ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA STROJNÍ

Studijní program: **N0715A270012 - Průmyslové inženýrství a management**

Studijní specializace: **Bez specializace**

DIPLOMOVÁ PRÁCE

Možnosti navigace po skladu pomocí rozšířené reality

Autor: Bc. Jan Pavlíček

Vedoucí práce: Doc. Ing. Petr Hořejší, Ph.D

Akademický rok 2022/2023

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta strojní
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jan PAVLÍČEK**
Osobní číslo:	**S21N0029P**
Studijní program:	**N0715A270012 Průmyslové inženýrství a management**
Téma práce:	**Možnosti navigace po skladu pomocí rozšířené reality**
Zadávající katedra:	**Katedra průmyslového inženýrství a managementu**

Zásady pro vypracování

1. Úvod
2. Současná řešení navigace pomocí rozšířené reality (AR) se zaměření na průmyslové aplikace
3. Výběr vhodné softwarové a hardwarové infrastruktury pro realizaci ukázky AR navigace
4. Návrh řešení AR navigace
5. Realizace AR navigace
6. Zhodnocení, vývojové problémy, závěr

Rozsah diplomové práce: **50 – 70 stran**
Rozsah grafických prací: **0**
Forma zpracování diplomové práce: **tištěná**

Seznam doporučené literatury:

1. OKITA, Alex. *Learning C# Programming with Unity 3D*. Second edition. Routledge, 2019. ISBN 1138336815.
2. SUNG, Kelvin, SMITH, Gregory. *Basic Math for Game Development with Unity 3D: A Beginner's Guide to Mathematical Foundations*. Apress, 2019. ISBN 978-1484254424.
3. LINOWES, Jonathan. *Unity Virtual Reality Projects: Learn Virtual Reality by developing more than 10 engaging projects with Unity 2018*. 2nd Edition. Packt Publishing, 2018. ISBN 978-1788478809.
4. LaVALLE, Steven. M. *Virtual Reality*. Cambridge University Press, 2020. dostupné online na http://lavalle.pl/vr/
5. *Oficiální Unity3D návody dostupné na https://learn.unity.com/*

Vedoucí diplomové práce: **Doc. Ing. Petr Hořejší, Ph.D.**
Katedra průmyslového inženýrství a managementu

Konzultant diplomové práce: **Ing. Tomáš Macháč**
Katedra průmyslového inženýrství a managementu

Datum zadání diplomové práce: **19. září 2022**
Termín odevzdání diplomové práce: **26. května 2023**

L.S.

Doc. Ing. Vladimír Duchek, Ph.D.
děkan

Doc. Ing. Michal Šimon, Ph.D.
vedoucí katedry

V Plzni dne 19. září 2022

Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

V Plzni dne: ……. …….

podpis autora

ANOTAČNÍ LIST DIPLOMOVÉ PRÁCE

AUTOR	**Pavlíček**	**Jan**	
STUDIJNÍ PROGRAM	N0715A270012 Průmyslové inženýrství a management		
VEDOUCÍ PRÁCE	**Příjmení (včetně titulů)** Doc. Ing. Hořejší, Ph.D.	**Jméno** Petr	
PRACOVIŠTĚ	ZČU - FST - KPV		
DRUH PRÁCE	**DIPLOMOVÁ**	~~**BAKALÁŘSKÁ**~~	**Nehodící se škrtněte**
NÁZEV PRÁCE	Možnosti navigace po skladu pomocí rozšířené reality		

FAKULTA	strojní		**KATEDRA**	KPV		**ROK ODEVZD.**	2023

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	57		**TEXTOVÁ ČÁST**	52		**GRAFICKÁ ČÁST**	5

STRUČNÝ POPIS (MAX 10 ŘÁDEK) **ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY**	Diplomová práce obsahuje rozbor problematiky navigace s prvky rozšířené reality a možnosti její implementace do vnitřního skladu. Cílem práce bylo vytvořit prototyp aplikace, která by navigovala uživatele pomocí prvků rozšířené reality. Na začátku práce bylo nutné vytvoření znalostní základny pro následnou práci s programem na vývoj aplikací pracujících s rozšířenou realitou. Výsledkem práce je aplikace, vytvořená na míru pro sklad společnosti Outdoor Concept a.s., která navádí uživatele mezi jednotlivými regálovými pozicemi. Aplikace byla testována v prostředí skladu, výsledkem byla schopnost navést uživatele v 83 % případů.
KLÍČOVÁ SLOVA **ZPRAVIDLA JEDNOSLOVNÉ POJMY, KTERÉ VYSTIHUJÍ PODSTATU PRÁCE**	rozšířená realita, navigace, sklad, ARCore, AR Foundation, Unity, obrazové rozpoznávání okolí

SUMMARY OF DIPLOMA SHEET

AUTHOR	**Pavlíček**	**Jan**	
STUDY PROGRAMME	N0715A270012 Industrial Engineering and Management		
SUPERVISOR	**Surname (Inclusive of Degrees)** Doc. Ing. Hořejší, Ph.D.	**Name** Petr	
INSTITUTION	ZČU - FST - KPV		
TYPE OF WORK	**DIPLOMA**	~~**BACHELOR**~~	**Delete when not applicable**
TITLE OF THE WORK	Possibilities of Augmented Reality Warehouse Navigation		

FACULTY	Mechanical Engineering	**DEPARTMENT**	Industrial Engeneering and Management	**SUBMITTED IN**	2023

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	57	**TEXT PART**	52	**GRAPHICAL PART**	5

BRIEF DESCRIPTION **TOPIC, GOAL, RESULTS AND CONTRIBUTIONS**	The thesis contains an analysis of the issue of navigation with augmented reality elements and the possibility of its implementation in the warehouse. The aim of the thesis was to create a prototype of an application that would navigate users with augmented reality elements. At the beginning it was necessary to create a knowledge base for subsequent work with software for developing applications working with augmented reality. The result of the work is tailor-made application for Outdoor Concept's warehouse, that allows the user to be guided between shelf positions. The application was tested in a warehouse environment and the findings were that application is able to navigate user successfully in approximately 83 % of cases.
KEY WORDS	augmented reality, navigation, storage, ARCore, AR Foundation, Unity, image recognition

Obsah

Přehled použitých zkratek a symbolů

AR	Rozšířená realita
2D	Dvourozměrný
3D	Trojrozměrný
AI	Umělá inteligence
AoA	Angle of Arrival
API	Application Programming Interface
APK	Android Package
AR	Augmented Reality
BR/EDR	Bluetooth Basic Rate/Enhanced Data Rate
CMS	Systém správy obsahu
CNC	Počítačem řízené obrábění
CPU	Central Processing Unit
EAN	European Article Number
FBX	Filmbox
FPS	Snímky za sekundu
FPS	Snímková frekvence
GHz	gigahertz
GPS	Globální polohovací systém
GPU	Graphics Processing Unit
GUI	Grafické uživatelské rozhraní
HCI	Human-Computer Interaction
HMD	Head-mounted display
IAWN	Indoor Augmented reality Warehouse Navigation
IMU	Inerciální měrná jednotka
IoT	Internet of Things
IP	Internetový protokol
JSON	JavaScript Object Notation
BLE	Bluetooth Low Energy
LiDAR	Light Detection and Ranging
mAh	Miliamperhodina

Mbits/s	Megabits per second
MR	Mixed Reality
NavMesh	Navigation mesh
NFC	Near Field Communication
OS	Operační systém
OST	Optical See-Through
QR kód	Quick Response kód
RAM	Paměť s přímým přístupem
RF	Radiofrekvenční
RFID	Radio-Frequency Identification
RGB	Red Green Blue
RGBA	Red Green Blue Alpha
RSSI	Příjmový signálový indikátor
SDK	Software Development Kit
SLAM	Simultaneous Localization and Mapping
TDoA	Time-Difference of Arrival
TMP	TextMeshPro
TWR	Two Way Ranging
UI	Uživatelské rozhraní
UWB	Ultra Wideband
VNA	Very Narrow Aisle
VR	Virtuální realita
VST	Video See-Through
VZV	Vysokozdvižný vozík
WLAN	Wireless Local Area Network
XRGO	Extended Reality Goes On

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Úvod

V posledních letech se rozšířené reality věnuje velká pozornost, což je způsobeno především jejím zpřístupněním koncovým uživatelům a širokému spektru tvůrců. Řada vývojářů a vědců se snaží testovat aplikaci prvků rozšířené reality v nejrůznějších situacích, od aplikací určené pro běžné koncové uživatele – na příklad pomocník pro vizualizaci nábytku, nebo aplikace mířící do profesionální sféry, kde lze zmínit řadu výzkumů věnující se např. asistenci ve výrobě.

Jednou z oblastí, kterému se vývoj mimo jiné věnuje, je využití prvků rozšířené reality k navigaci uživatele. Navigaci pomocí rozšířené reality implementoval např. Google do *Google Maps* pro navigování ve venkovním prostředí v režimu chůze. Kromě aplikací ve venkovních prostorách vznikají i aplikace zaměřené na navigaci uvnitř budov. Mezi nimi lze jmenovat aplikaci *Situm*, která naviguje uživatele v prostorách letiště nebo nákupního centra. Podrobně je tomuto tématu věnována kapitola *2 Současná řešení navigace*.

Cílem této práce je ověření použití navigace pomocí prvků rozšířené reality v prostorách vnitřního skladu. Potenciál této oblasti tkví v současném trendu poptávky po skladování zboží a následného vychystávání v malých zásilkách (prodeje přes e-shop). Lidský faktor v této metodě vyskladnění hraje stále klíčovou roli a je proto důležité věnovat pozornost právě tomuto prvku, který je náchylný na chybovost. Jedna z možných komplikací, která může vznikat při vyskladnění je dezorientace a nenalezení optimální cesty k požadované regálové pozici. S tímto problémem by mohla v budoucnu pomoct navigace, která by usnadnila hledání cesty v „regálovém bludišti“.

Přístupů k tvorbě AR navigace je více, liší se použitým zařízením a způsobem, kterým je určována poloha zařízení v prostoru. Shrnutí technologických možností k určování polohy v rámci vnitřních prostor skladu je věnována kapitola 1. Prostředí skladu je, co se týče navigace a určování polohy zařízení, specifické především z toho důvodu, že se jedná o podnikové prostředí, které je částečně proměnlivé a pohybují se v něm například vysokozdvižné vozíky aj. Při návrhu aplikace je proto nutné počítat i s těmito prvky. Praktická část této práce je věnována tvorbě aplikace, která by uživatele navigovala v prostorách skladu. Aplikace je určena pro mobilní zařízení s operačním systémem Android, který obsahuje vývojářský balíček ARCore, umožňující chod AR aplikací na těchto zařízeních. Vývoj aplikace je uskutečněn ve vývojovém prostředí aplikace Unity, která je obohacena balíčkem AR Foundation, sloužícím pro tvorbu a správu AR aplikací. Před vývojové fázi projektu se věnují kapitoly 3 a 3.3.

Aplikace byla vyvinuta pro použití ve skladu společnosti Outdoor Concept a. s. a její funkce je omezena pouze na tento prostor, tj. dáno zvolenou metodou tvorby. V kapitole 4 Tvorba aplikace jsou popsány jednotlivé části aplikace od vytvoření kopie prostředí ve vývojové aplikaci Unity přes programové funkce jednotlivých kódů až po ukázku a popis uživatelského rozhraní aplikace, tak jak ji vidí uživatel.

Evalvaci vytvořené aplikace je věnována kapitola 5, která popisuje, jak probíhalo průběžné testování aplikace a jak byla v návaznosti na to upravována. Testování se věnovalo především přesnosti navigace a schopnosti navést uživatele do cíle. Následuje zhodnocení výsledků testů a celkového chování aplikace při používání v testovaném prostředí, z čehož jsou následně vyvozené poznatky toho, jak se řešení tohoto typu chová v prostředí skladu a jak je možné dále navázat na poznatky z této práce.

1. Rozbor řešené problematiky

V následujících podkapitolách bude popsána daná problematika rozšířené reality. Cílem této kapitoly je vytvoření povědomí o principu zobrazení rozšířené reality, její historii a také souhrn zařízení, na kterých je možné jí zobrazit.

1.1. Rozšířená realita

Rozšířená realita je koncept, který zobrazuje virtuální objekty v reálném prostředí.[1] Touto kombinací prvků se vytváří multidimenzionální prostor, jež rozšiřuje realitu. Rozšířenou realitu můžeme použít pro široké spektrum aplikací od rekreačního využití (hry) po profesionální nasazení. Výhodou rozšířené reality je možnost snadného zobrazení téměř jakéhokoliv objektu v takové podobě, která uživateli usnadní správně pochopit zobrazovanou situaci.

1.1.1. Princip technologie

Rozšířená realita (AR) se soustředí na znázornění informací, které přímo souvisejí s fyzickým prostředím. Virtuální realita (VR), oproti tomu, zobrazuje uživateli plně počítačově generované prostředí. AR jde nad rámec výpočtů zařízení, protože přemosťuje mezeru mezi reálným a virtuálním světem, a to prostorově a kognitivně. S rozšířenou realitou dokáže uživatel vnímat digitální informace jako součást reálného světa. [2]

Světově uznávanou definici pro rozšířenou realitu stanovil Azuma v roce 1997. Stanovuje základní tři požadavky které musí AR splnit, konkrétně: [3]

- Kombinace reálného a virtuálního
- Interakce v reálném čase
- Fungování ve 3D prostoru

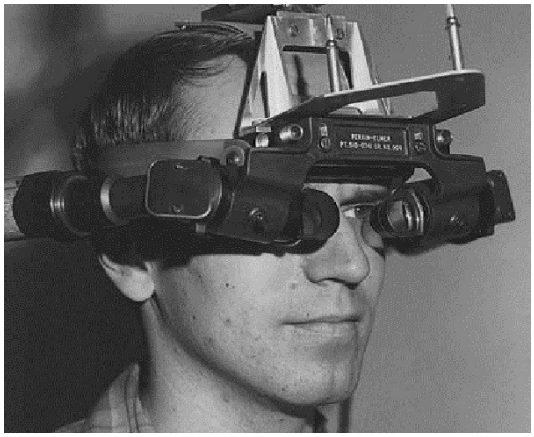
Z těchto požadavků vyplývá, že definice nezkoumá, jakým stylem je zprostředkován obraz pro uživatele, tedy zdali se jedná o displej v brýlích nebo mobilní telefon.

1.1.2. Historie

U zrození rozšířené reality stál Ivan Sutherland, který si ji v roce 1965 představoval jako *ultimate display* a popsal jej následovně:

„Ultimátní displej by byla místnost, ve které by počítač řídil existenci hmoty. Židle zobrazená v také místnosti by byla pohodlná k sezení. Pouta zobrazená v této místnosti by byla pevná a kulka zobrazená v této místnosti by byla smrtelná.[4]

Své myšlenky následně v roce 1968 Sutherland převedl do hmoty a vytvořil první AR systém „*head mounted*“ (volně přeloženo: montovaný na hlavu) viz Obr. 1.



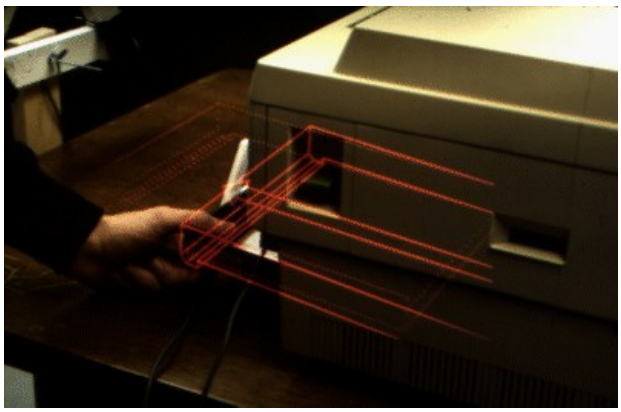
Obr. 1: První přístroj pro zobrazení 3D AR 1968 [5]

Přístroj byl tvořen soustavou skleněných čoček a hranolů, díky tomu byl velmi těžký a musel být připevněn ke stropu.

V osmdesátých letech dvacátého století probíhaly pokusy s videem, ve kterém figurovaly vrstvy naprogramované počítačem, které reagovaly na pozorovatele. Ve většině případů se jednalo o umělecké instalace.

Přelom nastal až v roce 1992 kdy se zrodil termín „*rozšířená realita*“, který se poprvé objevil v práci vědců Cudell a Mizell ze společnosti *Boeing*. Výsledkem bylo zařízení, které mělo sloužit jako asistence dělníkovi při sestavování kabelového svazku.

Dalším významným zařízením byla KARMA představená v roce 1993 Stevenem Fainerem. Tento systém byl schopen automaticky napovídat instrukce v sekvenci pro opravy zařízení. Uživatel měl před sebou vyobrazený úkon, který měl vykonat, detail na Obr. 2. Zobrazení probíhalo na prototypu brýlí s polopropustným zrcadlem, do kterého se odrážela promítaná nápověda z displeje nad čelem uživatele.

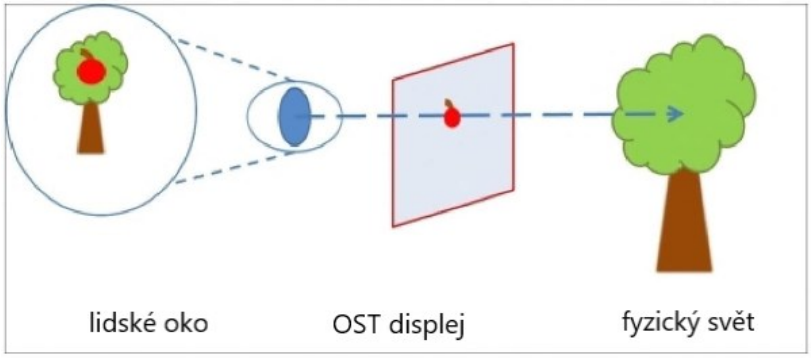


Obr. 2: Zobrazení nápovědy v KARMA, 1993 [6]

1.1.3. Zobrazení rozšířené reality

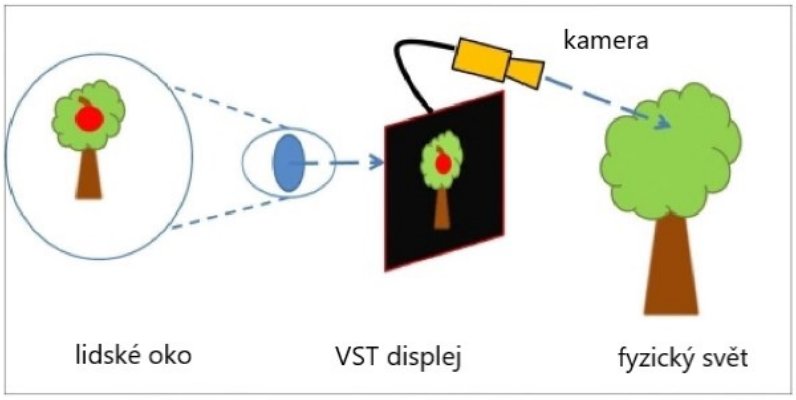
Jelikož rozšířená realita je poměrně volný pojem, je řada možností, jak zobrazit uživateli vytvořené digitální prvky.

Jednou z možností je metoda *optical see-through* (OST), v překladu opticky průhledný displej. Tato metoda používá částečně průhledný displej, na který je vyobrazen virtuální prvek. Složení rozšířené reality a okolního světa probíhá v lidském oku viz Obr. 3. [7].



Obr. 3: Princip zobrazení AR technologií optical see-through, vytvořeno na základě: [7]

Další metodou zobrazení AR je *video see-through* (VST) volný překlad: pohled skrz video. V tomto případě divák nesleduje přímo realitu, ale sleduje video monitor. Video zaznamenává realitu, do které se počítačově vkládá virtuální obsah. Princip sledování je znázorněn na Obr. 4.



Obr. 4: Zobrazení AR technologií VST, vytvořeno na základě: [7]

1.2. Současné technologie pro AR

Z kapitoly 1.1.3 je patrné, že existují dvě technologie zobrazení. Od toho se dále odvíjejí typy zařízení pro zobrazení AR.

Zařízení tedy můžeme rozlišit na dvě hlavní kategorie:

- mobilní telefony,
- chytré brýle (s polo průhledným displejem).

Popis těchto zřízení se nachází v následujících kapitolách.

1.2.1. Mobilní telefony

Virtuální realitu je možné v současnosti spustit na 86 % fungujících zařízeních s operačním systémem Android, která jsou aktuálně na trhu. Podmínkou je verze OS Android 7.0 nebo novější.[8] Zda je jednotlivé zařízení podporované a jaký je rozsah podpory AR, je možné zjistit přímo na stránkách společnosti *Google ARCore-devices*, která vydává operační systém Android.

V případě druhého nejpoužívanějšího operačního systému pro mobilní telefony iOS od společnosti Apple, jsou nutná následující kritéria: operační systém iOS 11 a k tomu musí být

zařízení osazeno procesorem A9 nebo novějším.[9] Podporovaná zařízení v případě iOS jsou vypsána na stránkách výrobce *Apple - Augmented reality*.

Tato práce se bude věnovat výhradně zařízením s operačním systémem Android, kvůli větší oblibě vývojářů (podpora komunity) a také možnosti využití, jak pro chytré brýle, tak pro mobilní telefony.

Mobilní telefony vytvářejí AR pomocí kamery a (neprůhledného) displeje, kde je kamerou zaznamenáno reálné okolí. Obraz je zobrazen na displeji, kde je do něj přidán virtuální člen, jako je znázorněno např. na Obr. 9. [10]

Obr. 5: Příklad využití AR na mobilním telefonu – hra Pokemon GO [10]

1.2.2. Chytré AR brýle

Na rozdíl od mobilních telefonů, kde je rozšířená realita pouze jako aplikace a telefony na ní nejsou stavěné, chytré AR brýle jsou pro rozšířenou realitu přímo navržené.

Velká část dostupných zařízení funguje na platformě Android nebo Windows, existují však i vlastní operační systémy jako např. *Nebula AR* od společnosti *Nreal*.

Konstrukční stránka brýlí je rozdílná, většinou se však jedná o poloprůhledný zorník, na který je promítán virtuální prvek. Brýle bývají doplněny o kamery, tak aby dokázaly snímat okolí a gesta uživatele. Ukázka toho, jak uživatel pracuje s virtuálním objektem zobrazeným v brýlích je na Obr. 6: AR brýle Microsoft Hololens 2 [11].

Obr. 6: AR brýle Microsoft Hololens 2 [11]

2. Současná řešení navigace

Navigace hrála v historii lidstva důležitou roli, například při plavbách po moři nebo dobývání území, byla správná navigace klíčová. Jinak tomu není ani dnes, kdy například potřebujeme najít cestu k nejbližší čerpací stanici nebo hledáme „správné dveře“ v nemocnici. Naše okolí se dnes mění rychleji, než kdy jindy a hledání té nejlepší cesty je často oříšek.

Navigaci můžeme obecně rozdělit na dvě kategorie:

- **Outdoor** navigaci (vnější prostředí)
- **Indoor** navigaci (vnitřní prostory)

Navigace pro obě skupiny je odlišná, zejména v možnosti použití senzorů pro určení přesné pozice. Senzory umožňující určení polohy jsou probrány v následující kapitole.

2.1. Technologie používané pro určování polohy

Pro navigaci je možné využít standardní senzory, kterými jsou osazeny chytré mobilní telefony nebo chytré brýle. Některé senzory dokážou přímo určit polohu, jiné jsou podpůrné nebo sbírají informace pro dopočtení polohy.

2.1.1. Radiofrekvenční systémy

Využívají k určení polohy radiofrekvenční vlny, které jsou vysílány mezi přijímačem a vysílačem. Přesnost těchto systému je závislá na použitém hardwarovém vybavení a konkrétní situaci, ve které je použit.

