutý dotykový režim, aby nešly přidávat další vrcholy grafu v době, kdy běží navigační prostředí. Jediné funkční elementy jsou na obrazovce tlačítka `End navigation` a `Change area`. Tlačítko `End navigation` ukončí navigaci, skryje vyznačenou cestu a zobrazí původní graf. Tlačítkem `Change area` se ukončí navigace a aplikace se přepne na scénu s výběrem oblastí k lokalizaci.

6 Testování

Tato kapitola popisuje ověření funkcionality aplikace za různých situací. Zkoumá správnou funkčnost a poukazuje na některé nedokonalosti řešení v oblastech skenování, vytváření grafu, lokalizace a navigace.

6.1 Skenování

Při skenování bylo ověřeno, že kvalitní model, který se dokáže rychle lokalizovat potřebuje co nejvíce dat. Pokud zařízení po celou dobu tvorby grafu zabírá pouze jedno místo, lokalizace je nejúspěšnější pouze z tohoto místa. Pokud ale uživatel nejprve projde celou oblast alespoň po obvodu, značně se tím zvyšuje úspěšnost lokalizace z různých míst skenované oblasti. Z toho vyplývá, aby byl vytvořen kvalitní popis oblasti, je nutné ji projít důkladně. To znamená projít všechny dostupné trasy tak, aby se uzavřelo co nejvíce smyček. Tím se maximalizuje korekce odchylky pohybu. Uzavírání smyček také snižuje odskakování AR objektů, protože v následném průchodu oblasti po správném oskenování budou korekce souřadnic minimální. Kvalitní model také přispívá k rychlejší lokalizaci, protože Tango má k dispozici dostatek záchytných bodů, podle kterých může danou oblast rozpoznat.

6.2 Vytváření grafu

Při testování grafu šlo spíše o vypořádání se s nepredikovatelným zacházením s aplikací z pohledu uživatele. Aplikace ve 2D pohledu nijak nerozlišuje, zda byl vrchol přidán na zeď či na zem. Pokud mají všechny vrcholy vždy dostatečně rozdílné hodnoty minimálně na dvou osách, v zobrazení ve 2D nenastane žádný problém. Pokud ovšem uživatel umístí například dva vrcholy na stěnu tak, že se jejich souřadnice mění například pouze na jedné ose, budou se tyto dva vrcholy ve 2D pohledu překrývat. Viz kapitola B.3 přílohy.

Další úskalím je samotné reálné umístění vrcholu. Pokud bude vrchol umístěn na předmět, který v průběhu času bude ze scény odebrán nebo změní pozici, vrcholy na tuto změnu nebudou reagovat. Pozice vrcholu v prostoru

je přidělena pouze jednou - a to při jeho vytvoření. V průběhu aplikace ji nelze měnit. Viz kapitola B.2 přílohy.

6.3 Lokalizace

Část lokalizace se zaměřuje hlavně na lokalizaci v oblasti, která byla zkoumána za různých světelných podmínek s různým počtem změn v prostředí. Lokalizace byla testována v místnosti o rozměrech 4,5 x 3,5 m. Oblast byla skenována jedním průchodem po obvodu místnosti za večerního světla. V místnosti se nacházela magnetická tabule, 2 stoly, židle a pohovka. Ta simulovala regál skladového prostředí. Na pohovce byly vyskládány krabice s věcmi. Na prostředku místnosti byl volný prostor na kterém byl vytvořen navigační graf. Velikost grafu při lokalizaci nehraje žádnou roli, protože graf není součástí ADF, a tak na lokalizaci nemá žádný vliv. Lze jím pouze potvrdit správnost lokalizace, kdy se graf nachází stále na stejném místě.

Byly vytvořeny dvě skenované oblasti A a B. Oblast B je potomkem A, protože byla vytvořena lokalizací v A a doplněním grafu o dvě hrany. Skenování probíhalo za mírného šera kolem 20:00 hod. Obě oblasti dosáhly stejných výsledků při testování. Výsledky jsou dostupné v tabulce B.1. Zajímavé je zjištění, že lokalizace s vypnutým učícím módem je rychlejší, robustnější a ve všech případech změn dosáhla kladného výsledku. Lokalizace se zapnutým učícím módem dokázala najít pozici pouze ihned po uložení skenované oblasti. Bylo nutné absolvovat stejnou trasu jako při prvním skenování, než se zařízení lokalizovalo. Při změně osvětlení už byla lokalizace neúspěšná.