GPS

Global position system (globální poziční systém) byl původně vojenským systémem pro určování pozice pomocí radiového signálu z družic na oběžné dráze. Od roku 1978 jsou na oběžnou dráhu vysílány satelity pro tuto komunikaci, v současnosti je GPS nenahraditelně používáno napříč odvětvími od zemědělství po námořní dopravu.

Jedná se tedy o komunikaci mezi mobilním zařízením na zemi a satelitem na oběžné dráze. Aby bylo možné zjistit polohu zařízení na zemi je nutné znát *polohu satelitu* na oběžné dráze, *přesný čas vyslání signálu* ze satelitu a *přesný čas příjmu signálu* do zařízení. Takto určenou pozici je ještě nutné upřesnit triangulací s dalším satelitem a výsledkem je poloha přesná na jednotky metrů (udává se cca 5 m). [12]

GPS je vhodné používat zejména pro vnější aplikace, při vnitřním použití je signál oslaben konstrukcí budovy, dochází k prodlení signálu a výsledná poloha nemusí odpovídat realitě.

Alternativou k GPS jsou další družicové systémy jako např. Galileo (evropský systém), GLONASS (Ruská federace) aj.

GPS–RTK

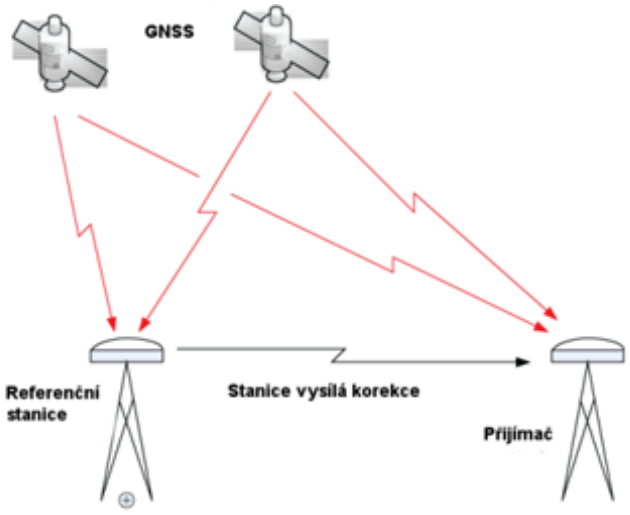
Někdy také GNSS RTK je zkratkou pro *Global Navigation Satelite System – Real Time Kinematic* (v překladu: globální navigační systém – kinematika v reálném čase). Pod tímto názvem se skrývá určení polohy pomocí signálu GPS a následné korekce v reálném čase.

K určení polohy je potřeba kromě satelitu i referenční stanici (statická), satelit a komunikační zařízení. Proces výpočtu vypadá následovně: referenční stanice přijme signál od satelitu, zpracuje jej a pošle jej spolu s korekčními údaji komunikačnímu zařízení. Komunikační

zařízení přijme signál jak z družice, tak i z referenční stanice. Tyto data jsou zaznamenána a jejich diferenciací je vypočtena poloha zařízení. Tento princip je vyobrazen na Obr. 7.

GPS RTK umožňuje vypočtení přesnosti od 20 cm do 1 m. [13]

Stejně jako GPS, trpí GNSS RTK zeslabením až ztrátou signálu ve vnitřních prostorách, proto se také hodí spíše pro vnější aplikace.



Obr. 7: Princip určení polohy GNSS-RTK [14]

RFID

Radio Frequency Identification, v překladu radiofrekvenční identifikace je princip komunikace mezi *readerem* (čtečkou) a *tagem* (štítkem). Komunikace probíhá pomocí radiových vln o různých frekvencích, podle toho se dají rozdělit do kategorií vyobrazených v Tab. 1.

Tab. 1 RFID frekvenční pásma [15]

kategorie	frekvenční pásmo	dosah
Nízko frekvenční RFID systémy	30-500 kHz, obvykle 125 kHz	až 15 cm
Vysoce frekvenční RFID systémy	3-30 MHz, obvykle 13.56 MHz	až 1,5 m
Ultra vysoce frekvenční RFID systémy	300-960 MHz, obvykle 433 MHz	až 10 m
Mikrovlnné RFID systémy	2.45 GHz	až kilometry

Dalším důležitým aspektem je, zda je RFID tag napájen vlastním zdrojem energie (baterie, z el. sítě aj.) nebo získává energii anténou z vysílaného signálu čtečky. Z toho plyne rozdělení na:

- RFID Tag **pasivní** (přímá el. energii ze signálu)
- RFID Tag **aktivní** (má vlastní zdroj el. energie)

Čtečka může mít mnoho podob, může se jednat o periferii připojenou k mobilnímu telefonu pomocí Bluetooth nebo kabelem. Integrovaná může být i součástí „ručního skeneru“.

Systém vyhodnocuje pro získání polohy z RFID tagů následující veličiny: sílu přijímaného signálu, úhel, po kterém signál přijde do zařízení, čas příchodu a časovou známku signálu.[16]

Výstupem komunikace čtečky a tagu je přesná identifikace daného tagu, tedy jeho číselný kód. Pokud máme databázi tagů s přiřazenou polohou, získáme načtením jednoho z nich přesný výstup polohy zařízení. Podmínkou správného určení polohy je čtení tagů na úměrnou

vzdálenost, tak aby nedocházelo ke čtení nechtěných tagů. RFID je možné využít ve vnitřních i venkovních prostorech.

Wi-Fi

„*Wireless Fidelity*“ je standardizované rozhraní pro výměnu dat, pracující na principu radiových vln. Frekvence pro komunikaci je 2.4 GHz a 5 GHz. Wi-Fi se nejvíce používá pro výměnu dat v rámci internetového připojení k síti pro tuto komunikaci máme výraz WLAN.

Wi-Fi je opatřena několika standardy komunikace, které se neustále vyvíjí. V současnosti je nejnovějším standardem IEEE Wi-Fi 802.11ax s pořadovým označením Wi-Fi 6, pracující v pásmu 2,4 nebo 5 GHz s maximální přenosovou rychlostí 10 500 Mbit/s. [17]

Určení polohy Wi-Fi je možné obdobně jako v případě GPS, kde je sledována zpětná vazba od Wi-Fi přístupových bodů rozmístěných po budově. Pokrytí Wi-Fi signálem musí být dostatečné, tak aby ve všech místech byl signál z vícero přístupových bodů. To je z důvodu výpočtu polohy, při kterém se používají data z více přístupových bodů. Pro výpočet polohy se používají metody *RSS fingerprinting*, *triangulation* nebo *trilateration*. Přesnost takto určené polohy je v rozmezí 2-5 metrů v závislosti na použitých zařízeních a výpočtech.[16]

Bluetooth

Bluetooth je název pro protokol radiové komunikace mezi zařízeními. Bylo navrženo pro spojení a přenos dat na kratší vzdálenost. K tomu využívá frekvenci v rozsahu 2,400 GHz až 2,483.5 GHz.

Bluetooth zaštituje dvě technologie, konkrétně *Bluetooth Low Energy* (BLE) a *Bluetooth Classic* (více známé jako *Bluetooth Basic Rate/Enhanced Data Rate*, zkráceně BR/EDR).

Bluetooth Low Energy (překlad: Bluetooth s nízkou energií) se vyznačuje zejména tím, že je velmi úsporný, co se týče spotřeby energie. Principem snížení energetické náročnosti je rozdělení vysílacího signálu a vzniku intervalu mezi jednotlivými signály. Tento princip se využívá např. pro *Bluetooth Beacon* (v překladu maják). Jedná se o zařízení, které pravidelně vysílá krátký signál. Na základě tohoto signálu, který je identifikován zařízením v dosahu, je určena poloha zařízení.[18]

Dosah Bluetooth je v závislosti na třídě, tabulka tříd je v Tab. 2.

Tab. 2 Třídy zařízení Bluetooth [19]

Třída	Spotřeba energie	Max. síla záření	Operační vzdálenost	Příklad zařízení
1	vysoká	100 mW (20 dBm)	až 100 m	USB adaptéry, routery
1,5	středně vysoká	10 mW (10 dBm)	až 30 m, obvykle do 5 m	beacon, nositelné senzory
2	střední	2,5 mW (4 dBm)	až 10 m	mobilní zařízení, adaptéry, chytré čtečky karet
3	nízká	1mW (0 dBm)	do 1 m	bluetooth adaptéry

Z Tab. 2 vyplívá, že zařízení jsou schopná přenášet informace na poměrně velkou vzdálenost, čehož lze využít při zjišťování polohy uvnitř budov. Pro zjištění polohy se využívá soustava majáků (Bluetooth Beacon), které pokrývají vnitřní prostor svým signálem. Při použití Bluetooth je nutné dbát na dostatečné pokrytí, protože signál slábne průchodem skrz překážky. Přesnost těchto systémů závisí na použitých hardwarových prvcích a prostředí, obecně lze předpokládat přesnost určení polohy v rozsahu 2-5 metrů. [16]

UWB

Ultra-wideband (UWB) v překladu: Ultra širokopásmové radiové vlny, jsou komunikační technologií určenou pro krátké vzdálenosti s rychlým a stabilním přenosem dat.

Technologie se vyznačuje především rychlým přenosem dat, nízkými pořizovacími náklady, malou spotřebou energie a vysokou přesností.

Přenosová frekvence není přesně stanovena a je možné využít pásmo mezi 3,1−10,6 GHz. Přenosová rychlost dosahuje až 480 Mbits/s. Dosah signálu je 10–200 metrů v závislosti na aplikaci.

Při určování se obdobně jako v případě Wi-Fi nebo Bluetooth postupuje několika způsoby, kdy se buď měří čas mezi vysláním a příjmem, odezva mezi zařízeními nebo úhel přijímaného signálu. Z toho pak vycházejí metody pro výpočet polohy zařízení *Time-difference of Arrival* (TDoA – překlad: časový rozdíl příchodu), *Two-way Ranging* (TWR – překlad: dvoucestné vytyčování) a *Angle of Arrival* (AoA – překlad: úhel příchodu). V případě technologie UWB lze dosáhnout přesnosti určení polohy v rozmezí 0,1-0,5 metru, záleží však na použitých zařízeních a softwarové. [20]

2.1.2. Obrazové rozpoznání

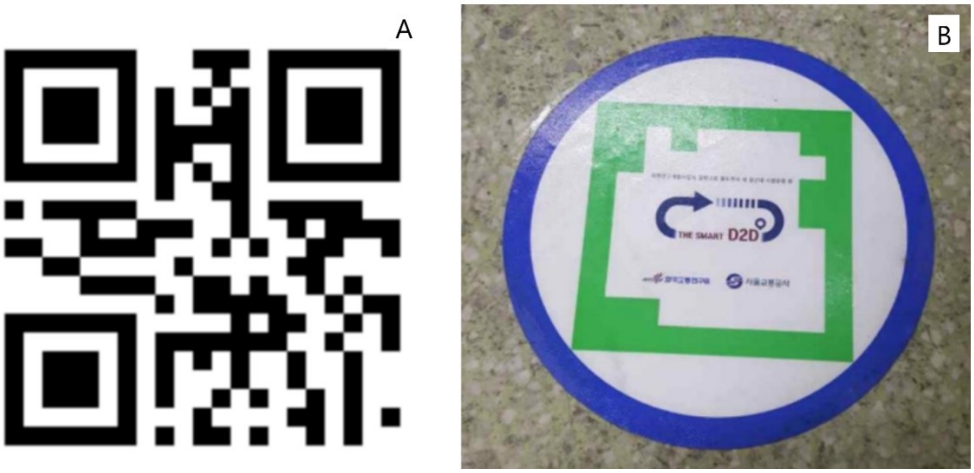
Video kamera v telefonu je tak, jako lidské oko klíčová k rozpoznávání 2D a 3D objektů v okolí uživatele. Jednotlivé objekty lze propojit se souřadnicí pozice a tím získat údaj o poloze zařízení. Tento princip strojového vnímání okolí je nazýván *Visual recognition*, více v následujících odstavcích. Obecně lze dělit obrazové rozpoznávání na:

- **marker based** – založené na rozpoznávání značek (markerů),
- **marker-less** – nepotřebují značky, snímají okolí jako celek.

Následuje popis jednotlivých metod:

Marker based

Jednou z možností je sledování „Markers“ (překlad: značky). Jedná se o 2D objekty, které pomocí kamery dokážou zařízení přesně identifikovat. Nejznámější podobou je QR kód, uvedený je na Obr. 8 část A. Je možné použít i speciální značky. Speciální značka použitá pro navigaci ve vnitřním prostoru je na Obr. 8 část B.



Obr. 8: část A: QR kód vygenerovaný na ZXing.com HELLO, část B: Speciální marker pro navigaci, zdroj:[21]

Princip určení polohy pomocí markerů je podobný jako v případě čtení RFID tagů „nablízko“, uživatel načte v tomto případě pomocí kamery marker, zařízení zjistí kód, spojí jej s polohou uloženou v databázi a tím je získána poloha zařízení.

Limitací pro tento způsob určení polohy je identifikace markeru, ten musí být dostatečně veliký a čitelný (nesmí být znečištěn nebo porušen, to by vedlo k chybnému vyhodnocení). Obecně je možné použití jak v interiéru, tak exteriéru. Je však opět nutné správně vyhodnotit požadovaný počet markerů pro přesnou navigaci.

Podobnou metodou je rozpoznávání obrázků: *Image recognition*, tato metoda rozpoznává vzory obsažené v obrázcích a porovnává je s databází.

Marker-less

Další možností je sledování okolního prostoru jako celku, tedy převod fyzického prostředí na digitální neboli S*patial mapping*. To lze provést řadou způsobů, nejčastěji se jedná o převod na mračno bodů nebo geometrické obrazce (trojúhelníky, čtverce, aj.).

V případě rozpoznávání prostoru lze systémy rozdělit na dvě technologie:

- **Rozpoznávání ploch a mračna bodů** (*AR Foundation*, *Azure Spatial Anchors*) – poloha je určena na základě počáteční polohy a snímání pohybu vůči okolí
- **Area Targets** (*Vuforia*) – poloha je určena na základě porovnání obrazu okolí s 3D skenem prostředí

V případě používání Area Targets je nejdříve nutné vytvořit 3D sken prostředí, ve kterém se bude navigovat. Jedním z možných softwarů je aplikace *Vuforia Generator*. Uživatel do ní nahraje data z 3D skeneru (*Matterport, Leica, Vuforia Target Creator App* pro iOS zařízení s LiDar[1]) a aplikace vytvoří digitální kopii prostředí s integrovanými targety[2]. Při spuštění aplikace na zařízení jsou v okolí vyhledávány shody s vygenerovanými targety, na základě kterých je pak určena přesná poloha zařízení. [21]

[1] LiDar (Light Detection And Ranging – překlad: světelná detekce a měření rozsahu): technologie měří odraz laserových paprsků od okolí, na základě nichž je vytvořena 3D mapa okolních ploch.
[2] Target – neboli cíl, tímto pojmem je označen objekt, který je možné dohledat v prostoru na základě jeho tvaru, označení se používá i v případě navigace pro cílovou pozici.

V případě určování polohy pomocí rozpoznání ploch je možné použít např. AR Foundation (ARCore), odlišnost je zejména v tom, že poloha není určována přímo z obrazu okolí, ale je dopočtena z výchozí polohy a snímaného relativního pohybu okolím. To je provedeno na základě detekce ploch a rozkladu prostředí na mračno bodů, ukázka ploch je na Obr. 9. Z toho vyplívá riziko, že zařízení může vyhodnotit pohyb špatně a z toho vznikne odchýlení vypočtené polohy od reálné. Pro navigaci je nutné vytvořit v programu Unity zjednodušený zrcadlový model reálného prostředí, podle kterého aplikace dopočte pozici. K tomu je nutné označit výchozí bod, ze kterého se uživatel bude pohybovat.

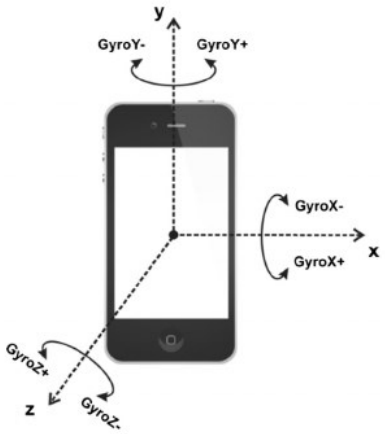
Obr. 9: AR Foundation ukázka nalezených ploch [22]

2.1.3. Podpůrné technologie

Podpůrné technologie podporují ostatní funkce, které přímo získávají informace o poloze zařízení. Jsou důležité zejména v případě výpadku nebo pro doplnění potřebných informací.

Gyroskop

Gyroskop slouží k určení relativního náklonu zařízení ve třech osách. Znázornění sledovaných os je znázorněno na Obr. 10.



Obr. 10: Osy chytrého telefonu sledované gyroskopem [23]

Jedná se o důležitý prvek pro zkoumání polohy zařízení, protože díky němu lze např. zjistit, zda se načtený kód nachází v přímé blízkosti před uživatelem nebo v dálce podle úhlu čtení kódu.

Kompas

Kompas je jednou z nejstarších pomůcek pro navigaci a podstatnou roli hraje i v současných mobilních zařízeních. Kompas je naváděn magnetickým polem země a díky němu je zařízení schopné určit směr natočení vůči ose rotace zeměkoule. Směr, kterým uživatel zařízení směruje je důležitý pro to, aby navigace správně sdělila uživateli směr jeho další cesty do cíle.

LiDAR

Jedná se zkratku *Light Detection and Ranging*, v překladu světelná detekce a měření rozsahu. Principem technologie je vysílání dávek laserových paprsků do okolí a detekce jejich odrazu od okolních objektů, na základě časového odstupu mezi vysláním a příjmem paprsku je určena vzdálenost. Tímto principem je vytvořena 3D mapa bodů okolního prostoru.[24]

Technologie je využívána 3D skenery (*Matterport, Leica*, aj.), nově je implementována i do nejvýkonnějších mobilních telefonů *Apple* např. iPhone 13 Pro.

Akcelerometr

Toto zařízení zaznamenává sílu zrychlení a její směr. Obdobně jako v případě gyroskopu na Obr. 10, akcelerometr rozkládá zrychlení do osového systému XYZ. Akcelerometr je užitečný hardwarový prvek při sledování, jakým směrem se zařízení pohybuje (údaje o zrychlování a zpomalení vůči osám zařízení).[25]

2.2. Vývojové prostředky pro tvorbu AR aplikací

Vývoj aplikací pro rozšířenou realitu lze provádět v různých softwarech, v této práci byl vývoj prováděn v aplikaci Unity. V této aplikaci bylo využíváno doplňku AR Foundation, který umožňuje správu rozšířené reality v Unity. Tato kapitola se věnuje uvedení daného vývojového softwaru a na to navazujících prvků.

Unity

Unity je engine[3] určený primárně pro tvorbu multiplatformních 2D a 3D her a zároveň podporuje tvorbu AR a VR aplikací. Unity je volně dostupný, díky tomu je široká základna tvůrců a podpory. Unity rovněž nabízí sérii kurzů sestavených tak, aby uživatele provedli kompletní tvorbou her a aplikací v enginu. Unity je určeno pro profesionální použití k tvorbě her a aplikací, ale díky tomu, že volně dostupné vzdělávací kurzy jsou velice přívětivé, je vhodný i pro začínající tvůrce.

Unity se skládá z velkého množství modulů pro kontrolu a renderování 3D objektů, osvětlení scény, kamer snímajících scénu, XR kamer a dalších prvků používaných v AR aplikaci. Výhoda Unity programu je, že aplikace lze v programu, jak vyvíjet, tak je testovat (pouze částečně např. generování naváděcí linie). Unity umožňuje tvorbu pro webové rozhraní, PC, mobilní zařízení s Androidem nebo iOS nebo pro konzole a VR a AR platformy (chytré brýle).

V následujících odstavcích budou rozebrána důležitá rozšíření a součásti Unity, která byla použita při tvorbě aplikace.

[3] engine: software, sloužící k vývoji videoher nebo aplikací (v tomto případě AR aplikace)

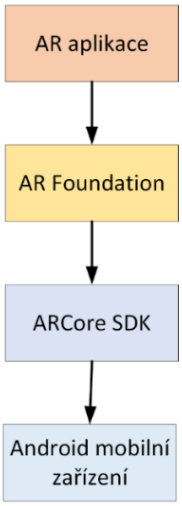
NavMesh

NavMesh je základní součástí Unity a je využíván k navigaci a hledání cest. Pomocí tohoto navigačního systému je možné tvořit postavy, které se inteligentně pohybují prostředím, na základě kterého je generovaný *NavMesh*.[26]

Tento prvek je možné využít pro generování navigační čáry mezi dvěma body na vygenerované ploše *NavMesh*.

AR Foundation

AR Foundation je balíček, který umožňuje tvorbu multiplatformních aplikací s rozšířenou realitou (AR) v Unity. Samotné AR Foundation nevytváří žádné prvky rozšířené reality, pouze je zprostředkovává. K implementaci AR prvků je nutné použít subsystém ve formě AR SDK[4] balíčku. Princip zprostředkovaného chodu AR aplikace na platformě Android je na Obr. 11.



Obr. 11: Funkční propojení komponent pro chod AR aplikace

AR Foundation umožňuje tvorbu aplikací jak pro platformu Android (SDK-ARCore), tak pro platformu iOS (SDK-ARKit).

[4] SDK: software development kit – v překladu: sada vývojových nástrojů

Funkce, které podporuje AR Foundation:

- **Sledování** (*Tracking*) – sledování zařízení (sledování polohy a orientace zařízení v prostoru), sledování obličeje, 2D obrázky a 3D objekty, jejich sledování a sledování lidského těla.
- **Detekce ploch** (*Plane detection*) – rozpoznání horizontálních a vertikálních ploch.
- **Kotva** (*Anchor*) – libovolný bod (včetně rotace) v prostoru který zařízení sleduje.
- **Odhad světla** (*Light estimation*) – odhad teploty světla a intenzity v okolním prostředí.
- **Průzkum prostředí** – generování mapy krychlí k reprezentaci části fyzického prostředí.
- **Síťování** (*Meshing*) – generuje trojúhelníkovou síť, odpovídající fyzickému prostředí.
- **Spolupracování účastníků** (*Collaborative Participants*) – sledování pozice a rotace ostatních zařízení ve sdíleném AR prostředí.
- **Rentgenový přenos** (*Raycast*) – jedná se o snímek, který je vyslán ze 3D nebo 2D objektu, pohybující se určitým směrem.
- **Průchozí video** (*Pass-through Video*) – optimalizované renderování obrazu z kamery zařízení na displej jako pozadí pro AR aplikaci.

ARCore SDK

Jedná se o SDK, který je určen pro tvorbu aplikací, pracujících na platformě Android. Sadu je možné nainstalovat do Unity pomocí ARCore *XR Pluginu*. ARCore je produktem společnosti Google, která jej publikovala v roce 2014 jako projekt Tango. Tento název se v roce 2018 změnil na ARCore.

Základní tři funkce umožňující kombinaci fyzického a virtuální světa jsou [27]:

- **Sledování pohybu** (*Motion tracking*) – umožňuje sledování zařízení na základě analýzy prostředí.
- **Rozklíčování prostředí** (*Environmental understanding*) – umožňuje zařízení detekovat velikost a polohu ploch v jeho okolí: vertikální, horizontální a plochy pod úhlem.
- **Odhad světla** (*Light estimation*) – umožňuje zařízení vyhodnocovat současnou situaci světla v okolí zařízení.

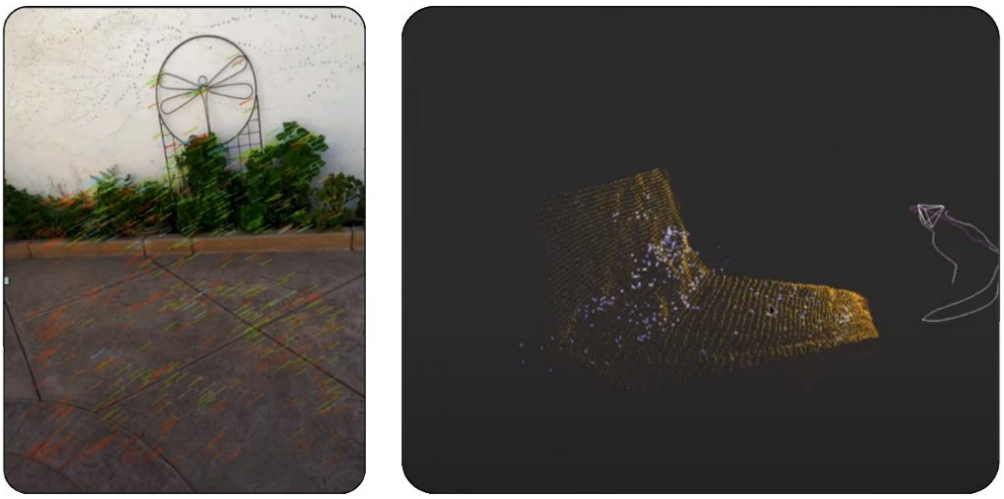
Pomocí těchto základních funkcí lze tvořit aplikace s rozšířenou realitou, které jsou schopné sledovat okolí, změnu polohy zařízení v prostoru a umisťovat virtuální prvky do obrazu prostoru v mobilním zařízení.

Klíčový prvek pro správné fungování aplikace je využití funkce sledování okolí a vnímání pohybu zařízení vůči okolnímu prostředí. To je zajišťováno základní funkcionalitou ARCore, která je velmi důležitá pro chod AR aplikací.

Základní nástroje, pomocí kterých ARCore zpracovává okolí jsou následující[28]:

- **Depth Blending** (míšení hloubek) – škálování obrazu podle vzdálenosti bodů (objektů).
- **Feature Tracking** (sledování objektu) – snímaný obraz je proložen body, na základě jejich pohybu a změny polohy se dopočítává relativní poloha zařízení. Mračno bodů jako interpretace obrazu je na Obr. 12, v levé části je záběr kamery na prostředí a v pravé je výsledný převod obrazu na body. Jehlan s vyznačenou trajektorií je záznam relativní polohy a úhlu zařízení na základě sledovaných bodů (žluté body představují snímanou hloubku obrazu (Depth API), modré body jsou mračno bodů (Point cloud API).
- **Inertial Measurement Unit – IMU** (setrvačná měrná jednotka) – jedná se o kombinaci měření pomocí gyroskopu a akcelerometru – souhrnně dodává informaci o pohybu (směr, zrychlení) zařízení, slouží k doplnění nebo nahrazení obrazového vnímání (dopočtení pohybu), v případech, kdy je získaný obraz rozmazán pohybem zařízení.

Dalším doplňujícím prvkem pro zpřesnění je integrace *machine learning*, který v případě zakrytí nebo výpadku obrazu, dokáže po opětovném získání obrazu dopočíst aktuální pozici zařízení a zobrazovaného AR obsahu díky kombinaci s dalšími senzory: kompas a gyroskop.



Obr. 12: ARCore Feature tracking – převod obrazu na mračno bodů [28]

2.3. Současná řešení pro navigaci

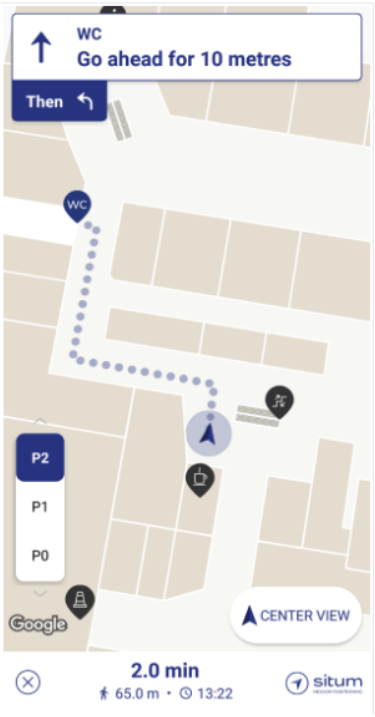
Pro navigaci existuje řada aplikací a služeb, které se věnují široké škále oblastí. Tato práce se zaměřuje na aplikace, které mají potenciál nebo jsou přímo navrženy pro profesionální využití v sektoru skladování. Jejich popis a rozbor funkce je v následujících odstavcích.

2.3.1. Situm – indoor positioning

Jedná se o aplikaci pro mobilní telefony, která dokáže uživatele navigovat nebo sledovat jeho pohyb po budově i mezi budovami. Projekt vznikl v roce 2015 ve Španělsku a jeho zakladatelé jsou Gregory Botanes, Víctor Álvarez, Adrián Canedo a Cristina Gamallo, od té doby bylo řešení aplikováno na více než 6000 budov. Aplikace se soustředí především na navigaci po rozsáhlých budovách jako např. nemocnice, letiště, nákupní centra, aj. Kromě toho je ale možné využití i pro sledování **pohybu pracovníků** nebo **vysokozdvižných vozíků** v reálném čase.

Systém pracuje na bázi sledování signálu Wi-Fi a Bluetooth Beacon, které jsou rozmístěny po daném objektu. K tomu je vytvořena velkoformátová detailní fotografie podlahy, podle které se mobilní telefon orientuje kamerou. [29]

Systém je tvořen dvěma prvky: webovou aplikací, která slouží pro přehled a správu systému a aplikací v mobilním zařízení. Ta je určena pro samotnou navigaci nebo sledování pozice. Podoba mobilní aplikace je znázorněna na obrázku Obr. 13, kde je uživatel naváděn na nejbližší WC.



Obr. 13: Prostředí mobilní aplikace Situm – navigace [29]

Pokud se společnost rozhodne pro její použití, postup integrace se řídí předem stanoveným procesem. Na začátku je nutné vytvořit mapu objektu, včetně současných přístupových bodů (*Wi-Fi* nebo *Bluetooth*). Pokud je počet přístupových bodů nedostatečný, je nutné jejich doplnění. Mapu je poté nezbytné validovat pomocí mobilního telefonu. To znamená projít celý objekt, aby se ověřila podoba prostředí a dosahy jednotlivých přístupových bodů.

Rozmístění Wi-Fi modulů a Bluetooth Beacon (majáků) je nutné udělat tak, aby bylo dostatečně husté a pokrývalo celou budovu. Příklad rozmístění přístupových bodů (barevné puntíky) v objektu zobrazeného v prostředí webové aplikace Situm je na Obr. 14.

Obr. 14: Situm mapa přístupových bodů ve webové aplikaci [29]

2.3.2. Sewio

Jedná se o novou českou společnost specializující se na vnitřní navigaci a vnitřní sledování pohybu materiálu, věcí i osob **v průmyslovém prostředí**. Má za sebou řadu aplikací v různých typech podniků a případů.

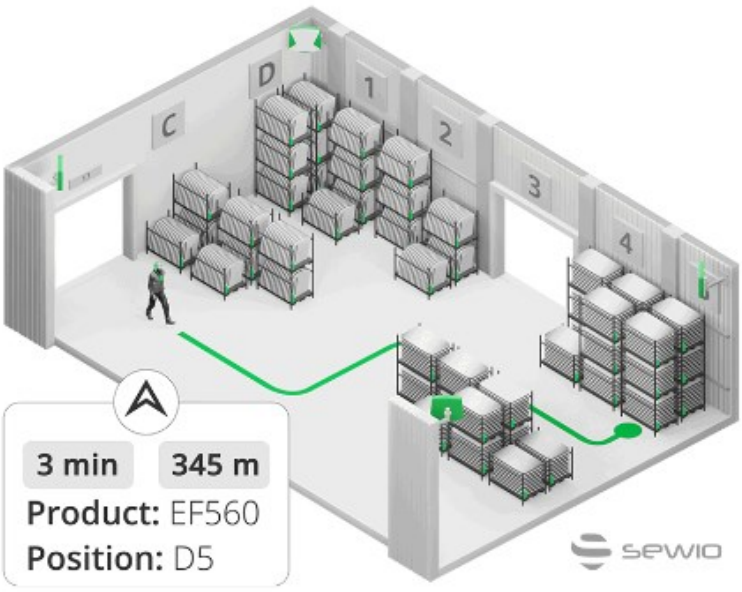
Společnost Sewio nabízí systém pracující s technologií UWB, díky které nabízí 30 centimetrovou přesnost zaměření polohy uvnitř budovy. Pro zjišťování přesné polohy používá software postup buď *Time Difference of Arrival* (překlad: časový rozdíl mezi příchody) nebo *Two Way Ranging* (překlad: dvou cestné vytyčení).

Společnost vytváří návrhy systému na bázi Ad-Hoc, nemá univerzální platformu, ale řeší aplikaci jako projekt. V její nabídce pro navigaci v průmyslovém prostředí jsou tyto varianty:

- Navigace návštěvníků a nováčků
- Navigace k potřebné položce
- Vnitřní navigace vysokozdvižného vozíku
- Vylepšení vyskladňovacího procesu

Mimo průmyslové aplikace nabízí společnost ještě aplikace do zdravotnictví, pro letiště, kancelářské budovy aj.[30]

Koncept navigace po skladu je znázorněn na Obr. 15. Jedná se o halu pokrytou signálem z UWB antén, které jsou umístěné v rohu místnosti. Pracovník je naváděn mobilním zařízením (vizuálně čte navigaci) nebo je naváděn pomocí hlasové navigace.



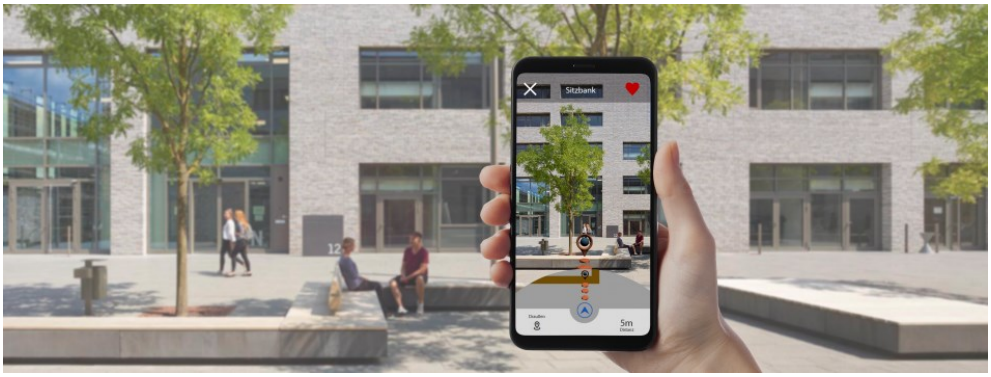
Obr. 15: Koncepční návrh navigace ve skladu – Sewio [30]

2.3.3. XRGO

Společnost XRGO se specializuje na softwarový vývoj spojený s rozšířenou realitou. Zabývá se např. návrhem vzdálené podpory, tvorbou návodů v AR nebo AR navigací.

Demonstrovaná navigace navádí návštěvníky nebo studenty po kampusu univerzity, a to jak venku, tak i uvnitř. Řešení, které firma vytvořila je odlišné tím, že nepoužívá žádné lokační senzory pracující na radiových vlnách (GPS, RFID atd.), místo toho je využíváno rozpoznávání okolních objektů pomocí kamery. Podle vyjádření společnosti nejsou třeba ani žádné dodatečné markery. Jediným požadavkem je připojení zařízení k internetu. [31]

K rozpoznání objektů v okolí je použit balíček Microsoft *Azure Spatial Anchors*, který zpracovává obraz okolí z kamery. Obraz se převede na mračno bodů, ze kterých jsou vypočteny tzv. kotevní body. Pokud jsou kotevní body spojeny s 3D modelem budovy, lze z této spojitosti vypočíst aktuální polohu telefonu. V Azure byl také vytvořen mřížkový model s potencionálními cíli, ze kterého je vypočtena nejkratší trasa. Trasa se uživateli zobrazuje přímo na displej telefonu do snímané reality pomocí prvků rozšířené reality, podoba je na Obr. 16.



Obr. 16: Navigace XRGO po kampusu univerzity [32]

3. Návrh aplikace

Dalším krokem po průzkumu technologické stránky potřebné k tvorbě aplikace a zkoumání „*state of the art*“ současných řešení, je navržení předběžné podoby aplikace, která bude následně vytvořena. Při návrhu je posuzována softwarová a hardwarová stránka aplikace a je zohledněna problematika prostředí. Výstupem je popis vlastností včetně vizualizace návrhu podoby aplikace.

Cíl tvorby aplikace

Cílem praktické části práce je vytvoření aplikace, která naviguje uživatele za pomoci zobrazení prvků rozšířené reality. Úkolem je také vytvoření takové aplikace, která by umožňovala navigování v prostoru skladu, jehož prostor skýtá určité výzvy např. frekventovaný pohyb osob a zařízení, regálové uličky (rušení signálu, obsah polic se mění atd.), velká plocha aj.

Aby aplikace co nejblíže zachycovala obsah činnosti, kterou vykonávají operátoři skladu, bude aplikace obsahovat možnost procházení několika skladových pozic za sebou, jako v případě vychystávání výdejky. Cestu bude uživateli vyznačovat navigační AR linie, ta se bude generovat na pochozích plochách. Pro explicitní vyjádření pozice, na kterou má uživatel dojít, bude tato pozice označena vizuálním AR prvkem, tak aby byla zřejmá cílová pozice.

Shrnutím lze cíle tvorby aplikace vypsat v bodech:

- navigování pomocí zobrazení **prvků rozšířené reality**,
- implementace v **prostředí skladu**,
- podpora běžně dostupných zařízení,
- **simulování vyskladnění** (procházení skladových pozic),
- **vizuální označení cílové pozice**.

Na tvorbu aplikace navazuje **ověření funkce** konceptuálního řešení a aplikace samotné.

3.1. Představení skladu Outdoor Concept a.s.

Aplikace bude vyvíjena pro prostory skladu společnosti Outdoor Concept a.s., zaměřující se na distribuci obuvi a oblečení. Distribuční centrum společnosti se nachází na okraji Plzně, v městské části Plzeň-Radčice. Objekt skladu je sdílen se sesterskou společností Rock Point a. s., skladovací prostor pro tuto společnost je oddělen na hale D.

Sklad společnosti Outdoor Concept a.s. se dělí na tři skladovací haly:

- **Hala A** – kombinace systému VNA v horních pozicích (výška 10 m), ve spodních pozicích (kusový) sklad obuvi s ruční obsluhou.
- **Hala B** – skladování oblečení a obuvi v dosahových úrovních a systém VNA v horních policích, u této haly nižší regály (6 m).
- **Hala C** – skladování paletových zásob, obsluha VZV.

Pro účely testování vytvořené aplikace byli vybrány prostory haly A, konkrétně její dvě dílčí části ze tří celkových – **ABX** a **ACX**. Výběr byl zmenšen především z důvodu časové náročnosti vytváření modelu a příprav pro testování viz kapitola 5.1.

Specifikace prostředí

Prostředí skladu může být velmi rozdílné, záleží na typu skladu (venkovní/vnitřní), druhu skladovaného zboží, množství pater, druhu osvětlení aj. Z toho plyne, že označení prostor

pouze typem sklad zahrnuje široké spektrum prostor a tvorba univerzálního řešení pro takový rozsah proměnnosti by bylo velmi náročné a velmi pravděpodobně nevhodné. V této práci bude tedy vyvinuta aplikace pro konkrétní skladovací prostor, která by však po modifikaci mohla být použita na typově podobné skladovací prostory.

Specifickým rysem této haly jsou vysoké regály a rozsáhlý prostor před regály, hala je prosvětlená světlíky doplněnými stropním osvětlením. Na hale se pohybují jak operátoři s ručně vedenými vychystávacími vozíky, tak VNA vozíky obsluhující horní pozice.

Prostor skladu **ACx** a **ABx** lze tedy typizovat jako:

- skladování oblečení a obuvi,
- ve spodní části regálů ruční vyskladnění,
- v horní části regálů vyskladnění vozíky VNA,
- vysoká hala (10 m regály),
- délka regálů 25 m,
- šířka regálových polí 2,8 m nebo 3,6 m,
- prosvětlení světlíky (nepravidelně) a závěsnými svítidly,
- okna v západní stěně.

Výše uvedené parametry jsou defacto limitacemi pro testování aplikace ve skladu. Výsledným zhodnocením výstupů bude možné více zobecnit zadání a odstranit limitace.

3.2. Návrh řešení softwarové a hardwarové stránky aplikace

Na základě získaných znalostí z průzkumu možností užití rozšířené reality pro navigaci uživatele prostředím a z představených aplikací pro navigaci pomocí AR, byl vytvořen návrh kombinace hardwarových a softwarových prvků aplikace. Při návrhu bylo nutné zohlednit vlastnosti prostředí skladu, ve kterém má být aplikace umístěna.

3.2.1. Hardwarové vybavení

Velmi důležitým prvkem, který ovlivňuje celou podobu řešení, jsou druhy použitého hardware. Jedním z nejdůležitějších aspektů je, jaké zařízení bude použito pro implementaci aplikace, na základě této volby je ovlivněna možnost použití dalších prvků. Další prvky jsou zvoleny např. na základně navrhované metody určování polohy a dalších požadavků. Více v dalších odstavcích: zařízení a ostatní hardwarové prvky.

Zařízení

Hardwarovým prvkem, na kterém bude výsledná aplikace spuštěna, byl vybrán **mobilní telefon**. Aplikace bude konkrétně vyvíjena pro zařízení s operačním systémem **Android.** Jedním z důvodů výběru bylo, že zařízení s operačním systémem Android jsou velmi běžná a oproti chytrým brýlím jsou cenově dostupná. Dalším faktorem bylo, že vývoj aplikací pro mobilní zařízení s androidem je rozšířený v komunitě vývojářů a snadněji se hledá podpora při řešení problémů.

Ostatní hardwarové prvky

Kromě samotného zařízení je třeba stanovit, zda budou potřeba další zařízení k provozu aplikace jako např. *Bluetooth Beacon*. Tato práce bude mířit na možnost vytvoření aplikace, která vyžaduje minimální nároky na instalaci hardwarových prvků do prostoru skladu a která by umožňovala navigovat uživatele pouze pomocí mobilního zařízení.

Přínosem takového řešení je, že pro vytvoření a spuštění aplikace nebude nutné instalovat další prvky do skladu, čímž by se mohly snížit náklady na implementaci aplikace i časový horizont spuštění testování.

Skladové pozice v testovaném skladu budou dočasně označeny markery ve formě QR kódů, které budou rozmístěny na jednotlivých skladových pozicích. Markery jsou důležité pro správné získávání informací o pozici, na které se uživatel s aplikací nachází. Výhodou QR kódů, sloužících jako markery, je jejich nízká cena a snadná tvorba např. volně dostupným generátorem na internetu. Další výhodou je možnost číst QR kódy pomocí kamery mobilního zařízení. Nevýhodou markeru je nutnost dodržet jejich přesné umístění v prostoru, které musí být v průběhu používání stálé.

3.2.2. Vývojové softwarové prostředky

Pro tvorbu aplikací na mobilní zařízení s operačním systémem Android je více možností, tato práce se věnuje tvorbě aplikace ve vývojovém programu Unity. Rozhodnutí vzniklo především kvůli oblíbenosti, online manuálu včetně kurzů a rozšířené uživatelské podpoře.

Jak již bylo představeno dříve v kapitole 2.2, v případě Unity je možné pro tvorbu AR aplikace použít doplňku AR Foundation, který spolupracuje s balíčkem ARCore přímo v mobilním zařízení. Spolu s Unity byl při vývoji používán i software Visual Studio Enterprise, ten byl přímo propojen do vývojového prostředí Unity. Visual Studio je software pro tvorbu širokého spektra aplikací, podporu je programovací jazyky jako např*.: C++, C Sharp, Visual Basic .NET*, aj.

Kódy jsou pro tuto AR aplikaci psány v programovacím jazyce C # (C Sharp).

3.3. Konkrétní návrh aplikace

V rámci této práce je jedním z cílů prozkoumat možnosti navigace pomocí rozšířené reality (AR) v prostředí skladu. Po zkoumání současných technických a systémových řešeních je dalším krokem sestavení experimentu. Jako experimentální část práce byla vytvořena aplikace, která by uživatele navigovala průmyslovým prostředím skladu.

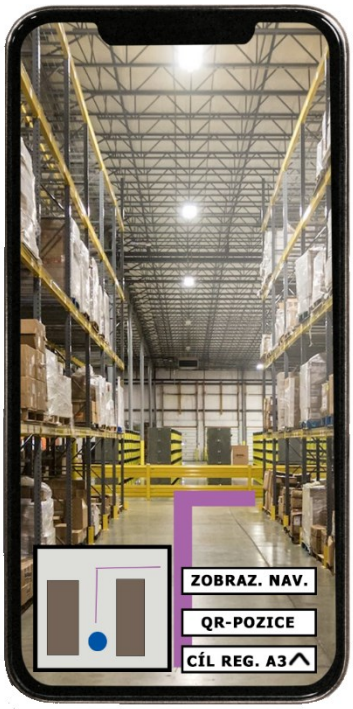
Popis navrhované podoby vyvíjené aplikace

Aplikace je vyvíjena pod pracovním označením **IAWN** [*Aijvn*], což reprezentuje zkratku pro: *Indoor Augmented reality Warehouse Navigation,* ve volném překladu: navigace rozšířenou realitou pro vnitřní sklad.

Jak již bylo zmíněno v kapitole 2.2 aplikace byla určena pro mobilní telefon s OS Android. Aplikace by měla být jednoduše ovladatelná a umožňovat navigaci pro uživatele neznámým prostředím. Konceptuální návrh je následující:

Navigování bude zprostředkováno AR prvkem, který bude uživatele navádět prostředím. Směr, kterým se má uživatel pohybovat je znázorněn naváděcí čarou (fialová linie na Obr. 17), která je umístěna do promítaného obrazu okolního prostředí (přenos obrazu z kamery) na displej telefonu. Návrh podoby aplikace je na Obr. 17.

Základem navigace je přenos obrazu z kamery, který slouží jako background celé aplikace – do tohoto prostoru se vkládají virtuální prvky. Ve spodní části je prostor pro funkční tlačítka a výřez mapy („**minimapa**“), ta umožní rychlý přehled o pozici uživatele. Ve výřezu mapy prostředí se bude v reálném čase zobrazovat aktuální poloha uživatele na 2D mapě prostředí skladu, která je umístěna v levém dolním rohu obrazovky (Obr. 17).



Obr. 17: Návrh podoby AR navigace na mobilním zařízení

Navigace umožňuje zadávání cíle ve dvou režimech:

- **režimu ručního (přímého) zadávání**
- **simulace vyskladnění výdejky**

V případě přímého zadávání si uživatel může vybrat svůj cíl ze seznamu, v průběhu navigace je možné přepínat mezi jednotlivými cíli navigace. Pokud je používán režim vyskladnění výdejky, uživatel postupuje postupně podle vygenerovaného listu pozic, pořadí nelze měnit.

Aplikace potřebuje pro navigaci znát **referenční polohu**, proto je po spuštění uživatel vyzván k volbě skenu skladové pozice nebo vycentrování telefonu na referenční pozici (před kanceláří ve skladu). Jednotlivé skladové pozice budou označeny markery, které slouží k identifikaci jednotlivé pozice. Tato identifikace propojuje v aplikaci marker s polohou a rotací uloženou v aplikaci.

Pozice uživatele v prostoru bude určována pomocí sledování prostoru kamerou zařízení. V případě odchylky zobrazované pozice od skutečné, je možnost korekce polohy v prostoru pomocí načtení QR kódu, ten slouží jako marker pozice. Sledování polohy okolí je řízeno pomocí ARCore, které rozkládá obraz na mračno bodů, díky čemuž dokáže rozpoznat pohyb zařízení (změny obrazu) viz kapitola 2.2. V případě výpadků obrazu nebo nedostatečné kvality obrazu je zapojeno pomocí *machine learning* dopočtení pohybu na základě dat z akcelerometru a gyroskopu zařízení.