Lokalizace za vypnutého učícího módu se dokázala lokalizovat ve všech případech téměř ihned. Při odklizení věcí z pohovky (vyklizení regálu) trvala lokalizace přibližně 3 sekundy. Největší problém nastal při přesunu pohovky z původní pozice na místo grafu. Zařízení se při pohledu na scénu, kde najednou pohovka chyběla, nechtělo lokalizovat. Bylo nutné zajít do rohu místnosti a sledovat tu část scény, která podstoupila nejméně změn. Nakonec se ale zařízení lokalizovalo a graf byl umístěn na správném místě.

6.4 Navigace

Při navigaci je vidět vždy celá cesta. Aplikace momentálně nerozpoznává okolí do takové míry, aby určila jaká část grafu je uživateli viditelná a která část je například za stěnou, tak by neměla být vidět nebo by měla být odlišena jinou barvou. Toto platí i pro vytvořený graf. Ten je také vidět neustále celý. Do budoucna se počítá s implementací metody, která bude do jisté míry schopna určit, jaká část grafu či navigace je či není viditelná, podle toho bude AR objekty zobrazovat.

7 Závěr

Cílem práce bylo prozkoumat hardwarové a softwarové možnosti, které umožňují vytvoření a správu navigačních map vnitřního prostředí. Úkolem bylo navrhnout reprezentaci a výpočet trasy mezi dvěma body v 3D prostoru s ohledem na pohyb objektu nezanedbatelné velikosti. Ze získaných informací měla být navržena vhodná datová struktura dat pro efektivní algoritmické řešení hledání nejkratší cesty navigace ve skladovém prostředí. Výsledkem práce mělo být jednoduché demo, které zvládá skenování, zpracování a správu vnitřního prostředí. Bude umožněn i náhled z jiné perspektivy, aby bylo možné oblast zkontrolovat a případně upravit.

Výsledkem práce je průzkum zadaných oblastí. Byl vytvořen návrh potřebných struktur a algoritmů pro realizaci navigace ve skladovém prostředí. Navržené řešení bylo implementováno v podobě aplikace pro Google Tango. Tato aplikace v konečné fázi umí kontrolu přítomnosti požadovaného API v zařízení, správu vlastních skenovaných oblastí i oblastí aplikací třetích stran. Umožňuje skenování oblasti s náhledem navigačního grafu ve 3D a 2D. V obou perspektivách povoluje úpravu grafu. Dále program umožňuje navigaci na jakýkoliv zvolený navigační vrchol v grafu. Zhodnocení dosažených výsledků je popsáno v kapitole 6.

Do budoucna se počítá s dalším vývojem dané aplikace. Mezi možná rozšíření a úpravy lze zařadit: změna matice sousednosti na seznam sousednosti, import a export skenovaných oblastí, import a export navigačních grafů, propojení s externí databází, AR vizualizace dat z databáze, částečná viditelnost grafu a navigace (rozlišení částí, které jsou například za stěnou).

Literatura

[1] *The Best Scanners of 2015* [online]. www.3ders.org, 2015. [cit. 2017/04/09]. 3D printer and 3D printing news. Dostupné z: `http://www.3ders.org/articles/20151209-best-3d-scanners-2015.html`.

[2] *3D laserové skenování* [online]. Fakulta strojní, ČVUT, 2017. [cit. 2017/04/01]. Výuková skripta. Dostupné z: `http://lfgm.fsv.cvut.cz/data/vvt/s1/laserteorie3d.pdf`.

[3] Alliez, P. – Saboret, L. – Guennebaud, G. *Poisson Surface Reconstruction* [online]. doxygen, 2017. [cit. 2017/06/14]. Dostupné z: `http://doc.cgal.org/latest/Poisson_surface_reconstruction_3/index.html`.

[4] *Artec Eva 3D Scanner* [online]. Direct Dimensions, 2017. [cit. 2017/04/08]. Artec Eva 3D Scanner brochure. Dostupné z: `http://www.dirdim.com/pdfs/EVA_Brochure_DDI.pdf`.

[5] Bajer, L. *Algoritmy pro pathfinding* [online]. Matfyz, 2006. [cit. 2017/06/05]. Dostupné z: `http://ksvi.mff.cuni.cz/~brom/hagents/H-likeAgents4_Bajer060410.pdf`.