4. Tvorba aplikace

Po analýze zadání a vytvoření vědomostního základu a určení cílové podoby aplikace byla zahájena samotná tvorba aplikace ve vývojovém nástroji Unity. Následující kapitoly popisují jednotlivé prvky, ze kterých je aplikace složena.

4.1. Model prostředí

Jak již bylo nastíněno v kapitole 3.1, aplikace je vytvořena na míru pro prostředí skladu Outdoor Concept a. s. V případě tvorby aplikace, jež sleduje pohyb prostředím na základě dat z ARCore, je pro vkládání objektů a pozic nutné nejdříve sestavit model fyzického prostředí v pracovním prostředí Unity.

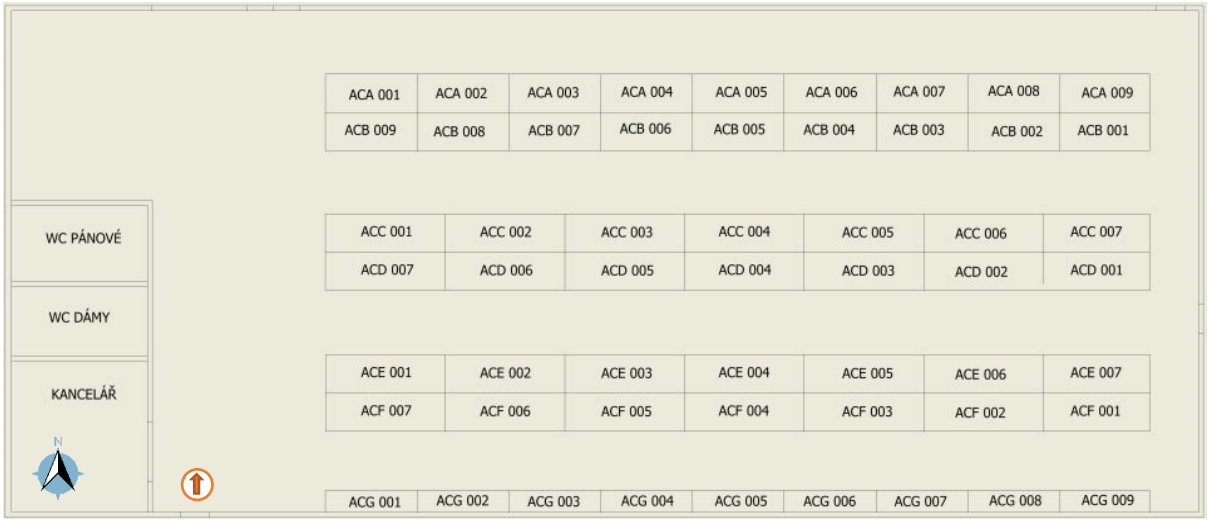
V případě tohoto přístupu je vytvořen „holý“ model prostředí, jedná se tedy o podlahu a svislé plochy – zdi a regály skladu. Tento model lze přirovnat kartonovému bludišti pro myši.

4.1.1. 3D model prostředí v Unity

Prvním krokem bylo změření kalibrovaným laserovým dálkoměrem *Stabila LD-320* prostor skladu, z toho byl vytvořen 2D plánek obou částí skladu A (ACx a ABx). Vytvoření 2D výkresu bylo provedeno v softwaru Autodesk Inventor Pro. Výsledný výkres skladu je na Obr. 18.

Rozměry půdorysu jednotlivých částí haly A jsou:

- **ABx** - 36 x 24 m,
- **ACx** - 36 x 15 m.



Obr. 18: Výkres skaldu ACX (podklad pro model prostředí v Unity)

V dalším kroku byly vytvořeny krychle (Unity element *cube*) v rozměrech půdorysu jednotlivých hal (měrné jednotky Unity jsou metry) a těmto kvádrům byly přiřazeny materiály s texturou[5] výkresu jednotlivých hal. Tím byl vytvořen základ tedy podlahová plocha.

Na základě výkresu byly rozmístěny „úzké“ kvádry na pozice zdí (výška není podstatná, byla nastavena na 4 m). Těmto objektům byl přiřazen materiál[6] *VR/SpatialMapping/Occlusion*, který přiřazuje vlastnost, že při renderování obrazu prostředí nebude objekt zobrazen.

[5] Unity textura – typ materiálu přiřazený objektu, určuje objektu vzhled, může být barva nebo obrázek aj.
[6] Unity material – materiál přiřazuje objektům vlastnosti – vzhled, fyzikální chování aj.

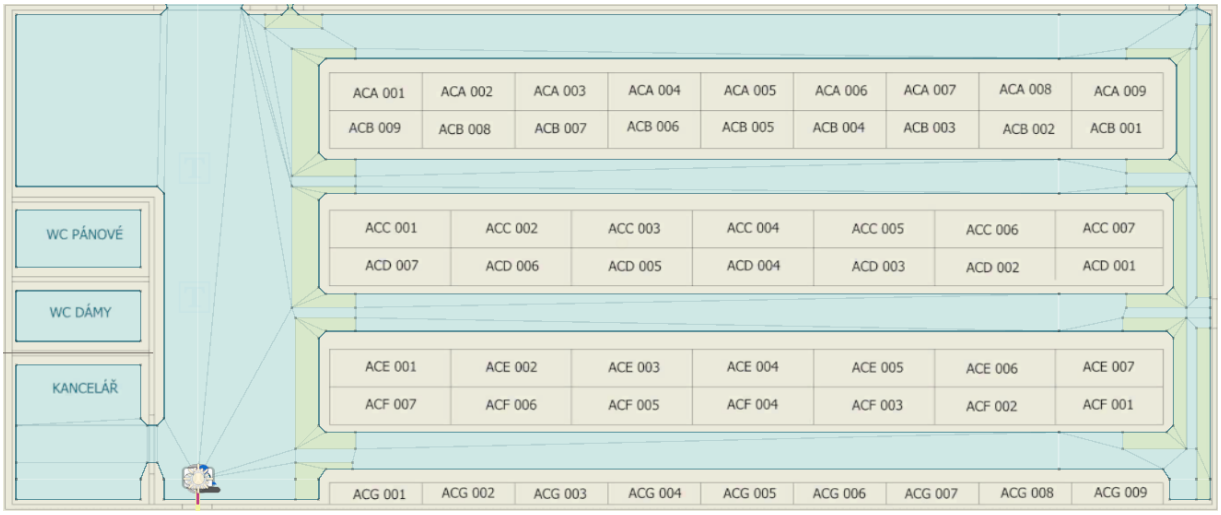
Této vlastnosti je využito především v renderování mini mapy. Minimapa je render (zobrazení) kamery, která je umístěna v Unity modelu prostředí přímo nad uživatele a snímá jeho blízké okolí z ptačí perspektivy.

Dalším základním prvkem prostředí jsou regály, ty byly obdobně jako zdi rozmístěny podle výkresu na pozice regálů. Opět i těmto objektům byla přiřazena vlastnost *Occlusion*. Výška těchto objektů byla omezena (1,5 m), tak aby nedocházelo k vytváření ploch *NavMesh* uvnitř těchto kvádrů viz. další kapitola 4.1.2.

4.1.2. Vytvoření NavMesh

NavMesh je v této aplikaci používán pro generování naváděcí čáry. Představuje oblast, po které je možné navádět uživatele.

Jako základní plocha pro generování *NavMesh* byla použita podlaha – konkrétně objekty *ACXHalaFloor* a *ABXHalaFloor*. Na jejich horní ploše byla programem vygenerována modře označená plocha *NavMesh* viz. Obr. 19, která je uzpůsobena pro pohyb *Bake Agent* (v překladu: tvůrčí postava – představuje uživatele). V tomto případě jej reprezentuje válec s poloměrem 0,3 m a výškou 2 m (tyto dispozice ovlivňují např. průchod dveřmi nebo generování ploch ve snížených prostorech atd.). Dalšími parametry, které ovlivňují generovanou plochu jsou úhel stoupání a výška kroku. Tyto hodnoty ale v tomto případě nejsou relevantní (plocha skladu je v jedné rovině). Výsledek generovaného *NavMesh* pro halu ACx je znázorněn na Obr. 19, zobrazení celkové plochy *NavMesh* i pro halu ABx je v *Příloha 1*.



Obr. 19: Generovaný NavMesh skaldu ACx

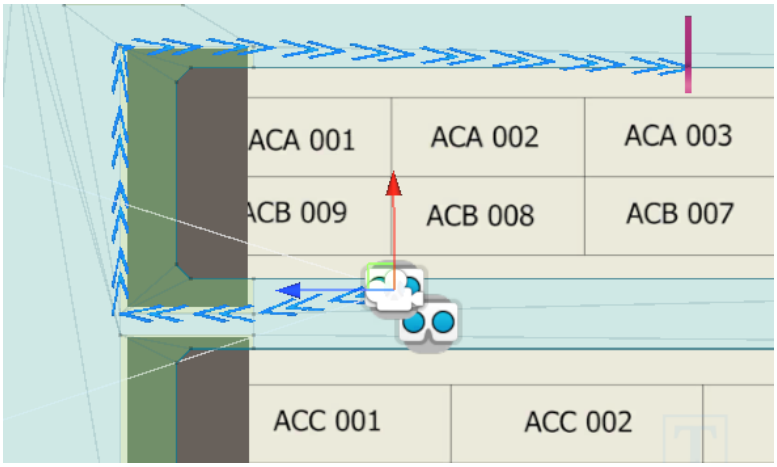
NavMesh obsahuje také možnost zařazení objektů do tříd a přiřazení třídě společné vlastnosti. Například *Not walkable* je předdefinovaná třída, která není průchozí, z toho důvodu se na ní negeneruje *NavMesh* – do této třídy jsou zařazeny regály.

Korekce generované trasy

Jak již bylo zmíněno, naváděcí linie je zobrazována po celé ploše generovaného *NavMesh*. Při testování se však jevilo jako zavádějící to, že při navigaci „za roh“ se navigační linie zobrazovala příliš blízko hrany regálu, což budilo dojem, že je nutné projít regálem. K regulaci tohoto jevu byli přidány *regulátory* na rohy regálů.

Princip regulátoru vychází z atributu funkce rychlosti pohybu po ploše *NavMesh*, která umožňuje nastavovat rychlost pohybu po dané skupině objektů. Výchozí rychlost pro pohyb je 1, pokud je nastavená hodnota větší než 1, bude plánovaný pohyb pomalejší. Pokud tedy vložíme objekt s atributem pochozí rychlosti větší než 1 do plánované trasy, algoritmus se mu snaží vyhnout, za takové podmínky, že průchod pomalejší částí (plochou) není rychlejší než obcházení. Tento jev by se dal přirovnat efektu bažiny, proto se objekty pracovně jmenují *swamp* (anglicky bažina) např. „*ACGswamp*“.

Této vlastnosti bylo využito při tvorbě třídy *NavTrajectoryAdjustment*, tato *NavMesh* třída má nastavenou rychlost pohybu na hodnotu 4. Do této třídy byly zahrnuty velmi ploché kvádry s vlastností renderování *Occlusion* (viz kapitola 4.1.1), které jsou umístěny na rohu každého regálu v minimální výšce nad podlahou. Tyto plochy tedy fungují jako virtuální kužely pro korekci vypočítávané trasy. Kromě rohů regálů byly umístěny kvádry i podél stěny za regály, kde ze stěny vystupují cca 30 centimetrů do prostoru nosníky, mezi kterými jsou často odložené krabice. Tím pádem je bezpečnější zamezit generování trasy příliš blízko u zadní stěny. Výsledek korekce trasy je vidět na screenshotu simulace v Unity na Obr. 20, navigační linie kopíruje obrys korektoru (zelená plocha *NavMesh*) a navádí uživatele (ikonka kamery) „obloukem“ za roh regálu.



Obr. 20: Korekce generování trasy v NavMesh

4.1.3. Cíle (Targety)

Pro navigaci jsou klíčové dvě pozice: současná poloha a cílová poloha. V tomto případě jsou cílovými polohami regálové pozice ve skladu.

Regály jsou všechny shodně dlouhé 25 metrů a jejich regálové pole jsou dlouhá buď 2,8 metru nebo 3,6 metru. Mají tedy 9 nebo 7 řad, v závislosti na délce regálového pole. Pro zmenšení množství dat, které je nutné zadat, je jako jedna pozice považován sloupec v regálu nikoliv jednotlivé police, jak je tomu ve skladovém systému. Jelikož i přesto zůstává množství pozic relativně velké, byly náhodně vybrány pouze některé. Bylo vybráno celkem 33 skladových pozic v částech skladu ABx a ACx.

K tomuto zmenšení velikosti vzorku bylo přistoupeno především z důvodu, že markery není možné na regálech nechat mimo testování. A to z důvodu, že jako markery jsou použity QR kódy, které je možné číst i mobilním terminálem *Zebra TC26*, který je ve skladu používán při vychystávání. Markery jsou umístěny v blízkosti v současnosti používaných čárových kódů, kterými jsou označeny pozice. Z této kombinace plyne značné riziko, že by docházelo

k chybným skenům pozic. Proto byly markery na regály rozmístěny pro každé testování a následně odstraněny.

Atributy regálové pozice

Regálové pozici je v Unity přiřazena souřadnice v prostoru a rotace. Rotace je pozici přiřazena kvůli funkci kalibrace, pro samotnou navigaci není třeba.

Každá regálová pozice je v běžící aplikaci reprezentována jako prázdný objekt (*Empty Object*), který má již zmiňovanou polohu a rotaci. Jednotlivé objekty jsou uloženy v listu objektů. Správu cílových pozic zajišťuje třída *TargetHandler* viz. kapitola 4.3.1, list se generuje při spuštění aplikace.

Pozice se načítají ze souboru *NavTargets.json*, který slouží jako databáze všech pozic. Pokud je požadavek na změnu nebo přidání pozice, pracuje se s tímto souborem. Data jsou uložena ve formátu *JSON*, jeho struktura je znázorněna v Zdrojový kód 1. Regálová pozice je zanesena ve formátu pozice a rotace, regálové pozice jsou rozklíčované podle názvu (`Name`). Z ukázky je patrné, že rotace (`Rotation`) i pozice jsou ve stejném formátu `x, y, z,` v tomto případ však hodnoty znamenají eulerovské úhly. Takto je uložena každá cílová pozice, se kterou aplikace pracuje. Unity umožňuje soubory *JSON* číst i přidávat položky, v případě této aplikace jsou pozice za běhu aplikace stálé. Tvorba pozic byla provedena při vývoji v Unity, přesněji v propojení s aplikací Visual Studio Code.

```
01  {
02      "TargetList": [
03          {
04              "Name": "ACA002",
05              "Position": {
06                  "x":  8.53,
07                  "y": 0.7,
08                  "z": 12.49
09              },
10              "Rotation" : {
11                  "x": 0.0,
12                  "y": 180.0,
13                  "z": 0.0
14              }
15          },
16      ]
17  }
```

Zdrojový kód 1: List regálových pozic Target list, struktura JSON

Marker pozice

Aplikace je vybavena čtením QR kódů, ty skýtají výhodu v jednoduchém vytvoření, snadném čtení a volitelné nositelné informaci. Markerem je tedy QR kód, ten je doplněn uprostřed o ikonu, aby se marker stal rozpoznatelnějším pro *AR Image Tracking* funkci testovanou v aplikaci. Marker je rovněž opatřen konturním rámečkem, který jej vymezuje a také podporuje rozpoznávání. Do markeru lze vložit libovolnou ikonu, cílem bylo vybrat jednoduchou, ale zároveň dostatečně různorodou ikonu pro snadné rozpoznání funkcí *AR Image Tracking*.

Z volně dostupných ikon v generátoru *QRPlanet.com* [33] se jevila jako vhodná varianta ikona na Obr. 21.

Pro každou pozici bylo nutné vytvořit vlastní marker, tak aby mohla být pozice při načtení markeru jednoznačně identifikována. Propojení s regálovou pozicí je pomocí klíče – názvu (`Name`), příklad je na Obr. 21.

Jak již bylo zmíněno, markery jsou určeny jak pro čtení pozice, tak pro využití funkce *AR Image Tracking* pro spínání čtení obsahu QR kódu. Knihovna markerů pro *AR Image Tracking* se spravuje v prostředí Unity, kde se přidávají jednotlivé obrázky (markery). Sledování obrázků je řízeno v *AR Foundation* pomocí komponentu *AR Tracked Image Manger*, ten spravuje sledování obrázků podle připojené knihovny. Akce po nalezení markeru je řízena třídou *ImageTracking*, jejíž popis je v kapitole 4.3.

Obr. 21: Marker pro regálovou pozici ACA002

4.2. Vývojové diagramy

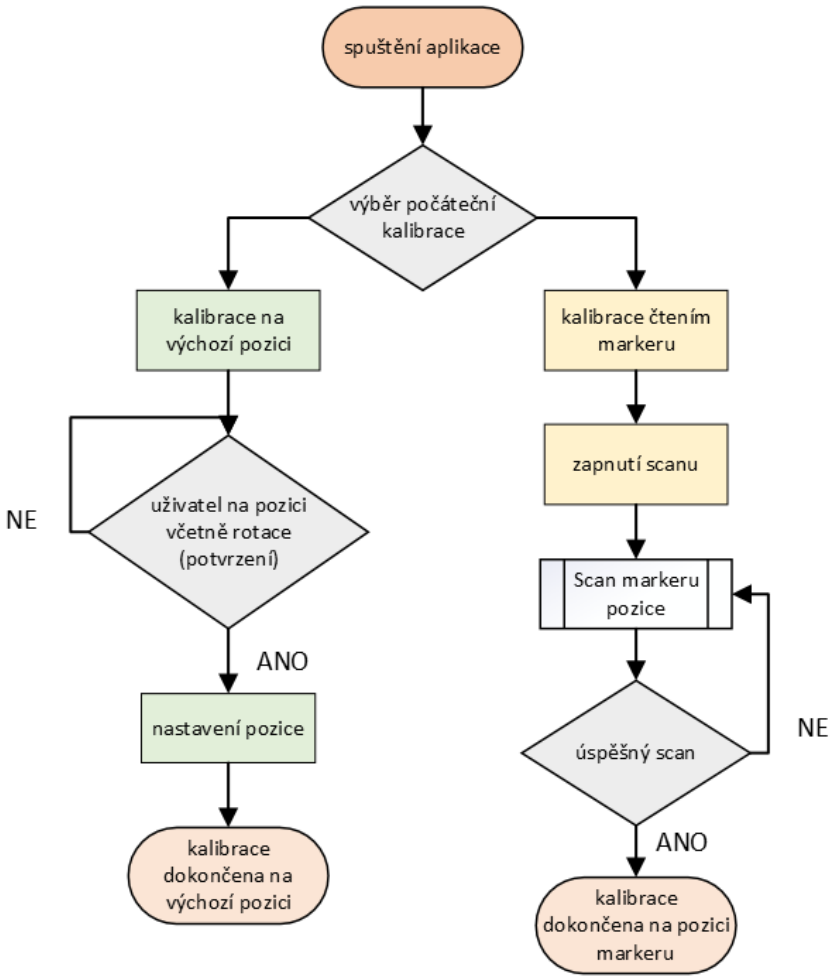
Z kapitoly 2.2 je známo, že aplikace umožňuje funkci ve dvou módech: **manuální zadávání** a **simulaci vychystávání**. V této kapitole je detailněji rozebrána funkcionalita v obou módech. Oběma módům předchází **počáteční kalibrace**.

Počáteční kalibrace

Tento proces nastává automaticky spuštěním aplikace, kdy je uživatel vyzván, aby zvolil, jakým způsobem se provede prvotní kalibrace polohy zařízení. Počáteční proces kalibrace je *klíčový*, protože jak již bylo zmíněno v kapitole 2.2, pro správné určení polohy zařízení je nutné znát souřadnice počáteční polohy (nebo poslední polohu průběžné kalibrace). Na výběr jsou dvě možnosti viz. Obr. 22:

- **kalibrace skenem markeru** – uživatel načte jakoukoliv regálovou pozici,
- **kalibrace na výchozí pozici** – uživatel se postaví do výchozí pozice před kancelář a nasměřuje telefon ve znázorněném směru (oranžová šipka na Obr. 18), poté tlačítkem potvrdí kalibraci.

Závisí na uživateli, jaký způsob si zvolí, obecně lze předpokládat, že možnost kalibrace skenem markeru je bezpečnější, polohu totiž omezuje rozsah čtení QR kódu. V případě kalibrace výchozí polohou je výsledek závislý na zkušenosti a dodržení polohy a rotace uživatelem, může tedy docházet k chybám.



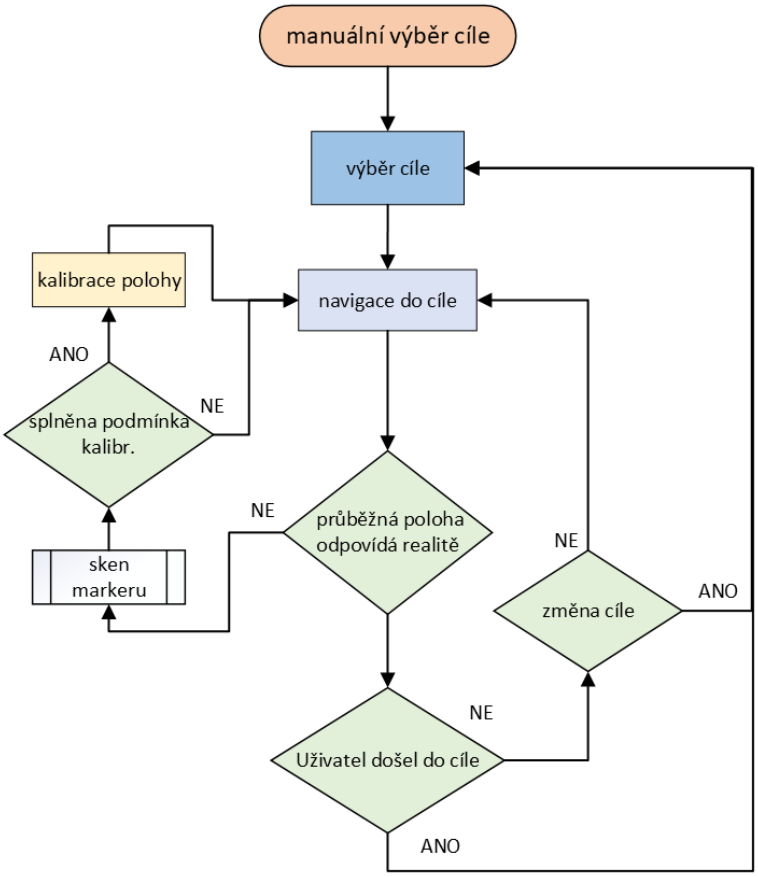
Obr. 22: Kalibrační menu po spuštění

Po provedení kalibrace je spuštěn automaticky mód manuálního výběru cíle z listu pozic. Výchozí cílová pozice je prázdná, pro zobrazení navigace je nutné vybrat cíl a stisknout tlačítko viditelnosti navigace.

Manuální zdávání

Proces se v módu manuálního výběru chová jako nekonečná smyčka, tj. stále se opakuje shodná posloupnost operací, ta je zobrazena na Obr. 23. Tento mód aplikace uživateli pouze zobrazuje navigaci na požadovanou pozici, není po uživateli požadováno, aby na zvolenou pozici došel a potvrdil, že se o danou pozici jedná. Uživatel může kdykoliv změnit cíl navigace, čímž se vrátí opět na začátek pomyslného cyklu. Cyklus lze větvit průběžnou kalibrací polohy, v případě, že si uživatel uvědomí na základě své orientace a zobrazovaných hodnot v aplikaci odchylky od reality (chybná pozice na mapě, chybně se zobrazující navigační linie, aj.).

Průběžná kalibrace je podproces, který je spouštěn skenem markeru. Pokud je splněna podmínka, je následně provedena kalibrace polohy. V opačném případě je uživatel pouze informován o skenu markeru. Funkční logiku ovládá třída *QRCodeRecenter* viz. kapitola 4.3.1. Podmínkou pro provedení kalibrace je naskenování stejného markeru dvakrát po sobě, v minimálně dvou sekundovém odstupu. Skenování dvakrát po sobě je nastavené, protože není žádoucí provádět kalibraci při každém skenu markeru, je požadováno, aby kalibrace byla spuštěna záměrně uživatelem.

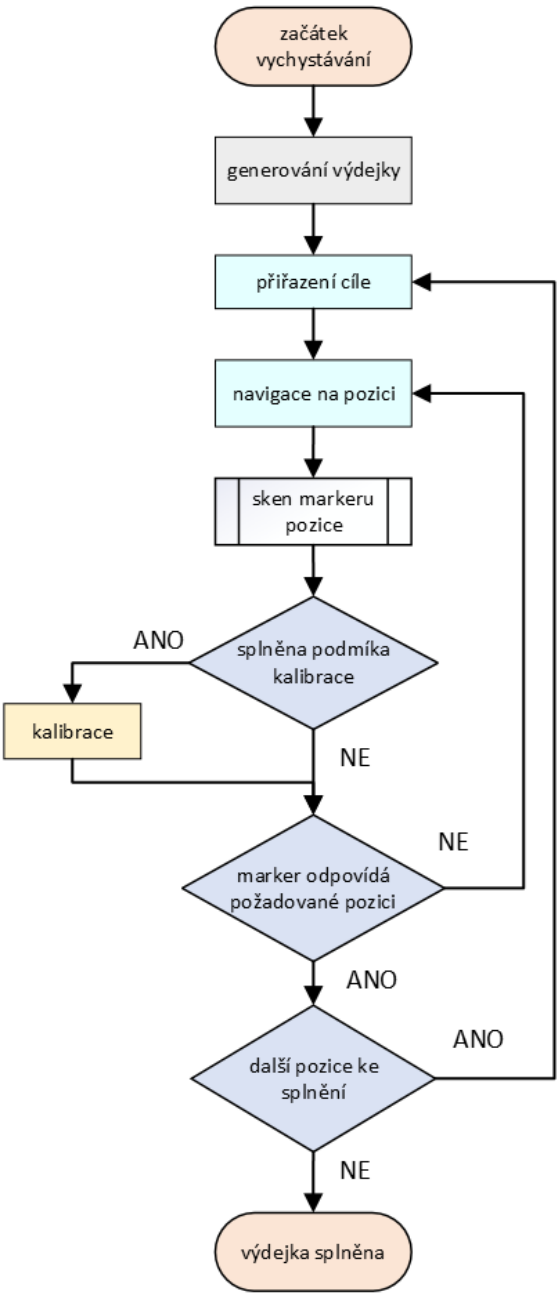


Obr. 23: Ruční (manuální) výběr cíle vývojový diagram

Simulace vyskladnění výdejky

Druhým, již zmíněným módem, je simulace vyskladnění výdejky (znázorněno na Obr. 24), v tomto režimu uživatel pracuje s vygenerovanou výdejkou. Výdejkou se v případě této aplikace rozumí náhodná kombinace (včetně opakování) pozic dostupných z listu pozic. V případě budoucího nasazení aplikace v pracovním provozu by výdejka obsahovala pozice vygenerované na základě objednávky.

Proces začíná generováním listu cílových pozic, list nabývá velikosti od 1 cílové pozice do 10 včetně. Pořadí cílů je pevné a nelze měnit jejich pořadí. Cyklus navádění na průběžnou cílovou pozici se opakuje, dokud není příkaz splněn. V průběhu cyklu je možné kalibrovat pozici, tato kalibrace nijak neovlivní navigaci na průběžný cíl.



Obr. 24: Vývojový diagram procesu simulace výdejky

4.3. C# zdrojové kódy

V aplikaci je vytvořeno několik kódů C# (*scripts*), jejichž funkčnost je oddělená, tak aby každý vykonával určitou funkci a mohl být samostatně přiřazen adekvátnímu Unity objektu. V souhrnu je práce tvořena zhruba 900 řádky kódu.

Scripty jsou rozděleny do tří složek, podle určení. V *Core Functions* jsou kódy určené pro správu základních funkcí navigace např. tvorba navigační linie, v *Target Models* jsou třídy, které jsou využívány pro tvorbu generování skladových pozic a ve složce *Utilities* jsou zařazeny podpůrné kódy např. zobrazení cílové šipky.

Rozdělení scriptů aplikace do složek je následující (s popisem funkce):

- **Core Functions**
  Navigation Controller – správa navigace,
  QR Code Recenter – skenování markeru,
  Target Handler – správa pozic,
  Work Cycle – simulace výdejky.

- **Target Models**
  Target – třída pozic,
  Target Wrapper – list třídy *Target,*
  Target Object – obalující třída pro *Target.*

- **Utilities**
  Button Text Changer – třída pro přejmenování tlačítka (*button*),
  Image Tracking – sledování markeru,
  Intro Menu – správa menu po spuštění,
  Show End Point – zobrazování cílové šipky,
  Switch – přepínání mezi režimy.

4.3.1. Core Functions

Core Functions neboli „jádro aplikace“ jsou základní třídy, které určují chování aplikace. V této kapitole je jejich popis a funkce v aplikaci.

Navigation Controller

Hlavním úkolem tohoto kódu je vykreslování a výpočet naváděcí linie. Jelikož se uživatel se zařízením pohybuje, je nutné průběžně generovat navigační čáru. Z tohoto důvodu je příkaz pro generovaní navigační trasy umístěn v metodě `Update`, která se spouští pokaždé, když je obnovena obrazovka – tedy např. 30krát za sekundu (fps displeje).

Jelikož generovat trasu není nutné v takové frekvenci, a navíc to zbytečně zatěžuje zařízení, je kód uvnitř `Update` zapouzdřen do metody `if`, která porovnává časové razítko. Interval, po kterém je podmínka splněna je nastaven na 0.5 f (čas má jednotku *float*, hodnota je rovna 0.5 s). V metodě `Update` je také ovládána viditelnost cílové šipky v závislosti na vzdálenosti od cílové pozice: soustava podmínek `if` na řádcích *13-32* ve Zdrojový kód 2.

Vykreslení navigační čáry je tvořeno pomocí Unity objektu *Line Renderer*. Vytvoření navigační trasy je příkazem `NavMesh.CalculatePath`, který je součástí *UnityEngine.AI*.

Pro výpočet naváděcí linie jsou zapotřebí čtyři parametry[26]:

- oblast *NavMesh*,
- výchozí bod,
- cílový bod,
- instance *NavMeshPath,* do které se cesta vytvoří.

Výpočet zmiňované trasy se v ukázce kódu (Zdrojový kód 2, řádka *08,09*) provádí již zmiňovaným příkazem **`NavMesh.CalculatePath`**, jehož vstupy jsou:

`transform.position` – pozice *AR Session Origin*,

`TargetPosition` – pozice cílového objektu (`Target`),

`NavMesh.All Areas` – všechny oblasti vygenerovaného NavMesh,

`CalculatedNavPath` – proměnná, do které se uloží vygenerovaná instance *NavMeshPath*.

```
01 private void Update()
02  {
03      //update každý 3 frame
04      if (Time.time >= nextTime)
05      {
06          if (line.gameObject.activeSelf && TargetPosition != Vector3.zero)
07          {
08              NavMesh.CalculatePath(transform.position, TargetPosition,
09                NavMesh.AllAreas, CalculatedNavPath);
10              line.positionCount = CalculatedNavPath.corners.Length;
11              Vector3[] HighedPath = AddYOffSetLine();
12              line.SetPositions(HighedPath);
13              if (Vector3.Distance(transform.position, TargetPosition) < 7.0f)
14              {
15                  if (Vector3.Distance(transform.position, TargetPosition)
16                                  < 2.5f)
17                  {
18                      ShowArrow.StartShowingFrame();
19                      if (Vector3.Distance(transform.position, TargetPosition)
20                              < 0.5f)
21                      {
22                          ShowArrow.EndShowingFrame();
23                      }
24                      else { ShowArrow.StartShowing(TargetPosition); }
25                  }
26                  else
27                  {
28                      ShowArrow.StartShowing(TargetPosition);
29                      ShowArrow.EndShowingFrame();
30                  }
31              }
32              else { ShowArrow.EndShowing(); }
33          }
34          DoubleClickCheck();
35          nextTime += interval;
36      }
37  }
```

Zdrojový kód 2: Příkaz Update v třídě NavigationController

Trasa navigace je generovaná na základě spojnic uzlů, které se vytvářejí v závislosti na počtu nutných zlomů křivky, vygenerované na ploše *NavMesh,* mezi polohou uživatele a cílovou pozicí. V základní formě je výška trasy dána výškou počátečního a koncového bodu, pokud jsou v různé výšce, navigační čára se poté svažuje. V případě této aplikace se výška navigační linie určuje pomocí odsazení od země.

Metodou `AddYOffSetLine` se jednotlivé uzly přepíší do listu instance *Vector 3,* již s požadovaným odsazením hodnoty souřadnice Y od země. Vektory pozic uzlů jsou poté vloženy do komponentu *Line Renderer* (*navLine*), který mezi jednotlivými body vyrenderuje čáru. V Unity je nastaven pro *Line Renderer* (název: *NavigationLine*) materiál v podobě textury (*Texture*) – obrázek modré dvojité šipky. Ten se v délce linie kopíruje za sebe a vytvořená linka šipek signalizuje směr navigované cesty.

Odsazení od země je závislé na snímaném okolí, v Unity okolí zpracovává aktivovaný prvek ***ARFoundation – AR Plane Manager***, který mapuje okolí a prokládá jej plochami viz. kapitola 3.2.2. V *AR Plane Manager* byly povoleny pouze horizontální plochy, protože se vychází z předpokladu, že podlaha je vodorovnou plochou. Dále byly „*prefabu*[7]“, ze kterého jsou generovány virtuální plochy, odebrány renderovací atributy (*Material Renderer, Line Renderer*), takže se vygenerované plochy nezobrazují.

Pro porovnávání *Y* souřadnic jednotlivých ploch je vytvořena metoda `LowestLayerPosition` (Zdrojový kód 3). Metoda prochází jednotlivé plochy v *arPlaneManager.trackables* a „probublává“ jimi nejnižší nalezenou hodnotu (kombinace příkazu `foreach` a `if` – řádky *04*,*08* v Zdrojový kód 3), která je na konci přiřazena výstupu metody. V průběhu procházení jsou jednotlivé plochy deaktivovány, aby nezatěžovaly zařízení.

Kromě tvoření navigační čáry jsou ještě ve třídě `NavigationController` zpracovány další funkce jako např.:

`SetLineVisibility` – viditelnost nebo skrytí navigační čáry,
`DoubleClickCheck` – kontrola dvojitého kliknutí na tlačítko,
`GetReportOfLine` – vypsání počtu uzlů a délky navigační čáry.

```
01 private Vector3 LowestLayerPosition ()
02     {
03         Vector3 lowestPosition = new Vector3 (0f, transform.position.y , 0f);
04         foreach (var plane in arPlaneManager.trackables)
05         {
06             Vector3 planeValue = plane.gameObject.transform.position;
07             plane.gameObject.SetActive(false);
08             if (planeValue.y < lowestPosition.y)
09             {
10                 lowestPosition.y = planeValue.y;
11             }
12         }
13         return lowestPosition;
14     }
```

Zdrojový kód 3: Metoda LowestLayerPosition pro hledání nejnižší plochy ve třídě NavigationController

[7] prefab – vzorový objekt, který Unity naklonuje a vloží jako nový objekt do prostředí

Target Handler

Třída `TargetHandler` je určena především pro správu regálových pozic, s čímž se pojí i správa průběžného cíle navigace (`Target`). Třída nemá metodu `Update`, protože akce jsou vyvolávány uživatelem nebo voláním z jiných metod. Výjimkou je metoda `Start`, která po spuštění aplikace načte regálové pozice ze souboru **NavTargets.json** a vygeneruje pozice na scénu aplikace (*Scene*). Následně jimi naplní list pozic `currentTargetItems`.
K rozklíčování obsahu souboru JSON je využit doplněk ***jillejr.newtonsoft.json-for-unity.converters***, který je součástí oficiálního UPM (Unity Package Manager).
Tento balíček obsahuje převodníky do běžných typů používaných v Unity. Jedná se o typy jako *Vector2, Vector3, Matrix4x4, Quaternions, Color,* dokonce i *ScriptableObject* a mnoho dalších.[34]

Inicializační proces je v okně Zdrojový kód 4, všechny metody v ukázce jsou spádově inicializovány právě zmíněnou metodou `Start`.

Po spuštění první metody `GenerateTargetItems` je vyvolána metoda druhá a to `LoadTargetListFromJSON`. V této metodě je použit zmiňovaný převodník příkazem `JSONConvert.DeserializeObject` na řádcích *23* a *24*, jehož výstupem jsou objekty ve formátu třídy `TargeWrapper`, jež je tvořen listem `TargetList` objektů `Target`.

Výstup metody `LoadTargetListFromJSON` vytvoří iterační list objektů třídy `Traget: IEnumerable<Target> LoadTargetListFromJSON`.

List je následně procházen příkazem `foreach` a pro každý prvek třídy `Target` je vytvořen `GameObject` příkazem `Object.Instantiate` (řádky *12* a *13*), jehož atributy jsou následující:

rodič: `targetObjectPrefab` (*EmptyObject*[8] v Unity),
pozice: `target.Position` – dědí pozici z `traget`,
rotace: `Quaternion.Euler(target.Rotation)` – dědí rotaci z `target`,
transformace: `Transforms [0]` – určuje umístění v Unity podle rodiče (Enviroment – NavigationTargets).

`GameObject` je následně aktivován a je mu přiřazeno jméno rodiče (`Target`), posledním krokem je přiřazení třídě `TargetObject`, k tomu slouží příkaz `AssignToTargetObject`. Tímto procesem byla každá pozice vygenerována na scénu v aplikaci jako objekt.
Další fází inicializačního procesu je naplnění rozevíracího seznamu *NavigationPositionsDropdown*, kde je operace provedena metodou `FillDropdownWithTargetItems`.

[8] Empty Object – prázdný objekt v Unity, který nemá tvar, nezobrazuje se v prostoru, ale má umístění a rotaci, případně k němu lze přiřadit další komponenty

```
01 private void Start()
02     {
03         GenerateTargetItems();
04         FillDropdownWithTargetItems();
05     }
06     private void GenerateTargetItems()
07     {
08         IEnumerable<Target> targets = LoadTargetListFromJSON();
09         foreach (Target target in targets)
10         {
11             //creating Target object in scene
12             GameObject targetNavObject = Instantiate(targetObjectPrefab,
13      target.Position, Quaternion.Euler(target.Rotation), Transforms[0]);
14             targetNavObject.SetActive(true);
15             targetNavObject.name = $"{target.Name}";
16            //change type to TragetObject & add to list
17             currentTargetItems.Add(AssignToTargetObject(targetNavObject));
18         }
19     }
20     private IEnumerable<Target> LoadTargetListFromJSON()
21     {
22         //transforms JSON to list of Targets
23         return JSONConvert.DeserializeObject<TargetWrapper>
24                                 (targetModelData.text).TargetList;
25     }
26     private TargetObject AssignToTargetObject(GameObject target)
27     {
28         TargetObject targetComponent = target.GetComponent<TargetObject>();
29         targetComponent.Name = target.name;
30 
31         return targetComponent;
32     }
```

Zdrojový kód 4: Inicializace regálových pozic ve třídě Target Handler

Další metody jsou podpůrného charakteru a jsou většinou volány z ostatních scriptů, jejich výpis je následující:

`SetSelectedTargetPositionWithDropdown` – změní průběžný cíl navigace na základě výběru z rozevíracího seznamu, hodnotu propíše i do třídy `ShowEndPoint`,
`GetCurrentlySelectedTarget` – metoda vrátí *Vector3* pozici regálové pozice podle zadaného pořadí v listu `currentTargetItems`,
`GetCurrentTargetByTargetText` – metoda vrátí `TargetObject` podle názvu regálové pozice,
`GetCurrentTargetRotation` – metoda vrací *Quateration* (rotaci) průběžného cíle,
`CountOfPositions` – metoda vrací počet (*int*) regálových pozic v listu `currentTargetItems`,
`GetTargetByIndex` – metoda vrací objekt `TargetObject` podle indexu v listu pozic,
`GetNameofTargetByIndex` – metoda třídy vrací název regálové pozice podle indexu v listu pozic,
`RewriteDestinyBut` – přepíše zobrazený text v tlačítku *DestinationTextBut,*
`GetCurrentlyNavigatedTarget` – metoda vrací název (*string*) průběžného cíle navigace.

QR Code Recenter

Posláním této třídy je zajištění procesu skenu markeru a případné kalibrace polohy a rotace zařízení. Čtení QR kódu je založeno na doplňku přidaného do Unity: **ZXing.NET Decoder**. Doplněk je volně dostupná čtečka, která podporuje QR kódy, EAN aj.[35]

Načtení a dekódování QR kódu je převzato od *FireDragonGameStudio*, které vytvořilo třídu zajišťující požadovanou funkci čtení QR kódů. Balíček je dostupný na *GitHub.com*. [36]

Metody, převzaté pro čtení QR kódu jsou následující, včetně stručného popisu:

`OnEnable` – metoda se spustí při načtení třídy, přiřadí třídě `OnCameraFrameReceived` poslední obrázek obdržený z kamery zařízení,

`OnDisable` – metoda se spustí při deaktivaci třídy, odebere obraz, který byl přidělen metodě `OnCameraFrameReceived`,

`OnCameraFrameReceived` – metoda se vstupním parametrem struktury `ARCameraFrameEventArgs`, která je součástí *AR Foundation* a souvisí se zpracováním obrazu. V této metodě je kontrolováno povolení skenování, při splnění je vyžádán poslední obraz z kamery zařízení, ten je následně převeden na formát *RGBA32* a poté ještě na 2D texturu. Takto zpracovaný obraz je zadán ke zpracování metodou `reader.Decode` (funkce *ZXing.NET*), výsledkem je dekódovaná hodnota QR kódu. V případě, že není `null`, je volána metoda **`SetReposition`** s atributem dekódovaného obsahu.

Pro zpracování dekódované hodnoty byly vytvořeny následující metody:

`SetReposition` (Zdrojový kód 5) – metoda pro správu kalibrace (ověření podmínky) srovná, zdali načtený obsah QR kódu odpovídá jedné z hodnot regálových pozic, při nesplnění je uživatel informován. V případě, že nastala shoda, je odeslán název pozice do metody `WorkCycle.TargetScanned` a uživatel je informován o načtení pozice. V dalším kroku je ověření podmínkou `if` (řádek *10,11,12*), jestli došlo k opětovnému naskenování stejného kódu a jestli byl dodržen časový odstup mezi skenováním – pojistka náhodného druhého skenování nebo obdržení vícero hodnot ze skenu. V případě nesplnění je ověřeno (podmínka *OR*), zdali se nejedná o první sken pozice, v případě splnění je spuštěna `ResetFunction`. Navazující podmínka `else if` ověřuje počet skenovaných pozic bez kalibrace, pokud počet dosáhne nastavené hodnoty (`calibrationLimit` – nastaven na 5) je spuštěna kalibrace. V případě nesplnění podmínek je uživatel informován, že pro kalibraci má znovu naskenovat kód, k tomu je navýšen čítač skenů a spuštěna metoda `DisableScannigTimeOut`.

Metoda `ResetFunction` (Zdrojový kód 5) zajišťuje kalibraci, nastaví polohu a rotaci zařízení, na hodnoty, které má naskenovaný `Target` (řádky *33* a *34*). Spustí příkaz `session.Reset,` který provede smazání všech AR prvků v *Session* a jejich opětovné vygenerování, k tomu spustí `arCameraAutoFocusToggle`. Ten vyvolá vypnutí a zapnutí autofokusu kamery zařízení. Tento prvek byl vložen na základě problémů s autofokusem po resetování *AR Session*. Dále je spuštěna metoda `DisableScannigTimeOut`, která navýší časové razítko a obnoví čítač skenů.

```
01 private void SetReposition(string targetText)
02     {
03        TargetObject scannedPos =
04                 targetHandler.GetCurrentTargetByTargetText(targetText);
05         if (scannedPos.Name != null)
06         {
07             WorkCycle.TargetScanned(scannedPos.Name);
08             noticeBar.text = $"naskenována pozice:" + scannedPos.Name;
09             // Reset position and rotation of ARSession
10             if (String.Equals(scannedPos.Name, lastTargetScanned,
11                 StringComparison.OrdinalIgnoreCase) & Time.time >= nextTime
12                 || lastTargetScanned == null)
13             {
14                 ResetFunction(scannedPos);
15             }
16             else if (numTargetScanned > calibrationLimit)
17             {
18                 ResetFunction(scannedPos);
19             }
20             else
21             {
22                 noticeBar.text += $"skenování bylo úspěšné <br> pro kalibraci
23                    naskenuj znovu";
24                 DisableScannigTimeOut();
25                 numTargetScanned++;
26             }
27             lastTargetScanned = scannedPos.Name;
28         }
29         else noticeBar.text = "QR code není v databázi";
30     }
31     public void ResetFunction( TargetObject scanTarget)
32     {
33             sessionOrigin.transform.position = scanTarget.transform.position;
34             sessionOrigin.transform.rotation = scanTarget.transform.rotation;
35             noticeBar.text += $"kalibrace úspěšně proběhla";
36             session.Reset();
37             LoaderUtility.Initialize();
38             arCameraAutoFocusToggle();
39             DisableScannigTimeOut();
40             nextTime += interval;
41             numTargetScanned = 0;
42     }
```

Zdrojový kód 5: Zpracování dat z QR skenu, třída QRCodeRecenter

Třída `QRCodeRecenter` obsahuje ještě další metody:

`ToggleScanning` – změní stav ze skenování *zapnuto* na *vypnuto* a ukončí skenování,
`EnableReadingTimeStop` – zapne skenování na 10 sekund, poté vyvolá `DisableScannigTimeOut`,
`DisableScannigTimeOut` – pokud je skenování zapnuté, dojde k jeho vypnutí,
`LastTargetScaned` – vrátí název (*string*) poslední naskenované hodnoty.

WorkCycle

`WorkCycle` je hlavním kódem pro mód simulace výdejky, obstarává generování výdejky (list objektů `TargetObject`) a postupné procházení tohoto listu. Script je aktivován přepnutím

tlačítka **mód** (ovládáno třídou `Switch`), aktivace začíná metodou `PickingStart` a je obnovována metodou `Update`.

Jednotlivé metody třídy `WorkCycle` jsou následující:

`PickingStart` – smaže předchozí list výdejky a spustí následující metody: `GeneratePointTrack`, `GeneratePointTrackList`, `ToggleCurrentTargetList` a nastaví průběžný cíl na první člen listu.

`EndOfMission` – ukončuje *pickování*[9], metoda nastává po dokončení výdejky nebo přepnutí režimu, metoda vykonává vypnutí *pickování*, vyprázdnění listu výdejky a další nastavení na počáteční hodnoty.

`SetNextTarget` – metoda přenastaví nový průběžný cíl, přepisuje pozici v `NavigationController.TargetPosition` – pro tvorbu nové navigační linie, přepisuje jméno dalšího cíle v `NavigationController`, mění nápis na informačním tlačítku a přiřazuje nový název průběžnému cíli `showEndPoint.currentTargetName`.

`TargetScanned` – metoda je volána při každém načtení QR kódu ve třídě `QRCodeScanner`, podmínka porovnává shodu skenované hodnoty se současným cílem navigace, v případě shody nastaví `targetScannedSuceed` hodnotu `true` a informuje uživatele, opačně nastaví `false` a rovněž informuje o chybě uživatele.