[6] Berger, M. et al. State of the Art in Surface Reconstruction from Point Clouds. *The Eurographics Association*. 2014, 1, 25, s. 25. Dostupné z: `http://lgg.epfl.ch/publications/2014/reconstar/paper.pdf`.

[7] Bernardini, F. – Rushmeier, H. The 3D Model Acquisition Pipeline. *Computer Graphics forum*. 2002, 21, 2, s. 149–172. Dostupné z: `http://www1.cs.columbia.edu/~allen/PHOTOPAPERS/pipeline.fausto.pdf`.

[8] Carter, J. et al. An Introduction to Lidar Technology, Data, and Applications. *NOAA*. 2012, s. 76. ISSN 202-2620. Dostupné z: `https://coast.noaa.gov/data/digitalcoast/pdf/lidar-101.pdf`.

[9] *Gocator 1300 Series* [online]. LMI Technologies, 2017. [cit. 2017/06/16]. Gocator - popis, specifikace. Dostupné z: `http://lmi3d.com/products/gocator/g1/1300-series/laser-point-profile-sensors#models`.

[10] *Area Learning* [online]. Google, 2017. [cit. 2017/06/03]. Dostupné z: `https://developers.google.com/tango/overview/area-learning`.

[11] *Coordinate Systems* [online]. Google, 2017. [cit. 2017/06/015]. Dostupné z: https://developers.google.com/tango/overview/coordinate-systems.

[12] *Tango Concept* [online]. GoogleATAP, 2017. [cit. 2017/04/27]. Tango API Documentation. Dostupné z: https://developers.google.com/tango/overview/concepts.

[13] *IIIDScan PrimeSense Review* [online]. 10TopTenReviews, 2017. [cit. 2017/04/08]. IIIDScan PrimeSense Review. Dostupné z: http://www.toptenreviews.com/computers/scanners/best-3d-scanners/iiidscan-review/.

[14] *iSense* [online]. 3dSystems, 2016. [cit. 2017/04/09]. iSense scanner guide. Dostupné z: https://www.3dsystems.com/sites/default/files/downloads/isense_flyer_5.5x8.5_en.pdf.

[15] *3D Systems iSense for iPad/iPhone 3D Scanner Review* [online]. tom's guide, 2015. [cit. 2017/04/09]. 3D Systems iSense for iPad/iPhone 3D Scanner Review. Dostupné z: http://www.tomsguide.com/us/3d-system-isense,review-2985.html.

[16] Kahzdan, M. – Bolitho, M. – Hoppe, H. Poisson Surface Reconstruction. *Eurographics Symposium on Geometry Processing*. 2006, s. 10. Dostupné z: http://hhoppe.com/poissonrecon.pdf.

[17] Kazhdan, M. *Surface Reconstruction* [online]. Symposiom on Geometry Processing, 2011. [cit. 2017/06/03]. Dostupné z: http://slideplayer.com/slide/7032445/.

[18] Kazhdan, M. – Hope, H. *Screened Poisson - Surface Reconstruction* [online]. Microsoft Reaserch and Johns Hopkins University, 2014. [cit. 2017/06/03]. Dostupné z: http://www.slideserve.com/fritzi/screened-poisson-surface-reconstruction.

[19] Maříк, V. – Štěpánková, O. – Lažanský, J. *Umělá Inteligence (1)*. Academica, 1993. ISBN 80-200-0496-3. S. 42.

[20] *Kinect for Windows Sensor Components and Specifications* [online]. Microsoft Corporation, 2017. [cit. 2017/04/03]. Kinect Documentation. Dostupné z: https://msdn.microsoft.com/en-us/library/jj131033.aspx.

[21] Pastorius, W. *Structured light vs. laser triangulation for 3D scanning and inspection* [online]. LMI, 2015. [cit. 2017/04/03]. Compare scanning technologies. Dostupné z: http://lmi3d.com/company/digital-hub/blog/structured-light-vs-laser-triangulation-3d-scanning-and-inspection.

[22] Patel, A. *Grids and Graphs* [online]. Red Blob Games, 2017. [cit. 2017/06/04]. Dostupné z: http://www.redblobgames.com/.