`GeneratePointTrackList` – generuje textový seznam pozic obsažených v listu výdejky (`currentPointTrack`), list je procházen příkazem `foreach` viz. Zdrojový kód 6, řádka *05*.

`GeneratePointTrack` – generuje list `currentPointTrack` typu `TragetObject` o náhodné velikosti v rozmezí **0–10 pozic** (`maxNumberPositions`). Každá pozice je naplněna příkazem `for,` který generuje náhodně zvolený prvek z listu regálových pozic. Kód metody je zapsán ve Zdrojový kód 6 řádka *13–25.*

`Update` – v případě této třídy slouží ke kontrole, jestli byla při skenování načtena požadovaná regálová pozice dle pořadí (znázorněno ve Zdrojový kód 7). Obdobně jako v případě `NavigationControler,` je i zde obsah metody prováděn v časovém intervalu 0.5 s (vnoření do podmínky `if` s kontrolou časového razítka). V případě splnění podmínky časového razítka a povolení *pickování* je zpracována podmínka, ověřující hodnotu proměnné `targetScannedSuceed`. V případě potvrzení, že se jedná o správnou pozici, je uživateli oznámeno že splnil pozici. Pokud se jedná o poslední pozici, je uživatel informován „mise splněna“ a spustí se metoda `EndOfMission`. Pokud se nejedná o poslední pozici, je přiřazena další pozice k navigaci.

`ToggleCurrentTargetList` – přepíná zobrazení listu výdejky a v závislosti na to mění nápis na tlačítku *Zavři list/Zobraz výdejku.*

[9] Pickování – vybírání produktů z pozic, odvozeno z anglického slova pick (vybírat)

`RewriteDestinyBut` – přepisuje zobrazený text na informačním tlačítku *DestinationTextBut*.

```
01 private void GeneratePointTrackList()
02   {
03       int targetListIndex = 0;
04       targetList.text = ("<b>Výdejka </b><br>");
05       foreach (var pointTrack in currentPointTrack)
06       {
07           targetList.text += targetListIndex.ToString()
08                + $". pozice:" + pointTrack.name + $"<br>";
09           targetListIndex++;
10       }
11       targetList.text += "zavři tlačítkem Zavři list";
12   }
13   private void GeneratePointTrack()
14   {
15       int numberOfStops = Random.Range(1, maxNumberPositions);
16       Debug.Log("number of stops:" + numberOfStops.ToString());
17       //filling the list
18       for (int i = 0; i < numberOfStops; i++)
19       {
20         int targetToGo =
21               Random.Range(0, TargetHandler.CountOfPositions);
22         Debug.Log("target index rand: " + targetToGo.ToString());
23         currentPointTrack.Add(TargetHandler.GetTargetByIndex(targetToGo));
24       }
25    }
```

Zdrojový kód 6: Generování výdejky v třídě WorkCycle

```
01 public void Update()
02   {
03       if (Time.time >= nextTime && pickingEnable)
04       {
05           //target scanned
06           if (targetScannedSuceed)
07           {
08               noticeBar.text +=
09        "<br> pozice úspěšně splněna, postupuj na další";
10               if(currentTargetIndex == currentPointTrack.Count)
11               {
12                   noticeBar.text = "mise splněna";
13                   EndOfMission();
14               }
15               else
16               {
17                  currentTargetIndex++;
18                  SetNextTarget(currentTargetIndex);
19               }
20               targetScannedSuceed = false;
21           }
22           nextTime += interval;
23       }
24   }
```

Zdrojový kód 7: Update třídy WorkCycle

4.3.2. Target Models

V této složce jsou uložené třídy určující vlastnosti objektu `Target` (regálová pozice), umožňující čtení ze souboru JSON a manipulaci s těmito objety.

Popis tříd je následující:

Target

Třída určuje, jaké má atributy objekt Target, atributy jsou:

```
string – Name,
Vector3 – Position,
Vector3 – Rotation.
```

Target Wrapper

Třída, sloužící pro extrahování pozic ze souboru JSON, je tvořena atributem:

`IEnumerable <Target> TargetList` (list třídy `Traget`).

Target Object

Třída, reprezentující objekt vygenerovaný na scéně (Unity Scene), atributem je:

`string` **`Name.`**

4.3.3. Utilities

Doplňující kódy, které rozšiřují nebo podporují funkci hlavních kódů (*Core Functions*), jejich výpis s funkcionalitou je následující:

ButtonTextChanger

Třída pro přejmenování tlačítka (*button*), v projektu je zahrnuta, protože třída tlačítka *TextMeshProButton* neumožňuje přímou změnu textu. Ten je totiž jeho potomek, proto je nutné volat textový objekt tlačítka (potomka TextMeshProUGUI) právě metodou:

`RewriteBut` – vstupní atributy jsou název tlačítka a požadovaný text.

Image Tracking

Image Tracking spravuje sledování obrázku, v tomto případě markeru regálové pozice. Tato třída obsahuje metodu `OnChange`, která je převzata z Unity příručky [26]. Její funkcí je při zaznamenání obrázku z databáze sledovaných obrázků aktivovat čtečku QR kódů (Zdrojový kód 8 řádka *03*), což umožňuje uživateli číst markery bez nutnosti aktivace tlačítka *Scan*. Tato třída byla deaktivována, více v kapitole 5.2.1.

```
01 void OnChanged(ARTrackedImagesChangedEventArgs eventArgs)
02     {
03         foreach (var newImage in eventArgs.added)
04         {
05             QRCodeScanner.EnableReadingTimeStop();
06         }
07
08         foreach (var updatedImage in eventArgs.updated)
09         {
10             // Handle updated event
11         }
12
13         foreach (var removedImage in eventArgs.removed)
14         {
15             // Handle removed event
16         }
17     }
```

Zdrojový kód 8: Image Tracking metoda OnChanged

Intro Menu

Spravuje počáteční kalibraci a nastavuje, který *canvas* je viditelný. Také vyvolává případné skenování markeru.

Show End Point

Třída spravuje zobrazení nebo skrytí AR objektů informujících o cílové regálové pozici. Objekty, které třída spravuje jsou *Arrow* a *DestinationFrame* (více o AR objektech v kapitole 4.4.4), oba objekty k sobě mají přiřazený text (*TMP_Text*). Prvky jsou zobrazovány a skrývány příkazem `.SetActive` (viz Zdrojový kód 9 řádka *10*), text zobrazující se na *DestinationFrame* je upravován podle názvu aktuálního průběžného cíle.

```
01public void StartShowing(Vector3 TargetPosition)
02     {
03          if (!ArrowEnable)
04         {
05           ArrowGroup.transform.position = TargetPosition;
06           ArrowGroup.transform.rotation =
07                TargetHandler.GetCurrentTargetRotation();
08           Debug.Log(TargetHandler.GetCurrentTargetRotation().ToString());
09           ArrowEnable= true;
10           ArrowGroup.SetActive(ArrowEnable);
11         }
12     }
```

Zdrojový kód 9: Metoda StratShowing třídy ShowEndPoint

Switch

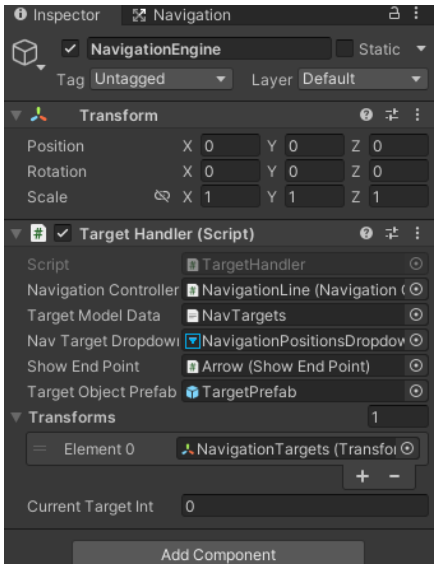
Hlavní funkcí třídy je správa změny módu navigace. Ve třídě je pouze jedna metoda, která je volána stisknutím tlačítka *Mode*, metoda se jmenuje `ChangeVisualization`. Na základě současného režimu je rozevírací seznam buď zobrazen nebo skryt, to platí i o tlačítku zobrazující výdejku. V návaznosti na to metoda volá `workCycle.PickingStart` nebo `workCycle.EndOfMission`, tyto metody poté provedou změnu režimu.

4.3.4. Propojení scriptů s objekty v Unity

Zdrojové kódy jsou v Unity aktivní pouze pokud jsou přiřazeny objektu. Přiřazení objektu je možné před vytvořením samotného scriptu nebo až po jeho vytvoření. Výsledek připojení scriptu k objektu je na Obr. 25, v tomto případě je připojen script *TargetHandler* k objektu typu *Empty Object*: *NavigationEngine*. Script je ale běžně připojován i přímo k funkčním objektům jako je tomu v aplikaci u spojení třídy `NavigationControler` – *NavigationLine* (*Script – Object*).

Ovládání objektů, spouštění metod z jiných scriptů a např. čtení hodnot z druhých scriptů je v Unity řešeno pomocí propojení mezi jednotlivými scripty. K propojení je nutné vytvořit odkazy mezi jednotlivými scripty nebo mezi scriptem a požadovaným objektem (nebo obráceně např. *Button – OnClick*).

Aby toto propojení mohlo vzniknout, je nejdříve nutné v kódu vytvořit instance jiných tříd, pokud je požadavek spravovat instanci jako soukromou (`private`), je nutné pro toto spojení použít třídu *SerializeField*, ukázka je znázorněna v Zdrojový kód 10. Na základě tohoto výčtu se vygeneruje pole odkazů, které je třeba vyplnit v Unity, výsledek po vyplnění je na Obr. 25. Je možné si povšimnout, že pokud proměnné ve skriptu přiřadíme vlastnost `public`, tato proměnná se také vypíše v poli odkazů u daného scriptu v Unity.



Obr. 25: Script Target Handler připojený k Empty Object NavigationEngine

```
01  public class TargetHandler : MonoBehaviour
02  {
03      [SerializeField]
04      private NavigationController navigationController;
05      [SerializeField]
06      private TextAsset targetModelData;
07      [SerializeField]
08      private TMP_Dropdown navTargetDropdown;
09      [SerializeField]
10      public ShowEndPoint ShowEndPoint;
```

Zdrojový kód 10: Výčet instancí cizích tříd ve třídě TargetHandler (výběr)

4.4. Uživatelské prostředí aplikace

Uživatelské prostředí vychází z návrhu aplikace v kapitole 0, rozdíl oproti návrhu je možné pozorovat především v proměnnosti zobrazených tlačítek v závislosti na režimu a zobrazujícím se textovým polím (notifikační textové pole a tlačítko zobrazující cílovou pozici). Cílem bylo vytvoření jednoduché a přehledné aplikace, která však nebude „bránit“ uživateli ve výhledu na zprostředkovaný obraz z kamery.

Aplikace se spouští z prostředí Android jako běžné aplikace a je opatřena ikonou pro snadnou rozpoznatelnost. Ikona aplikace je na Obr. 26.

Obr. 26: Ikona aplikace IAWN

Výchozím zobrazením po spuštění aplikace je obrazovka počáteční kalibrace, po zkalibrováni se uživateli zobrazí obrazovka manuálního režimu navigace.

V jakém režimu se uživatel zrovna nachází je patrné podle spodního pravého tlačítka, pokud tlačítko zobrazuje název regálové pozice jedná se o manuální zadávání, v případě zobrazení *Zobraz výdejku/Zavři list* je aplikace používána v režimu simulace výdejky.

Uživatelské prostředí je vytvořeno v Unity pomocí objektu **UI – Canvas** (v překladu: plátno), tomu je přiřazena velikost v pixelech 1080 x 1920 (režim portrét – „na výšku“). V aplikaci je povolena pouze poloha *portrét* (na výšku). Prvky zobrazení jsou umístěny na jednotlivá plátna, díky čemuž je možné snadno přepínat zobrazení. Hierarchické rozdělení je následující:

Aplication Canvas

QRScanEnviroment
IntroMenuCanvas
BottomMenuCanvas

V základním *canvasu* je přímo umístěn pouze prvek *NoticeBar* – notifikační řádek (*TMPro.TextMeshProUGUI*), který je vždy viditelný a nezávislý na používaném režimu. Prvky jednotlivých zobrazení jsou rozebrány v následujících podkapitolách.

4.4.1. Intro menu

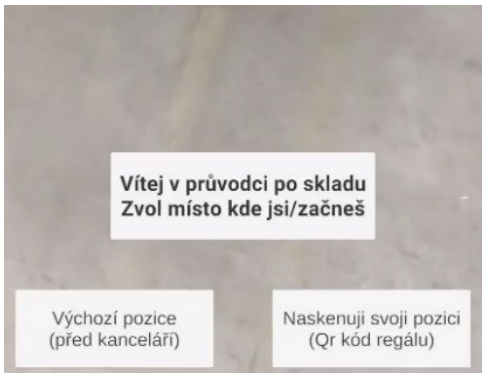
Toto zobrazení se uživateli ukáže v případě, že právě spustil aplikaci. Jedná se o úvodní obrazovku, která slouží pro určení první polohy zařízení ve skladu. Uživatel má na výběr ze dvou možností: kalibrace na výchozím místě nebo kalibrace naskenováním regálové pozice. Menu je zobrazené na Obr. 27.

Pokud uživatel zvolí možnost kalibrace na výchozí pozici, musí dojít před kancelář na vyznačenou pozici a nasměrovat telefon ve vyznačeném směru (na sever) a poté potvrdit, že je zařízení v referenční pozici. Pro kalibraci skenem pozice stačí naskenovat libovolný marker a aplikace se zkalibruje na tuto pozici.

Když se aplikace spustí, je zobrazen *canvas IntroMenuCanvas*, ostatní jsou skryté. Toto plátno je tvořené pouze třemi tlačítky. Funkční stránku k tomuto menu spravuje script `IntroMenu`, ve skriptu je rovněž naprogramované skrytí tohoto plátna po dokončení kalibrace.

4.4.2. Bottom Menu Canvas

Po dokončení kalibrace je zviditelněné plátno *BottomMenuCanvas*, které je společné pro oba režimy, jak manuální zadávání, tak simulace výdejky. Je tomu tak především, protože podoba obou menu je z velké části stejná a pokud by byla plátna pro každé zvlášť, docházelo by ke zdvojování prvků, tím pádem by se musely při úpravě měnit prvky dvakrát a mohla by se vytvořit neheterogenita.



Obr. 27: Úvodní menu pro kalibraci počáteční polohy – výřez obrazovky (spodní třetina)

V následující tabulce (Tab. 3) je vypsáno složení prvků v *canvas BottomMenuCanvas* s typem entity a popisem.

Tab. 3: Složení canvasu Bottom Menu

Název entity	Druh entity	Popis
MiniMapRawImage	Raw Image	Minimapa
LineVisibility	Button TextMeshPro	Přepínání viditelnosti navigační linie
NavigationDropDown	Drop Down TextMeshPro	Rozevírací seznam regálových pozic
QRScanBut	Button TextMeshPro	Tlačítko skenování – zapnutí/vypnutí
DestinationTextBut	Button TextMeshPro	Ukazatel názvu cíle
ShowPickListBut	Button TextMeshPro	Zobrazení/skrytí výdejky
PickingPointCanvas	Canvas	Plátno pro textový výpis listu výdejky

Jednotlivé pohledy, které jsou zobrazeny uživateli jsou pospány v následujících odstavcích.

Manuální zadávání

Zobrazení vychází z navrhovaného rozložení, ve spodní části obrazovky se nachází mapa a funkční tlačítka. Na této obrazovce lze volit cílové pozice z rozevíracího seznamu (*TMP Drop Down*). Po zvolení se název cílové pozice propíše do informačního tlačítka (*Button TextMeshPro*), díky zobrazení cílové pozice získává uživatel průběžný přehled, na jakou pozici směřuje. V případě manuálního režimu se zobrazuje název cílové pozice i na rozevíracím seznamu.

Informační tlačítko má kromě zobrazení ještě „*skrytou*“ funkci pro zobrazení atributů trasy. Jedná se o dvojité poklepání, které do notifikačního řádku vypíše délky trasy a počet uzlů. Tato informace slouží při testování spolehlivosti jako kritérium náročnosti trasy.

Dalšími funkčními tlačítky jsou tlačítka pro přepínání režimu (*Mód*) a tlačítko pro změnu viditelnosti navigační linie (*Zobr. Navigace*). Významným tlačítkem je *Scan*, toto tlačítko zapíná nebo vypíná režim čtení QR kódu, po zapnutí se zobrazí pomocný „terč“, který pomáhá uživateli se zaměřením skenovaného markeru. V případě úspěšného naskenování je terč skryt.

Obr. 28: Uživatelské prostředí aplikace v režimu manuálního zadávání

Simulace výdejky

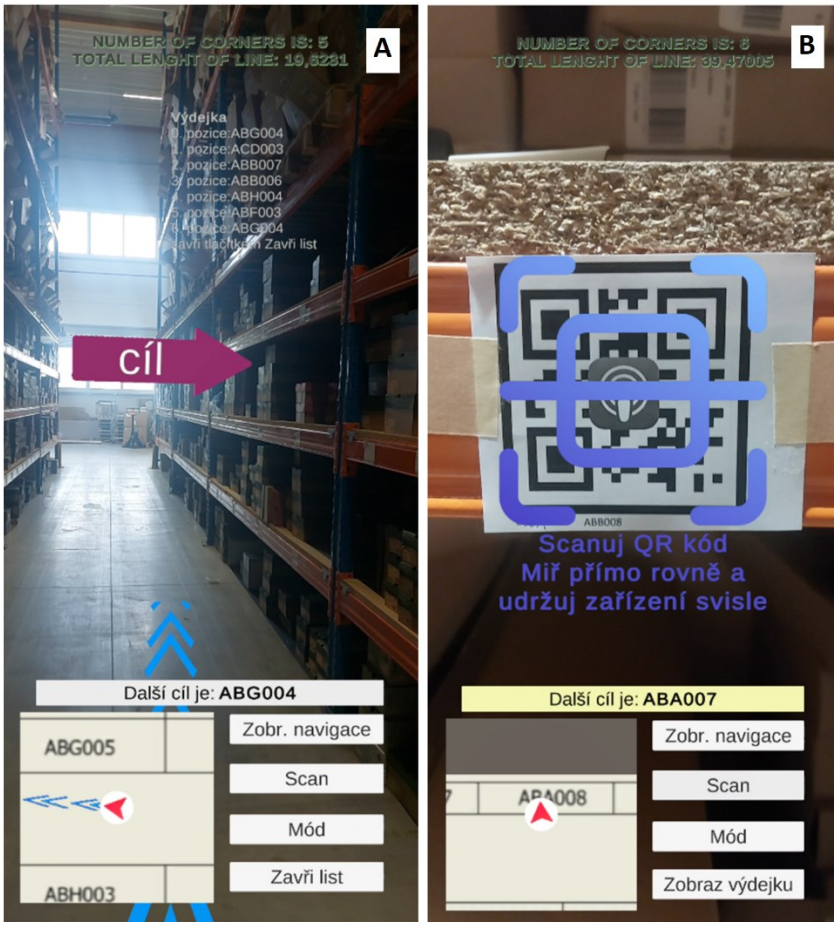
Zobrazení je velmi podobné manuálnímu zadávání, rozdílné je pouze to, že místo rozevíracího seznamu je zobrazené tlačítko *Zobraz výdejku/Zavři list*. Snímek obrazovky je na Obr. 29 část A. Hlavním kódem, který tvoří funkčnost pro tento režim je `WorkCycle`.

Stisknutím tlačítka *Zobraz Výdejku* dojde k zobrazení plátna *PickingPointCanvas*, na kterém se uživateli vypíše list výdejky pomocí prvku TextMeshPro: *TMPro.TextMeshProUGUI*. Ten je naplněn vygenerovanými názvy regálových pozic. Pro skrytí seznamu musí uživatel stisknout tlačítko *Zavři list*.

4.4.3. QR Scan Enviroment

Jedná se doplňující *canvas*, který se zobrazuje společně s dalšími plátny, při jeho zobrazení se ostatní neskrývají. Jedná se dvojici pláten, jedno plátno obsahuje text a druhé (podřazené) zobrazuje obrázek jako texturu. Obrázkem je terč, který je částečně průhledný, proto aby umožnil uživateli zamířit správně na marker, který chce načíst viz. Obr. 29 část B.

Druhým prvkem, který je na tomto *canvasu* je text (*TMPro.TextMeshProUGUI*), který je nastaven do jedné z barev zaměřovacího terče a funguje jako nápověda pro uživatele. Funkcionalita skenování a zobrazování je ovládaná skrze script `QRCodeRecenter`.



Obr. 29: Prostředí aplikace: část A – režim WorkCycle, B – obrazovka skenování QR kódu

4.4.4. AR prvky

V aplikaci se uživateli zobrazují prvky rozšířené reality, které nejsou závislé na tom, v jakém režimu je zrovna aplikace používána. Aplikace zobrazuje tři prvky rozšířené reality (zobrazeny na Obr. 30) a ty jsou následující:

- **šipka ukazující cíl** (*Arrow*)
- tabulka oznamující **název cílové pozice** (*DestinationFrame*)
- **naváděcí linie** (*NavigationLine*)

Arrow

Neboli šipka je v Unity vložena jako 3D model ve formátu *FBX*, modelu je přiřazen materiál (fialová barva) a šipka je dále doplněna textem *TMP_Text.* Zobrazuje se text: „CÍL“, který je umístěn na obou stranách šipky. Tento text je stálý a není v průběhu chodu aplikace měněn.

Jak již bylo popsáno v kapitole 4.3, zobrazení *Arrow* v prostoru je řízeno třídou `ShowEndPoint`. Její hrot je umístěn na souřadnici cílové skladové pozice.

Destination Frame

Tento objekt je vytvořen pomocí elementu *Cube*, kterému je přiřazen materiál (žlutá barva), doplněn je stejně jako *Arrow,* textem *TMP_Text* z obou stran. Na rozdíl od šipky **je text** této skupiny **měněn** na základě názvu cílové regálové pozice, tedy např. jako na Obr. 30 „pozice: ABA009“.

Naváděcí linie

K zobrazení naváděcí linie je využit komponent *Line Renderer*. Tvar křivky je generovaný na základě *NavMesh* a cílové destinace, podoba křivky je ovlivněna přiřazeným materiálem, ten je zde v podobě textury. Obrázkem, použitým pro zobrazení linie, je dvojitá modrá šipka. Textura se při renderování křivky „kopíruje za sebe“ (*Texture Mode: Tile*) a vytváří tak výslednou podobu směrové naváděcí čáry.

Obr. 30: Vygenerované prvky AR v testovacím prostředí Unity

5. Testování a zhodnocení aplikace

Aplikace, jak již bylo zmíněno, byla vytvořena pro navigování ve skladu společnosti Outdoor Concept a. s., forma, kterou byla vytvořena (kapitola 3.2) podmiňuje její funkčnost pouze pro to prostředí. Proto její závěrečné testování bylo prováděno výhradně v prostorách skladu Outdoor Concept a. s. K testování aplikace lze využít i prostředí Unity, kde je simulován chod aplikace, tato možnost se hodí spíše pro průběžné testování např. akcí typu *OnClick* tlačítka atd.

5.1. Metodika testování

Testování aplikace bylo prováděno jak v průběhu vývoje aplikace, tak následně po sestavení prototypu. Jelikož je aplikace vázaná na prostředí konkrétního skladu bylo možné testovat při vývoji v domácím prostředí pouze určité funkce např. čtení markerů. Funkčnost navigace a orientace v prostoru bylo možné testovat pouze ve skladu.

Z toho lze vyvodit, že se testování dělilo na:

- testování mimo sklad
- testování v prostorách skladu

Ve skladu společnosti Outdoor Concept a. s. Testování bylo mířeno na zkoumání přesnosti a stability navrženého řešení. Zkoumáno bylo také celkové chování aplikace při navádění uživatele do cíle. Pro testování byl do aplikace doplněna metoda `GetReportOfLine` pro určení „náročnosti“ trasy, která spočetla délku trasy a počet uzlů, ze kterých se skládá.

Body, které byly zaznamenávány **při testování každé trasy**:

- trasa (mezi jakými pozicemi byla)
- délka trasy [m],
- počet uzlů,
- odchylka [m],
- dovedení do cíle (ANO/NE),
- druh chyby,
- komentář.

Měření odchylky bylo prováděno za pomoci porovnání polohy zobrazované pomocí cílové šipky (její poloha je určena souřadnicí skladové pozice) a fyzické polohy skladové pozice (markeru). Vzdálenost byla měřena ručním metrem, tato skutečnost bohužel negativně ovlivňuje přesnost určení odchylky.

Do komentáře k trase je vepsán nestandardní popis situace, která nastala při procházení trasy. Takové situace lze dělit v základu na dvě skupiny a to jsou: **kritické chyby**, které vedou přímo ke ztracení uživatele a nedovedení do cíle a **chyby nekritické**, které nezpůsobí vážný výpadek nebo ztrátu pozice jako např. výpadek zobrazení navigační linie.

Testování bylo provedeno na zařízení **Samsung Galaxy A52 5G**, které podporuje ***ARCore Depth API***. Kromě tohoto zařízení byla aplikace nainstalována také na mobilní telefon Xiaomi Redmi Note 7. Na tomto zařízení se aplikace nechovala stabilně, u pozdějších verzí se nespustila aplikace vůbec (pouze černá obrazovka).

Příprava na testování

Před zahájením každého testování ve skladu bylo nutné opatřit skladová místa vytištěnými markery, aby mohly být načítány regálové pozice. Z toho důvodu, že ve společnosti jsou využívány nové ruční skenery, které dokážou číst kromě čárových kódů i QR kódy, musely být **markery** vždy po testování **odstraněny**. Důvodem bylo především, že současně používané čárové kódy, označující danou regálovou pozici, se nacházeli v blízkosti umístěných markerů a mohli by tak způsobovat zmatení pracovníků a chybné skenování pozic, což by bylo nežádoucí.

5.2. Testování aplikace

Jak již bylo zmíněno, aplikace byla testována jak v průběhu vývoje, tak se podrobila i rozsáhlejšímu testování v závěru tvorby, které probíhalo přímo ve skladu. V následujících podkapitolách je podrobně popsáno průběžné testování při vývoji a testování ve skladu.

5.2.1. Testování dílčích funkcí mimo sklad

V průběhu vývoje aplikace bylo u dílčích částí aplikace, u kterých to bylo možné, provedeno testování mimo sklad. Výhodou testování při vývoji je, že bylo možné reagovat na chyby okamžitě a ihned je opravit. Testování bylo prováděno jak ve vývojové aplikaci Unity, tak v mobilním telefonu.

Testování simulace v Unity skýtalo oproti testování v mobilním telefonu výhodu snadného „debuggování"[10] a tím pádem hledání příčin možných chyb v běhu aplikace. V Unity však nebylo možné testovat prvky pracující s kamerou, pro takové případy bylo nutné sestavit aplikaci a nahrát ji do mobilního telefonu.

V podstatě se tímto způsobem testovala celá aplikace kromě samotného navádění, to je podmíněné prostředím skladu.

Funkce **Image Tracking** byla testována pouze v testovací verzi při vývoji aplikace a nebyla zahrnuta ve finální verzi aplikace. Původní myšlenka zakomponování Image Tracking bylo spínání čtečky QR kódu při zachycení markeru. Testováním se zkoumala možnost, jestli jsou vytvořené markery dostatečně sobě podobné tak, že by bylo možné vložit do knihovny obrázků pouze jeden obrázek markeru a funkce by se spínala pro všechny markery.

Z provedených testů vyplynulo, že pro AR Foundation Image Tracking jsou i přes vložení ikony a výrazného rámečku jednotlivé markery pozic unikátní. Když byl tedy vložen jeden marker do knihovny, algoritmus jiný nerozpoznal. Pro funkčnost by bylo nutné vložit všechny používané markery do knihovny v Unity. Další možnost by byla vytvořit složku s obrázky v Unity, ze které by se automaticky po spuštění aplikace nahrály obrázky do knihovny Image Tracking.

[10] debugging – průběžný výpis předem nastavených průběžných hlášení z chodu programu, slouží pro ladění programu

5.2.2. Testování I. – ověření konceptu

Jednalo se o první test chování prototypu aplikace v prostorách skladu. Aplikace byla předtím laděna pouze v prostředí Unity a na mobilním telefonu bez možnosti ověření funkce navádění.

Cíl testování

Cílem tohoto testování bylo ověření prvotního návrhu aplikace, především schopnost určovat pozici a navigovat uživatele ve skladu.

Obsah testování

První testování ve skladu bylo zahájeno s omezeným množstvím pozic a pouze v prostorách haly ACx. V této fázi byla aplikace vytvořena pouze v režimu manuálního zadávání a regálové pozice byly ručně vytvořené pomocí objektů *Empty Object*.

Výsledek testování

Výsledkem testování bylo, že navržený koncept aplikace vykazoval schopnost určovat polohu a navigovat uživatele. Aplikace byla schopná číst markery pozic a provádět kalibraci na základě jejich načtení. Z tohoto měření nebyl pořízen záznam tras.

Testem této verze se potvrdil potenciál řešení pro navádění, na omezeném počtu pozic navádění fungovalo velmi slibně. Při používání byly nalezeny dílčí nedostatky v navrženém systému, např. jako problematické se jevilo spuštění pouze na kalibrační pozici, po prvním spuštění byla navigační křivka velmi vysoko, navigační křivka často protínala rohy regálů, uživatel nebyl informován, na jakou pozici je naváděn. Při skenování markeru byl vyvolán restart prvků ve virtuální scéně (*AR Session*), který měl za následek výpadek funkce autofokusu kamery.

Úpravy aplikace v návaznosti na testování

V prvním testování byla ověřena funkčnost konceptu, bylo však nutné změnit řadu nedokonalostí: např. výška navigační křivky nebo problémy s autofokusem kamery po skenování, aj.

Před dalším testováním byl do aplikace doplněn režim simulace výdejky, souřadnice Y navigační AR linie byla snížena a došlo také k **rozšíření navigace na další halu ABx**. Načtení pozic v této verzi probíhalo již z databázového souboru formátu *JSON*, ten byl naplněn vybraným množstvím náhodně zvolených regálových pozic. Byl doplněn příkaz zapnutí autofokusu kamery po restartu *AR Session*.

V reakci na zjištěné nedostatky byly přiřazeny do *NavMesh* prvky *NavTrajectoryAdjustment*, které upravují generování naváděcí křivky, více o úpravě trasy v kapitole 4.1.2. Rovněž bylo přidáno notifikační tlačítko, které zobrazuje, na kterou pozici uživatel směřuje. Rozšířené byly také AR prvky o informační tabuli *DestinationFrame*. K nastavování výšky naváděcí linie byl využit prvek **AR Plane manager**, který umožnil stanovit pevně výšku naváděcí křivky v závislosti na ploše podlahy. Problém s ostřením byl vyřešen pomocí vynuceného vypnutí a zapnutí autofokusu AR kamery. Počet pozic byl rozšířen, tak aby pokrýval co největší plochu. Ve výsledku byl počet zanesených regálových pozic roven **33 markerům**. Po zapnutí může uživatel volit mezi kalibrací pomocí skenu markeru skladové pozice nebo postavením se na kalibrační pozicí.

5.2.3. Testování II. – přidání pozic, simulace vychystání výdejky

Druhé testování bylo význačné především tím, že se testovala řada změn, které byly vytvořeny po testu prototypu. Rovněž byla zařazena do testování dílčí část skladu a rozšířen počet skladovacích pozic.

Cíl testování

Cílem testování bylo ověřit chování aplikace zejména na delších trasách a podrobit aplikaci navigování několika tras po sobě. Rovněž se také jednalo o ověření oprav nedostatků z prvního testování.

Obsah testování

Testování bylo provedeno v halách ACx a ABx, s celkovým počtem 33 skladových pozic. V průběhu testu bylo procházeno 30 různými trasami, jejichž průběh byl zaznamenán do tabulky viz. Příloha 2. Test probíhal v režimu simulace výdejky, podle níž byly procházeny postupně jednotlivé pozice z vygenerované výdejky. V testu byly měřeny náležitosti trasy a přesnost, jakou navigace v cíli vykazovala. Případné projevy navigace byly zaznamenány v komentáři trasy.

Světelné podmínky testování byly příznivé, bylo oblačno, z čehož neplynuly velké světelné rozdíly mezi uličkami se světlíky a bez.

Výsledek testování

Výsledkem měření bylo, že navigace byla úspěšná v 73 % případů viz. Tab. 4, což lze považovat za působivou hodnotu. Při bližším pohledu na data je však možné pozorovat, že se objevují i poměrně velké odchylky, např. 2,9 m, 1,5 aj.. Přestože průměrná odchylka je 0,45 m, 50 % tras dovedených do cíle vykazuje odchylku větší než 0,3 m. Směrodatná odchylka byla při tomto měření větší než 1, z čehož plyne že rozdíly v přesnosti navigace jsou poměrně velké.

Tab. 4: Souhrn měření spolehlivosti navigace

Počet pokusů	Dovedení do cíle	Průměrná odchylka	Směrodatná odchylka	Průběžný výpadek navigace (nekritická chyba)
30	22 (**73 %**)	0,45 m	1,12 m	20 %

Kromě číselných výsledků je nutné věnovat pozornost i nestandardním událostem, které provázely testování. Ve 20 % případů byly pozorovány výpadky navigace, např. i přesto, že pokus o navigaci mezi pozicemi ABB005 – ACF002 dopadl úspěšně, cestou byly zaznamenány výpadky navigace v oblasti u zadní stěny.

V průběhu testování bylo zaznamenáno vícero případů s problémy u východní stěny skladu, prostor je v této části relativně úzký (1,3 m) a tmavý (zastínění seshora regálem). Kromě toho jsou u stěny vyskládané krabice, což tvoří složitější a ještě užší prostředí. Kontrast k tomu tvoří některé pokusy, kdy i přesto, že trasa procházela tímto prostorem, výsledná odchylka byla minimální.

Výpadky byly převážně způsobeny mírnou odchylkou polohy zařízení ve virtuální scéně, která kvůli úzkému prostoru (malá vzdálenost uživatele od regálu) způsobila, že se virtuální pozice

ocitla mimo plochu *NavMesh* a nemohla tak být generována navigační trasa. Pokud uživatel pokračoval v ohybu správným směrem, navigace se po chvilce opět zobrazila (virtuální pozice byla opět na ploše *NavMesh*).

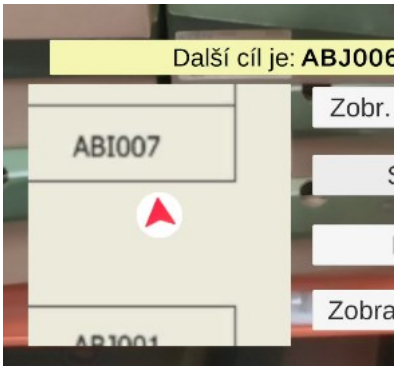
V průběhu testování byl často zaznamenán problém s relativní rotací zařízení v prostoru (rotace v aplikaci byla posunutá vůči realitě), což vedlo ke špatné navigaci často doprovázené kritickou chybou (27 % tras). Tento případ nastal ve většině případů po **chybně provedené kalibraci pozice a rotace**. Aplikace v této verzi obsahovala automatický kalibrační proces, který po naskenování markeru spustil kalibraci polohy zařízení. Kalibrace proběhla při každém skenu.

Generování trasy kolem regálů bylo v této verzi lépe čitelné díky provedené úpravě. Výška generované navigační linie nebyla uspokojivá, její zobrazení nebylo dostatečně přehledné.

Úpravy aplikace v návaznosti na testování

Hlavní změnou v aplikaci bylo **zrušení automatické kalibrace** po naskenování markeru. K tomuto kroku bylo přistoupeno především na základě zjištění z předešlého testování, ze kterého vyplynulo, že kalibrace je náchylná k chybě. Automatická kalibrace byla změněna na kalibraci podmíněnou počtem skenů bez kalibrace nebo vyvolanou uživatelem. Očekáváním této změny byla větší stabilita a menší chybovost.

Další změnou v aplikaci byla editace výšky navigační linie, která se v předešlé verzi zobrazovala příliš nízko nad zemí. Jednalo se o nápravu chyby, ve které se špatně počítala vzdálenost od země. K této úpravě se váže i další změna zobrazení navigace uživateli a tou je přidání **směrové šipky** pohledu uživatele v zobrazení minimapy. Tento prvek je zlepšením oproti signalizaci pouze polohy (v předešlé verzi tečka) a to především v tom, že umožňuje uživateli lépe se zorientovat. Využitím směrové šipky se může uživatel např. ujistit že je otočený správným směrem. Podoba směrové šipky je na Obr. 31.



Obr. 31: Detail směrové šipky na minimapě aplikace

5.2.4. Testování III. – finální testování

Poslední testování proběhlo na upravené verzi aplikace a soustředilo se především na zkoumání chování aplikace při dlouhodobějším testování.

Cíl testování

Testování bylo zaměřeno na ověření chování aplikace při dlouhodobějším používání a stálost navigace v případě, že kalibrace neproběhne na začátku každé trasy. Dále bylo cílem ověřit spotřebu baterie při intenzivním používání.

Obsah testování

Testování proběhlo na stejném počtu skladových pozic jako předešlé (33 skladových pozic) a rovněž bylo provedeno v režimu simulace výdejky. V případě tohoto testu bylo provedeno celkem 71 průchodů po navigované trase. Každá trasa byla zaznamenána do tabulky viz. Příloha 3.

Test byl proveden za jasného slunečného dne v dopoledních hodinách. Hala je prosvětlená světlíky a okny ze západní strany, ze kterých do haly pronikalo ostré světlo. Mezi testováním II. a III. proběhla na halách demontáž konstrukce, na které byly uskladněny palety, čímž se odkryla horní řada oken a do prostoru začalo vnikat ještě **více světla ze strany.** Při průchodu uličkou, ve které bylo světlo pouze z oken ve stěně a závěsných svítidel (bez světlíku), tvořila tato kombinace **tunelový efekt.** I pro lidské oko bylo místy složité soustředit se na detaily, protože podlaha s cementovou stěrkou a lakované plochy regálů značně odrážely sluneční světlo (pocitově ostré přímé světlo). Naopak, pokud trasa vedla od oken směrem ke zdi a ulička byla opatřena světlíkem, světelné podmínky byly daleko přívětivější a detaily obrazu zřetelnější (měkké světlo rovnoměrně jdoucí svrchu). K podmínkám při testování je dobré doplnit, že při testování probíhaly stále práce na demontáži konstrukce a k tomu bylo ve skladu prováděno vychystávání zboží v běžném režimu. Bylo tedy často nutné upravovat trasu a neřídit se přímo podle navigace např. kvůli pohybu VZV, skladníků nebo procházejícím dělníkům s materiálem.

Výsledek testování

Záznam z testování je umístěn v Příloze č. 2, tabulka zaznamenává jednotlivé trasy, mezi jakými pozicemi byla trasa naváděna, její délku, počet uzlů a výsledek navigace, tedy navedla-li úspěšně do cíle či ne, s jakou odchylkou a případné chyby v navigaci. Výstupem testování č. III. je souhrn z této tabulky v Tab. 5.

Tab. 5 Výstup z testování aplikace č. III.

Počet pokusů	počet úspěšných pokusů	Průměrná odchylka	Směrodatná odchylka	Průběžný výpadek navigace (nekritická chyba)
71	59 (**83 %**)	0,48 m	0,91 m	26 %

Z výsledkové tabulky (Tab. 5) je patrné, že došlo k mírnému zlepšení navigačních schopností aplikace, dovedení do cíle se zlepšilo o 10 % vůči testu II. a snížila se i směrodatná odchylka pod jeden metr. Zásluhu tohoto zlepšení lze hledat ve větší stabilitě aplikace díky odebrání příliš časté kalibrace. Z naměřených hodnot je vidět, že pokud byla kalibrace správně provedena, není problém projít 3–5 skladových pozic bez nutnosti další kalibrace.

Problém při testování opakovaně nastával v případě **pohybu proti ostrému protisvětlu**, v takových případech měla aplikace problémy správně určovat ušlou vzdálenost a často docházelo k náhodným odskokům naváděcí linie a následně ke špatnému určení regálové pozice. Na displeji zařízení bylo možné pozorovat, že velká část obrazu je v takovém případě přesvětlená a muselo být obtížné detekovat správně body. Někdy však byla aplikace schopna takovou chybu zachytit a dopočíst svou polohu, ale až např. po ujití vzdálenosti jednoho metru. Zvětšení procenta (z 20 % na 26 %) průběžných výpadků navigace lze přičíst tomuto problému, který se v předchozím měření nevyskytoval.

Kromě těchto chyb setrvával problém s občasnými výpadky navigační linie, protože aplikace v určitý moment v průběhu trasy zobrazovala polohu mylně v regálu a tím pádem mimo plochu *NavMesh*, z toho důvodu nemohla být generována trasa. Pokud však uživatel pokračoval v zobrazené trase před výpadkem, došlo po opuštění regálu ve většině případů k obnovení zobrazení a navigování do cíle.

V průběhu testování byla v případě zpozorování větší odchylky zobrazované polohy od skutečnosti (více než 0,3 m) provedena manuálně kalibrace polohy s důrazem na dodržení orientace zařízení, tak aby se zamezilo chybám. Při testování byla verze aplikace nastavena na max. 10 naskenovaných pozic bez kalibrace, k dosažení této hodnoty však nedošlo (vlivem vzniku příliš velké odchylky polohy v průběhu), proto byla následně snížena na max. 5 pozic.

K testování je podstatné doplnit, že probíhalo zhruba 4,5 h s drobnými pauzami. V průběhu testování vykazovalo zařízení zvýšenou teplotu, citelnou na dotek. Dalším zjištěním bylo, že za tuto dobu (o něco málo více než polovina pracovní doby) došlo z plného stavu baterie k jejímu **vybití**. K tomuto faktu je ale nutné uvést, že část testování byl nahráván displej zařízení, což mohlo vést k větší zátěži zařízení.

Obr. 32: Oslnění kamery protisvětlem z bočních oken

Subjektivní zjištění z tohoto testování byl nepříjemný pocit z neustálého sledování displeje za pochodu po skladu. Toto zprostředkované vnímání okolí přes displej vytváří pocit, že se uživatel pohybuje v displeji, a ne v realitě a mohl by být potenciálním rizikem pro střet s pohybujícími se objekt nebo lidmi.

Úpravy aplikace v návaznosti na testování

Z důvodu, že se jednalo o poslední testování aplikace a dále již vývoj nepokračoval, jsou poznatky z testování obsaženy v doporučení pro další vývoj aplikace.

5.3. Zhodnocení aplikace

Aplikace byla podrobena několika testům, zaměřeným na prozkoumání chování aplikace při používání v prostorách skladu. Celkový počet zaznamenaných testů je *101 tras*, tento vzorek sice není na tolik velký, aby mohl pokrýt všechny situace, které mohou při používání nastat, je však tak obsáhlý, že z něj lze vyčíst základní charakteristiky chování takové aplikace v provozu.

Tvorba aplikace

Jednoznačnou výhodou aplikace je povaha její tvorby, a to konkrétně jednoduchost a rychlost se kterou je možné ji vytvořit. Zároveň je možné ji dobře modifikovat např. přidat další sklad nebo část skladu vyjmout z navigace. Pro její zprovoznění není nutná instalace další elektroniky do skladovacích prostor, je pouze nutné opatřit skladovací místa markery.

Schopnost navigování

Navigování uživatele probíhalo na celkové ploše 36 x 40 m, některé z tras byly dlouhé i přes 40 m (viz Příloha 3.) a i na takových vzdálenostech byla aplikace schopna vykazovat minimální odchylku od reality např. měření č. 48, Příloha 3., kde navigace zobrazovala pozici naprosto přesně. Ovšem je třeba připomenout i fakt, že v některých případech aplikace nedovedla uživatele do cíle i na menší vzdálenosti, vždy záleželo na působení negativních vlivů (např. protisvětlo, chyba kalibrace aj.).

V průběhu testování vykazovala aplikace adekvátní odezvy na změny směru, obcházení překážek a další neplánované výchylky z trasy. Pro generování trasy nebyl problém použít jinou souběžnou uličku, než doporučovala navigace, bylo však nutné jít chvíli proti navigaci, než uživatel došel do bodu, kdy byla nejkratší trasa v před a poté se zobrazila navigační linie před uživatelem podle očekávání.

Spolehlivost

Z dat naměřených při testování lze stanovit, že aplikace v této podobě projevuje relativně vysokou spolehlivost dovedení uživatele do cíle. Přesto je nutné brát v potaz průběžné výpadky, které provázeli více než 20 % navigovaných tras.

Z toho je patrné, že aplikace projevuje jisté nedokonalosti, které za určitých okolností stěžují její užívání. Jedním z hlavních problémů je **citlivost na světelné podmínky**, např. problém s orientací v užších místech, které jsou špatně osvětlené nebo naopak s protisvětlem.

Dalším prvkem, který je klíčový pro správné určení polohy je kalibrace. Zejména při druhém testu bylo zjištěno, že přesnost navigace je přímo závislá na správně provedené kalibraci, která je však náchylná na chybovost. Proto je nutné **kalibraci provádět obezřetně** a jak vyplynulo ze srovnání, neprovádět ji při každém skenu.

Pokud je však hodnocena kladná stránka aplikace, v průběhu testování bylo několikrát dosaženo velmi dobrých výsledků přesnosti a to i na delší vzdálenosti (40 m a více). Aplikace reagovala pružně na změny směru v případě např. obcházení překážek v trase a byla i přesto schopna navigovat uživatele do cíle.

Používání

Aplikace je také vytvořena tak aby byla jednoduchá na používání a přehledná pro uživatele. Díky tomu, že zjednodušeně řečeno, uživatel musí pouze „následovat čáru“ je velmi pravděpodobné, že díky tomu bude pro uživatele velmi jednoduché dojít do cíle podle navigace. Pro lepší přehled je cíl označen šipkou, což se při testování velmi osvědčilo, protože mnohdy nebylo z dálky zcela zřejmé, na kterou pozici je uživatel naváděn.

Po úpravě trajektorie kolem rohů byla i navigační čára dobře čitelná a poskytovala lepší přehled o směru navigace.

Účelem této práce nebylo přímo sestavení aplikace pro koncové uživatele, ale otestování, zdali je toho možné dosáhnout se současným hardwarovým vybavením a softwarovými řešeními. Proto prostředí aplikace bylo vytvořené jednodušeji, nežli by tomu bylo v případě určení pro koncové uživatele, určité prvky proto vyžadovali mírně pokročilejší obeznámenost s principem funkce aplikace jako např. kalibrace vyžadovala přidržet telefon po naskenování markeru cca 1–2 s, než proběhla kalibrace polohy.

Jak bylo zmíněno v kapitole 5.2.4, při používání navigace je nutné sledovat displej telefonu a držet jej tak aby mohl snímat směr kterým se uživatel pohybuje. Z této skutečnosti plyne potenciální riziko, že může uživatel ztratit pojem o okolí a dojít ke kolizi. Držení zařízení v ruce rovněž představuje komplikaci pro vykonávání pracovní náplně skladníka, který často při vychystávání výdejky před sebou tlačí v obou rukou *pickovací vozík*.

Technická stránka

Třetí testování, které mělo delší časový průběh přineslo zjištění, že aplikace poměrně výrazně spotřebovává energii baterie zařízení. Konkrétně u testovaného telefonu je baterie o kapacitě **4500 mAh**, která **vydržela 4 h** testování.