[23] *Plošné integrály* [online]. ČVUT, 2017. [cit. 2017/07/06]. List v prostoru R3 a jeho parametrizace. Dostupné z: http://euler.fd.cvut.cz/predmety/mta2/K611MA2_soubory/files/KS/PR6.pdf.

[24] *Ocular Specifications* [online]. Ocular Robotics, 2017. [cit. 2017/04/03]. REO5 3D LIDAR Scanner - Specifications. Dostupné z: http://www.ocularrobotics.com/products/lidar/re05/.

[25] *Rovinné grafy a barvení* [online]. Stanislav Palúch, 2017. [cit. 2017/06/20]. Dostupné z: https://frcatel.fri.uniza.sk/users/paluch/Prezentacie/GrafPrez_08.pdf.

[26] Russel, S. J. – Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003. ISBN 0-13-790395-2. S. 97–104.

[27] *3D skenery* [online]. ABBAS, a.s., 2017. [cit. 2017/04/01]. Dostupné z: http://www.skenovanive3d.cz/3d-skenery/.

[28] Strobl, K. et al. The Self-Referenced DLR 3D-Modeler. *Intelligent Robots and Systems*. 2009, s. 21–28. ISSN 978-1-4244-3804-4. Dostupné z: http://www.robotic.dlr.de/fileadmin/robotic/stroblk/publications/strobl_2009iros.pdf.

Přílohy

A Uživatelská příručka

V této kapitole bude popsán postup obsah CD, instalace a zásady používání.

A.1 Umístění a nároky

A.1.1 Umístění

Program je umístěn na CD, které je součástí dokumentu. Médium se nachází na přední desce v přihrádce. Obsahuje soubor `README.txt` a složky `Text`, `Program` a `Poster`. `README.txt` je stručný popis obsahu jednotlivých složek. `Text` obsahuje elektronickou formu psané práce. Ve složce `Program` jsou umístěny všechny komponenty potřebné pro spuštění aplikace. `Poster` obsahuje plakát se souhrnem práce.

A.1.2 Nároky na zařízení

Aplikace není ke stažení na *Google Play*. Je tedy nutné ji nainstalovat na zařízení ručně. Je důležité mít zařízení podporující technologii Tango. Aplikace podporuje verze od Androidu 4.4.2 KitKat do 6.0 Marshmallow.

A.1.3 Vlastní překlad

Pokud si chce uživatel nějaký způsobem aplikaci upravit nebo jen vytvořit nové, vlastní APK, může tak učinit překladem projektu, z `Program/Projekt` nebo stažením `https://github.com/dzejkob23/IndoorNavigation`, v prostředí Unity. Aplikace byla vyvíjena v `Unity 5.5.1f1 (64bit)`. Verze Tango SDK pro Unity byla `Eisa Unity5`, která již není ke stažení, ale nachází se na CD v `Program/TangoSDKUnityPackage`. Návod ke zprovoznění jakéhokoli projektu Tango v Unity je dostupný zde:
`https://developers.google.com/tango/apis/unity/`.

A.2 Instalace

Pro zahájení instalace je důležité mít na zařízení povolenou instalaci aplikací třetích stran. Pohyb v menu pro povolení instalace je následující:

1. Nastavení
2. Zabezpečení
3. Neznámé zdroje (povolit instalaci)

Postup se může lišit mezi různými verzemi systému Android. V dalším kroku je nutné přesunout APK soubor do úložiště zařízení. Jakmile je APK na zařízení, stačí ho spustit a rozběhne se instalační proces. Soubor APK je uložen na CD ve složce `Program/Apk`.

A.3 Průběh aplikace

Aplikace je nyní spustitelná ikonou s názvem `IndoorNavigation`. Po spuštění se zobrazí validační obrazovka, která kontroluje přítomnost a aktuálnost Tango Core v zařízení. Tlačítko `Start` provede vstup do aplikace.

A.3.1 Správa oblastí

Po vstupu do aplikace se uživatel dostane na scénu, která obsahuje výpis všech skenovaných oblastí viz obrázek 5.2. V pravém dolním rohu se nachází dvojice tlačítek

1. `Remove area description`
2. `New area description`
3. `Localize area`

Tlačítko (1) slouží k odstranění jednotlivých uložených popisů oblastí. Tlačítko (2) spustí vytváření nového popisu oblasti ve 3D prostředí. Tlačítko (3) spustí úpravu popisu ve vybrané oblasti, kterou je nutné předem vybrat v seznamu oblastí. Před přepnutím do režimu úprav ve 3D je nutné se v oblasti lokalizovat. Uživatel bude vyzván obrazovkou s hláškou **Walk around to relocalize.** aby se prošel po oblasti a lokalizoval se. V levém dolním rohu se nachází přepínač `Learning mode`. Ten spouští učící mód. Kombinace jednotlivých nastavení a jejich vliv na aplikaci je popsán tabulkou 4.1.

A.3.2 3D prostředí

Prostředí 3D obsahuje menu nastavení vlastností a strategie umisťování vrcholů grafu. Nad menu jsou dvě tlačítka. Prvním je `Change area`, které přepne pohled zpět na seznam uložených oblastí, a druhé je `Save area model`, které provede uložení oskenované oblasti do paměti. Menu se skládá ze dvou skupin přepínačů a ovládacích tlačítek viz obrázek 5.3. První skupina přepínačů obsahuje možnosti:

1. **Crossroad** - označuje křižovatku
2. **Warehouse position** - označuje skladovou pozici

Oba typy přepínačů se chovají stejně, jako rozcestník, mají jen ve 3D odlišnou barvu. Jsou to elementy předpřipravené pro další vývoj. Druhá skupina přepínačů umožňuje nastavení strategie umisťování a propojování vrcholů. Obsahuje možnosti:

1. **Create line with last created vertex,**
2. **Add/delete line between two vertices,**
3. **Create independent vertex.**

Volba (1) propojí hranou automaticky nový vrchol s posledním vytvořeným. Volba číslo (3) vytvoří samostatný vrchol bez jakékoliv automaticky vytvořené hrany. Volba (2) vytvoří nebo odstraní hranu mezi dvěma vrcholy v závislosti na jejím předchozím výskytu. Vždy je nutné označit a potvrdit oba vybrané vrcholy. Posloupnost akcí:

1. Označit první vrchol dotykem obrazovky.
2. Potvrdit označený vrchol tlačítkem `Select vertex`.
3. Dojde k barevnému odlišení vybraného vrcholu.
4. Označit druhý vrchol dotykem obrazovky.
5. Potvrdit označený vrchol tlačítkem `Select vertex`.

Pokud mezi vybranými vrcholy již hrana byla, bude zrušena, v opačném případě se vytvoří hrana nová. Posledním nezmíněným tlačítkem v menu je `Move to 2D map`. To přepne aplikaci do 2D perspektivy vytvořeného grafu.

A.3.3 2D prostředí

V tomto prostředí je pohled na graf z ptačí perspektivy, viz obrázek 5.4, je tak lépe vidět jeho rozloha. Součástí prostředí je také podpora gest. Přibližováním či oddalováním dvou prstů na obrazovce se mění velikost grafu. Tahem jedním prstem po obrazovce se mění pozorovaná oblast. Modrý kruh s bílou šipkou je indikátor aktuální pozice zařízení vůči grafu. V pravém horním rohu se nachází centrovací tlačítko. Je-li černé, indikátor pozice je vždy středem obrazovky. Je-li červené, indikátor není pevně svázaný s pohledem a pohybuje se po celé obrazovce. Stiskem centrovacího tlačítka dojde k vycentrování pohledu a indikátor bude opět středem obrazovky. Vrcholy grafu mohou nabývat různých barev:

1. **červená** - defaultní barva vrcholu
2. **šedá** - označený vrchol, cíl navigace
3. **zelená** - vybraný vrchol, odstranění/přidání hrany
4. **žlutá** - vrchol nejblíže k zařízení vzdušnou čarou

Prostředí 2D dále obsahuje ovládací menu, které zajišťuje možnosti přepnout se do 3D scény, spravovat hrany grafu nebo spustit navigaci. Seznam všech zobrazitelných tlačítek v menu:

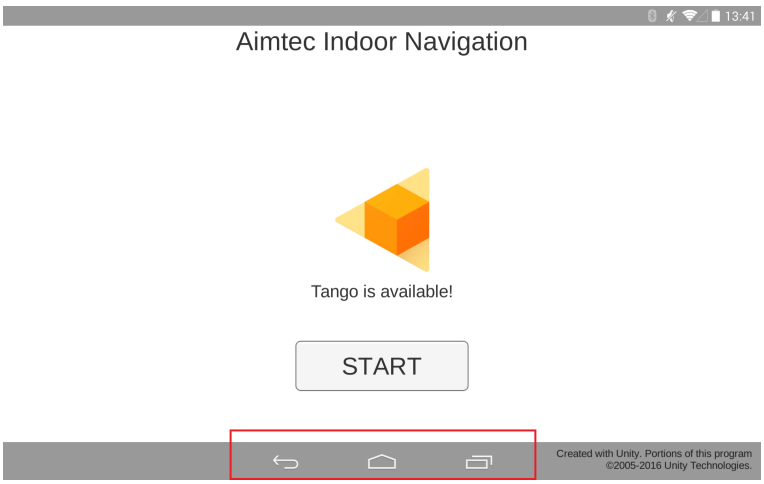
1. **Save area model** - uložení oblasti
2. **Move to 3D map** - přesun zpět do 3D perspektivy
3. **Add/del line** - přidat/odebrat hranu mezi vrcholy
4. **Select vertex** - vybrat označený vrchol
5. **Finish** - dokončení úprav grafu
6. **Navigate** - spuštění navigace

Tlačítko (1) uloží upravený navigační graf. Tlačítko (2) přepne aplikaci zpět do 3D scény. Uživatel se nemusí bát, že o dosud provedené změny přijde. Každá změna ve 2D se automaticky projeví i ve 3D. Tlačítko (3) přepne menu do režimu úprav, který skryje tlačítka (3) a (6) a zobrazí tlačítka (4) a (5), viz obrázek 5.5. Režim úprav slouží ke správě hran mezi vrcholy. Princip

a postup je stejný jako ve 3D. Tlačítko (4) zastává stejnou funkci, jako tlačítko `Select marker` v 3D scéně. Proto nemá cenu ho dále popisovat. Po dokončení úprav se tlačítkem (5) přepne menu z režimu úprav do defaultního režimu. Tlačítko (6) spustí navigaci ke zvolenému vrcholu.

A.3.4 Navigace

Po puštění navigace se spustí pohled přes kameru. Uživatel uvidí cestu k cíli ve tvaru navigačních šipek viz obrázek 5.6. Cíl je označen vrcholem grafu. V pravém dolním rohu se nachází dvě ovládací tlačítka `End navigation` a `Change area`. První tlačítko ukončí navigaci a přepne aplikaci do 3D prostředí s vizualizací navigačního grafu. Druhé tlačítko ukončí navigaci a přepne pohled na obrazovku se správou oblastí.



Obrázek A.1: Úvodní obrazovka aplikace s navigační lištou, která je vyznačena červeně. Zleva obsahuje *šipku zpět*, *domeček* a *kontextové menu aplikací*.

A.3.5 Ukončení aplikace

Aplikaci je možné opustit vyvoláním defaultní navigační lišty. Vyvolání se provede tahem prstu ze spodní hrany obrazovky směrem nahoru. Stiskem domečku bude aplikace opuštěna z jakékoliv obrazovky. Aplikace používá šipku zpět z navigační lišty pouze ke dvěma účelům. V prvním případě při neúspěšné lokalizaci a dále ve 3D, 2D a navigační scéně slouží k přesunu na scénu se seznamem skenovaných oblastí. Ve druhém případě v seznamu oblastí a na úvodní obrazovce slouží k ukončení aplikace.

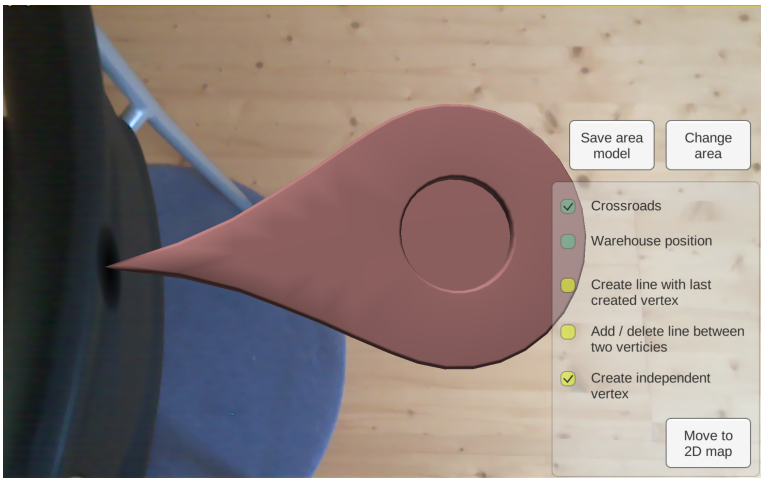
B Výsledky testování

B.1 Měření lokalizace za různých podmínek

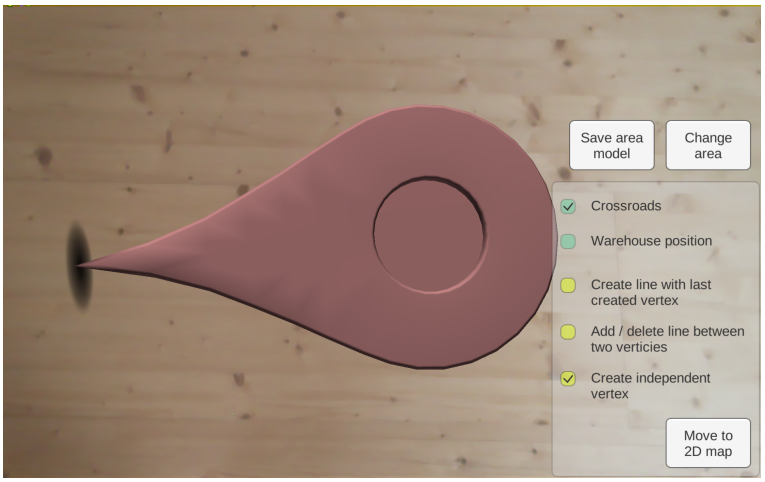
Čas lokalizace	**Learning mode**	**Lokalizace**	**Počet změn v oblasti**	**Osvětlení oblasti**
20:00 hod	Off	Ano (hned)	Žádné	Šero
20:00 hod	On	Ano (5s)	Žádné	Šero
00:30 hod	Off	Ano (hned)	Žádné	Umělé osvětlení
00:30 hod	On	Ne	Žádné	Umělé osvětlení
7:00 hod	Off	Ano (hned)	Žádné	Denní světlo
7:00 hod	On	Ne	Žádné	Denní světlo
7:00 hod	Off	Ano (2s)	Přesun menších objektů na místo zobrazení grafu.	Denní světlo
7:00 hod	On	Ne	Přesun menších objektů na místo zobrazení grafu.	Denní světlo
7:00 hod	Off	Ano (3s)	Odebrání krabic z pohovky. Simulace vyklizení regálu.	Denní světlo
7:00 hod	On	Ne	Odebrání krabic z pohovky. Simulace vyklizení regálu.	Denní světlo
7:00 hod	Off	Ano (10s)	Přesun pohovky na místo grafu. Simulace přesunu regálu.	Denní světlo
7:00 hod	On	Ne	Přesun pohovky na místo grafu. Simulace přesunu regálu.	Denní světlo

Tabulka B.1: Vliv různých změn v prostředí na lokalizaci aplikace.

B.2 Vrchol na pohyblivém objektu

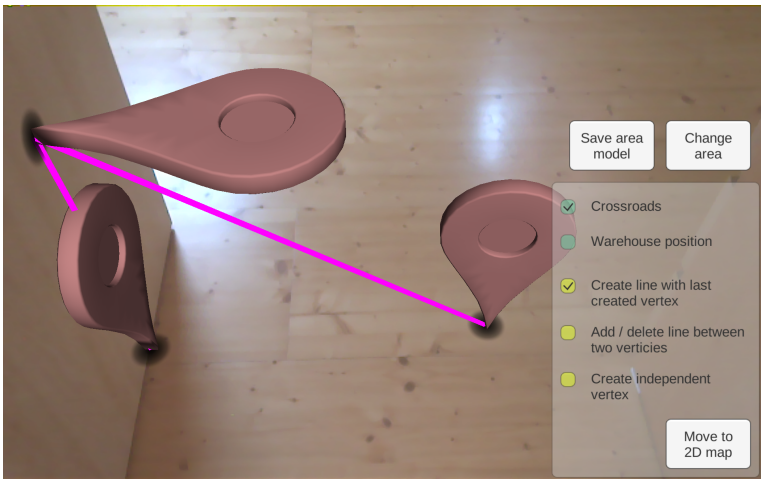


Obrázek B.1: Vrchol umístěný na pohyblivém objektu.

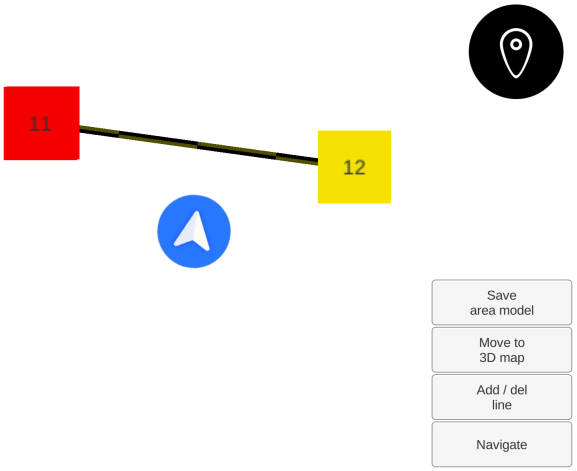


Obrázek B.2: Vrchol poté, co objekt změnil pozici.

B.3 Překrývající se vrcholy



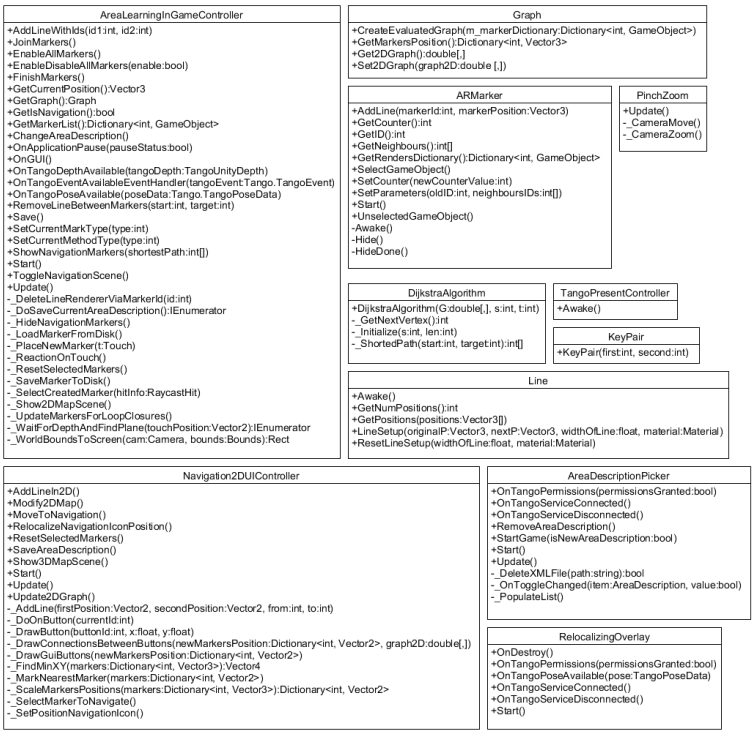
Obrázek B.3: Vrcholy, mající téměř stejnou hodnotu na dvou osách při pohledu ve 3D.



Obrázek B.4: Vrcholy, mající téměř stejnou hodnotu na dvou osách při pohledu ve 2D.

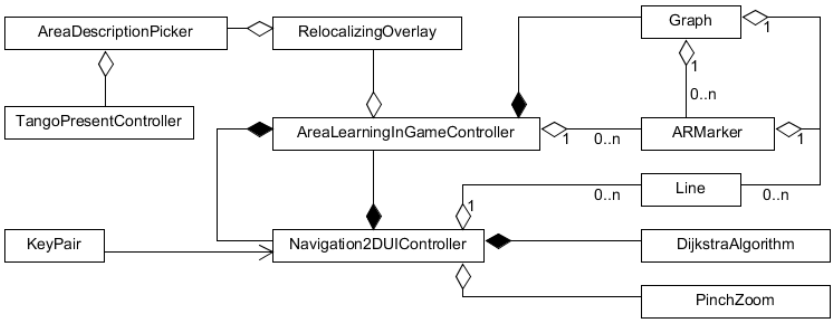
C Diagramy

C.1 Přehled tříd



Obrázek C.1: Přehled tříd s výpisem všech metod, které obsahují.

C.2 Class diagram



Obrázek C.2: Diagram tříd, které byly vytvořeny v rámci aplikace.