Aplikace v této podobě spoléhá na určení polohy modulem *ARCore*, který využívá primárně obrazové rozpoznávání. Z tohoto modulu je přímo získaná poloha, která není dále modifikována, poloha je tedy zcela závislá na algoritmech sestavených společností *Google*. Výhodou této kombinace je, že při programování nebyla řešena problematika rozpoznávání obrazu a v použité SDK *ARCore* jsou zahrnuty také pokročilé prvky *machine learning*, které jsou neustále vyvíjeny a umožňují dokonalejší zpracování obrazu i v případě výpadků. Na druhou stranu poloha není v aplikaci dále upravována kromě kalibrace.

V aplikaci je úspěšně implementován prvek pracující s obrazovým rozpoznáváním okolí, založený na funkci *AR Foundation – AR Plane Manager*, který upravuje výšku naváděcí linie.

Aplikace je určena pro zařízení s operačním systémem Androidem a z pokusu o spuštění na starším zařízení *Redmi Note 7*, na kterém se aplikace nespustila, je možné vyvodit závěr, že pro bezproblémový chod je vhodné zvolit zařízení s dostatečně výkonným hardwarem a jedním z nejnovějších verzí systémů Android.

5.4. Další vývoj aplikace

Další směřování aplikace je možné vidět ve zdokonalení určení polohy, tak aby se eliminovala náchylnost aplikace na světelné podmínky. Z přehledu možností pro určení polohy v prostoru v teoretické části práce se nabízí možnost soustředit se pouze na obrazové rozpoznávání a využít technologii Area Target vytvořenou společností Vuforia. V případě této technologie je však potřeba zvážit proměnnost objektů ve skladu (zaskladnění/vyskladnění krabic v regálu, postavená paleta na jiném místě aj.), ta by mohla způsobovat nepřesnosti a potíže v určení polohy. Za úvahu v tomto směru také stojí, zda není prostředí testovaného skladu příliš stejnorodé pro technologii Area Target – stejné regály a velmi podobný obsah polic.

Zajímavou možností by byla kombinace s jednou z radiových technologií, např. UWB. Jelikož z pozorování vyplývá, že pokud byla navigace provedena v rámci jedné uličky, byl výsledek navigování velmi přesný, nabízí se zde idea hybridní kombinace. Možnou kombinací by mohla být např. kalibrace polohy bezdrátovou radiofrekvenční metodou v hlavní uličce a následná navigace na bázi prostorového vnímání ARCore uličkou regálu (bez kalibrace po načtení markeru).

Pozornost je třeba věnovat také samotnému používání aplikace, jak jej hodnotí uživatel a jestli jej neomezuje nebo neohrožuje. Při testování byla provedena i zkouška s „pickovacím klecovým vozíkem“, který je používán při vyskladnění z regálů. Při tlačení vozíku a používání aplikace (telefon nasměrován vpřed a držen v ruce) bylo obtížné zatáčet s vozíkem (prázdný vozík) a směřovat jej správným směrem, vozík navíc „zastiňoval“ výhled mobilnímu telefonu. Tato kombinace tedy nebyla vůbec vhodná, řešením by mohl být přechod na platformu chytrých brýlí, což by pracovníkovi uvolnilo ruce, bez ztráty zobrazení AR prvků. Výhodou by mohl být i fakt, že brýle bývají zpravidla opatřeny vícero kamerami/senzory než mobilní telefony. Případný přechod aplikace na tuto platformu by neměl být zásadně problémový, protože vývoj se provádí také v Unity.

V dalším vývoji by bylo přínosné zkoumat vliv neustálého zobrazení pokynů (naváděcí čára) a sledování displeje telefonu (nebo brýlí) na schopnost uživatele reagovat adekvátně na hrozbu srážky s pohybujícími se objekty (vozíky, lidé, aj.).

6. Závěr

V práci byl z počátku prozkoumán teoretický podklad, ve kterém bylo získáno povědomí o tom, co si lze pod pojmem rozšířená realita představit a jaké jsou její možné podoby. Na tyto příklady bylo navázáno se stručným popisem zařízení, na který lze prvky rozšířené reality uživateli zobrazit.

V následné kapitole byl uveden přehled v současnosti používaných technologií, které umožňují nalezení polohy zařízení uvnitř prostoru skladu. Správné určení polohy je totiž základním předpokladem pro správné fungování navigace.

Navazujíc na poznatky získané při zkoumání současného „state of the art" byl vytvořen návrh možné podoby aplikace, zvážena technologie a vybrána vhodná varianta, která reflektovala požadavky. Při návrhu aplikace byl vytvořen i vizuální návrh uživatelského prostředí s ovládacími prvky. Byla předběžně určena podoba navigačního AR prvku: navigační linie.

Po prvotním návrhu následovalo hledání teoretických znalostí, nyní se však pozornost soustředila především na softwarové prvky tvorby AR aplikace. Jak již bylo řečeno v úvodu, aplikace byla vyvíjena ve vývojářské aplikaci *Unity*, s rozšířením *AR Foundation*. Pro práci s tímto vývojovým nástrojem bylo nutné se nejdříve seznámit se základní metodikou tvorby aplikací (např. mobilních her) a tu následně rozšířit o přístupy k tvorbě AR aplikací (práce s prvky AR Camera aj.), k tomu sloužil převážně Manuál Unity a video kurzy.

Prvním krokem vývoje aplikace bylo vytvoření modelu prostředí, po zaměření a tvorbě výkresu byl vytvořen model v *Unity* jako základ pro následné generování plochy *NavMesh*. Plocha *NavMesh* tvoří v aplikaci dílčí roli, protože na jejím základu je generována křivka, která je následně vyrenderována do prostoru a zobrazuje se uživateli jako zmíněný prvek rozšířené reality ukazující cestu. Pro přesnější navádění v závěru trasy byla vytvořena šipka ukazující na cílovou pozici. Metoda, na které byla aplikace založena, a to jakým způsobem aplikace funguje, byl popsán v kapitolách Vývojové diagramy a C# zdrojové kódy.

Závěrem celé práce je zprovoznění vytvořené aplikace a její testování, které bylo zaznamenáno a vyhodnoceno. Měřena byla délka každé trasy, počet uzlů, odchylka polohy na konci měření a další. Pokud nastala v průběhu trasy neočekávaná událost, byla zaznamenána do komentáře, aby bylo následně možné vyvodit ze záznamu souvislosti.

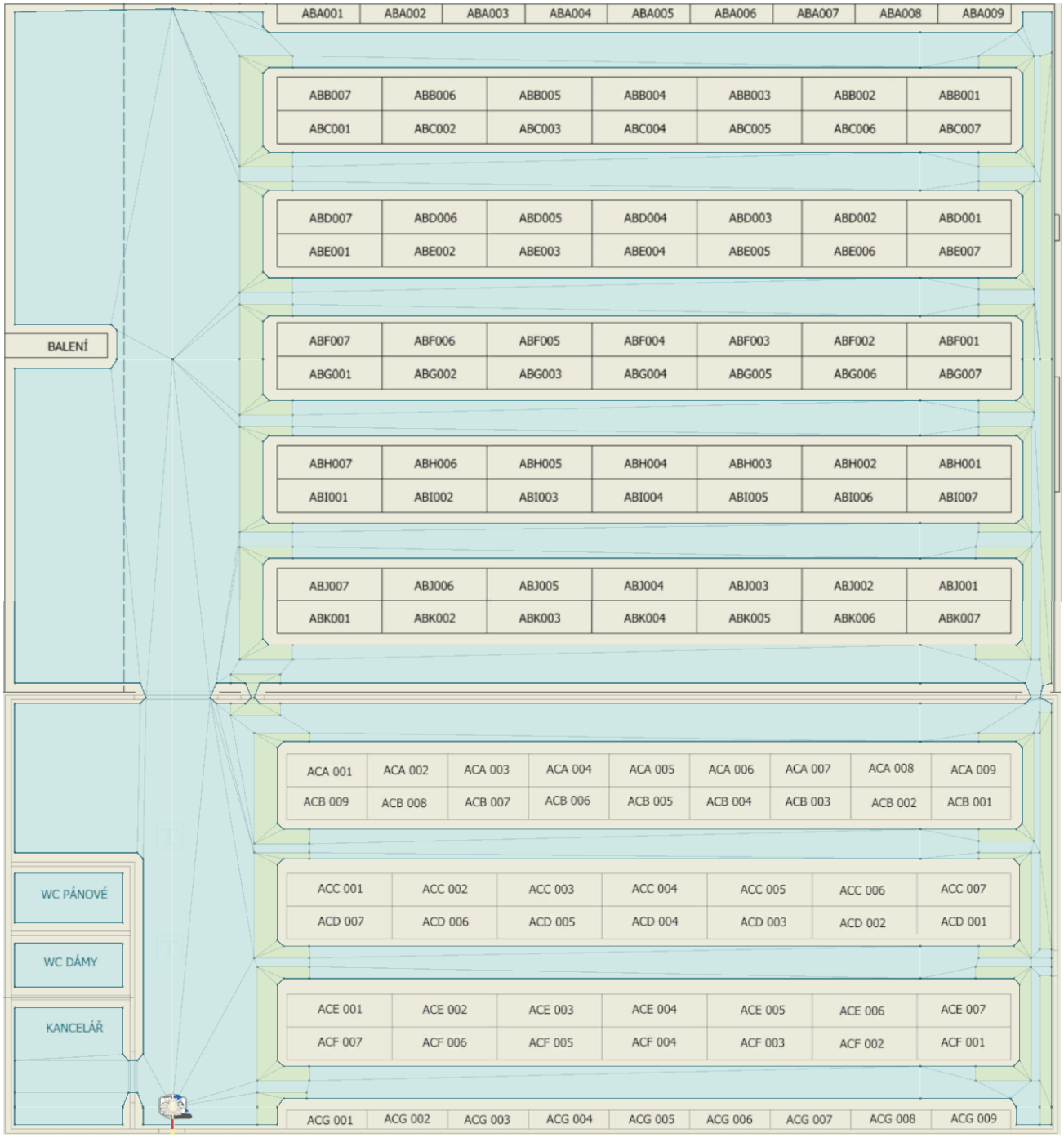
Výsledkem zkoumání navigačních schopností aplikace bylo, že podle posledního testování (Testování III. – finální testování) je úspěšnost dovedení do cíle 83 %. Procento úspěšnosti navigace je působivé, ale ve 26 % z těchto tras docházelo v průběhu navigace k výpadkům zobrazení linie. Výpadky byly často způsobeny citlivostí aplikace na světelné podmínky v prostředí skladu (protisvětlo nebo naopak tmavé užší místo), která vedla k mírné odchylce polohy a následnému dočasnému výpadku, viz kapitola 5.2.3. Z těchto nedokonalostí obrazového rozpoznávání okolí pomocí SDK ARCore plynula výsledná průměrná odchylka ve finálním testování 0,48 m.

V případě, že navigace dovedla uživatele do cíle, bylo ve většině tras snadné určit o jakou pozici se jedná, protože nepřesnost byla vůči velikosti regálové pozice (2,8 – 3,6 m) dostatečně malá. Přesto v některých případech byla odchylka tak velká, že bylo obtížné přiřadit na jakou pozici je uživatel naváděn.

Cílem práce bylo vytvořit aplikaci na které by se otestovala navržená metoda navádění uživatele v prostorách skladu pomocí rozšířené reality. Aplikace byla úspěšně vytvořena a pomocí testování bylo odhaleno úskalí metody, které je citlivost obrazového vnímání na světelné podmínky. Citlivost systému zapříčiňovala v určitých případech pouze zvýšenou nepřesnost navigace, jindy vedla k vážnější ztrátě polohy aplikace, což vedlo k nedovedení uživatele do cílové pozice. Řešení polohy aplikace je založené na balíčku ARCore, který je vyvíjen společností Google, balíček je pravidelně updatován a Google se jeho schopnosti snaží zdokonalovat novými funkcemi a algoritmy. Z toho pramení, že spolehlivost aplikace je z velké části závislá na verzi ARCore a použitém mobilním zařízení. Lze tedy očekávat, že s vývojem softwarového balíčku a zdokonalením mobilního telefonu bude možné dosáhnout lepších výsledků.

Na toto zjištění je navázáno v kapitole Zhodnocení aplikace, ve které bylo pojednáno nad dalšími možnostmi, které by mohly vylepšit nedostatky v současném řešení aplikace. Jednou ze zmiňovaných možností by mohlo být např. vyzkoušet tento konceptuální návrh na chytrých brýlích.

PŘÍLOHA č. 1 – Plocha NavMesh skladu ABx a ACx



PŘÍLOHA č. 2 – Tabulka měření spolehlivosti navigace I.

číslo	trasa	délka trasy [m]	počet uzlů	odchylka [m]	dovedl do cíle	druh chyby [kritické/nekritické]	komentář
1	ACA002 - ACF002	30	5	2,9	ANO	N	chvilkový výpadek linie
2	ABB005 - ACF002	57	7	0	ANO	-	-
3	ACF002 - ABF004	45	7	0	ANO	N	výpadek naváděcí linie
4	ABF004 - ABI005	30	7	x	NE	K	špatně provedená kalibrace
5	ABF002 - ABG004	23	6	0,3	ANO	N	zobrazení cíle v regálu
6	ABH006 - ABH003	6	2	0	ANO	-	-
7	ABG004 - ACF001	37	6	x	NE	K	ztráta polohy u stěny za regály
8	ACF001 - ACB007	39	6	0	ANO	-	-
9	ACB007 - ABB007	40,6	6	0	NE	K	navigování do regálu
10	ABD001 - ACG001	40,5	6	x	NE	K	navedl na špatnou pozici
11	ACA003 - ABA009	42	6	0,5	ANO	-	-
12	ABA009 - ABI001	42,5	6	0,5	ANO	-	-
13	ABI001 - ABH003	15,6	6	x	NE	K	chyba při kalibraci
14	ACA002 - ACF001	34,5	6	0	ANO	-	-
15	ACF001 - ACC006	16,3	6	0,4	ANO	-	-
16	ACC006 - ACA004	25,8	5	1,5	ANO	N	chybná rotace zařízení
17	ACA003 - ACA002	3	2	0	ANO	-	-
18	ACA003 - ACF001	32	5	0,4	ANO	N	výpadky u stěny za regály
19	ABI007 - ABI005	7	2	0,3	ANO	-	-
20	ABI005 - ABC001	32	6	5	NE	K	dezorientace zařízení
21	ABC001 - ABH003	26,6	5	x	NE	K	navigace do regálu
22	ABC001 - ABB006	14	6	0	ANO	-	-
23	ABB006 - ABI002	29	6	0,5	ANO	-	-
24	ABI006 - ABF005	30	7	0,5	ANO	-	-
25	ABF005 - ABB007	21,6	5	1	ANO	-	-
26	ABI006 - ACF001	26,5	6	x	NE	K	chyba kalibrace
27	ABH003 - ABI001	15,6	6	x	NE	K	chybná rotace zařízení
28	ABG004 - ABF002	23	6	0,3	ANO	N	výpadky navigace u cíle
29	ACF002 - ABB005	57	7	0,3	ANO	-	-
30	ABF005 - ABI006	30	7	0,5	ANO	-	-

PŘÍLOHA č. 3 – Tabulka měření spolehlivosti navigace II.

číslo	trasa	délka trasy [m]	počet rohů	odchylka [m]	kalibrace na začátku	dovedl do cíle	druh chyby [kritické/nekritické]	komentář
1	ACG002- ACB007	21	5	0	ne	ano	-	
2	ACB007 - ACA003	19,5	6	0,3	ne	ano	N	průběžný výpadek
3	ACA003 - ABG004	32,5	7	0,5	ne	ano	N	navigování do regálu
4	ABG004 - ACD003	39	7	0	ano	ano	-	
5	ACD003 - ACB008	26	4	0,4	ne	ano	N	ztráta směru
6	ACB008 - ACG003	21	5	0	ne	ano	-	
7	ACG003 - ACG002	2	2	0	ano	ano	N	průběžné zmizení linie
8	ACG002 - ABA007	57	5	3	ne	ano	N	výpadky cestou
9	ABA007 - ABC001	28	5	0,3	ano	ano	-	
10	ABC001 - ACB008	32	6	0,5	ne	ano	N	navigace do regálu
11	ACB008 - ABG002	25	5	0	ano	ano	-	
12	ABG002 - ACF002	49	7	x	ne	ne	K	výpadek
13	ABI007 - ACE002	36	7	x	ne	ne	K	špatná kalibrace
14	ACE002 - ABG004	38	5	0	ne	ano	N	negeneruje navigaci
15	ABG004 - ABJ002	20	4	0	ano	ano	N	
16	ABJ002 - ABI002	10	2	0	ano	ano	-	
17	ABI002 - ABA007	39	6	0,3	ne	ano	N	výpadek, po chvíli ok
18	ABA007 - ACA004	42	6	0,3	ne	ano	N	navedl do ACA002, pak správně ACA004
19	ACA004 - ABC001	33	7	0	ne	ano	-	
20	ABC001 - ACA004	33	6	0	ne	ano	-	
21	ACE002 - ACB007	18	4	0,5	ano	ano	-	
22	ACB007 - ACB008	2	2	1	ne	ano	-	
23	ACB008 - ABG002	24	6	1	ne	ano	-	
24	ABG002 - ABB007	18	5	1	ano	ano	N	ztráta orientace cestou, pak ok
25	ABB007 - ABB005	7	2	0	ano	ano	-	
26	ABB005 - ABG003	31	6	0	ne	ano	-	
27	ABG003 - ACF002	45	7	x	ne	ne	K	navigace do regálu, ztráta orientace
28	ACE002 - ABI007	36	7	0,4	ano	ano	N	výpadek cestou
29	ABI007 - ABH006	26	6	0	ne	ano	-	
30	ABH006 - ABI007	26	5	0	ne	ano	-	
32	ABI007 - ABJ006			x	ne	ne	K	negeneruje navigaci
33	ABI007 - ABF005	26	6	x	ne	ne	K	ztráta orientace

34	ACE002 - ABA009	35	7	0	ano	ano	-	průběžná ztráta orientace
35	ABA009 - ABA007				ne	ne	K	ztráta orientace
36	ABA007 - ACE002	53	5	x	ne	ne	K	výpadek, ztráta orientace
37	ACE002 - ABI001	23	5	0	ano	ano	-	
38	ABI001 - ABH06	12	5	0,5	ne	ano	-	
39	ABH06 - ACB008	23	6	0,5	ne	ano	N	navigace do regálu
40	ACB008 - ABF004	37	6	x	ano	ne	K	ztráta a nalezení
41	ABF004 - ABA007	30	4	1	ano	ano	N	výpadek cestou
42	ABA007 - ABI007	20	6	x	ano	ne	K	spadla aplikace
43	ABI007 - ABJ002	24	5	0,5	ano	ano	-	
44	ABJ002 - ABG004	20	4	0,5	ne	ano	-	
45	ABG004 - ABB006	33	6	0	ano	ano	-	
46	ABB006 - ACC005	52	5	6	ne	ano	-	
47	ACC005 - ABB005	52	5	0,3	ano	ano	N	cestou pozice v regálu
48	ABB005 - ACG002	49	6	0	ne	ano	-	
49	ACG002 - ABJ002	44	6	0,5	ne	ano	N	průběžná ztráta orientace
50	ABJ002 - ABI007	4	2	0,5	ne	ano	-	
51	ABI007 - ABG004	32	5	2	ano	ano	N	ztráta – protisvětlo
52	ABG004 - ACD003	37	7	x	ne	ne	K	ztráta
53	ABG004 - ABA008	30	5	0	ne	ano	-	
54	ABA008 - ABH006	36	4	0,4	ano	ano	N	výpadek
55	ABH006 - ABH004	6	2	0	ne	ano	-	
56	ABH004 - ACC005	34	6	0	ano	ano	-	
57	ACC005 - ABF004	30	7	0,5	ne	ano	-	
58	ABF004 - ACC005	40	5	0	ano	ano	-	
59	ACC005 - ACA002	26	5	0,5	ne	ano	-	
60	ACA002 - ABH006	21	7	0	ne	ano	-	
61	ABH006 - ABD001	31	5	1	ne	ano	N	průběžný výpadek
62	ABD001 - ABI005	23	7	0,5	ano	ano	-	
63	ABI005 - ABG003	27	4	0	ne	ano	-	
64	ABG003 - ACG003	41	6	0,5	ne	ano	-	
65	ACG003 - ABG002	36	5	0,4	ano	ano	-	
66	ABG002 - ABH004	6	2	0	ne	ano	-	
67	ABH004 - ACE002	39	5	0,6	ne	ano	-	
68	ACE002 - ACC005	27	5	0	ano	ano	-	
69	ACC005 - ABA008	41	5	2	ne	ano	N	průběžný výpadek
70	ABA008 - ACG001	46	5	x	ano	ne	-	
71	ACG001 - ACF002	25	6	0,3	ano	ano	-	

Bibliografie

[1] AZUMA, R., Y. BAILLOT, R. BEHRINGER, S. FEINER, S. JULIER a B. MACINTYRE. Recent advances in augmented reality. *IEEE Computer Graphics and Applications* [online]. 2001, **21**(6), 34–47. ISSN 1558-1756. Dostupné z: doi:10.1109/38.963459

[2] SCHMALSTIEG, D. a Tobias HÖLLERER. *Augmented reality: principles and practice*. Boston: Addison-Wesley, 2016. ISBN 978-0-13-315320-0.

[3] AZUMA, Ronald T. A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments* [online]. 1997, **6**(4), 355–385. Dostupné z: doi:10.1162/pres.1997.6.4.355

[4] SUTHERLAND, Ivan E. The Ultimate Display. In: *Proceedings of the Congress of the Internation Federation of Information Processing (IFIP)* [online]. 1965, s. 506–508. Dostupné z: http://www.wired.com/beyond_the_beyond/2009/09/augmented-reality-the-ultimate-display-by-ivan-sutherland-1965/

[5] STAFF, V. R. The Sword of Damocles (1968) - The Complete History of VR. *The VR Shop* [online]. 21. prosinec 2021 [vid. 2022-12-02]. Dostupné z: https://www.virtual-reality-shop.co.uk/the-sword-of-damocles-1968/

[6] FEINER, Steven, Blair MACINTYRE a Dorée SELIGMANN. Knowledge-based augmented reality. *Communications of the ACM* [online]. 1993, **36**(7), 53–62. ISSN 0001-0782. Dostupné z: doi:10.1145/159544.159587

[7] GRUBERT, Jens a Raphaël GRASSET. *Augmented reality for randroid application development: learn how to develop advanced augmented reality applications for android*. Birmingham, UK: Packt Publishing, 2013. ISBN 978-1-78216-856-0.

[8] ARCore supported devices. *Google Developers* [online]. [vid. 2022-11-26]. Dostupné z: https://developers.google.com/ar/devices

[9] Rozšířená realita. *Apple (Česká republika)* [online]. [vid. 2022-11-26]. Dostupné z: https://www.apple.com/cz/augmented-reality/

[10] You'll soon be able to use Pokémon Go's tech to make your own AR games and experiences. *Digital Arts* [online]. [vid. 2022-11-26]. Dostupné z: https://www.digitalartsonline.co.uk/news/hacking-maker/youll-soon-be-able-use-pokemon-gos-tech-make-your-own-ar-games-experiences/

[11] Buy HoloLens 2: Find Specs, Features, Capabilities & More. *Microsoft Store* [online]. [vid. 2022-11-26]. Dostupné z: https://www.microsoft.com/en-us/d/hololens-2/91pnzzznzwcp

[12] MIHAJLOW, Radko a Vladimir DEMIREV. APPLICATION OF GPS NAVIGATION IN AGRICULTURAL AGGREGATES. *ANNUAL JOURNAL OF TECHNICAL UNIVERSITY OF VARNA, BULGARIA* [online]. 2018, **2**, 14–19. Dostupné z: doi:10.29114/ajtuv.vol2.iss2.84

[13] *Application of GPS-RTK Technology in the Land Change Survey | Elsevier Enhanced Reader* [online]. [vid. 2022-11-27]. Dostupné z: doi:10.1016/j.proeng.2012.01.511

[14] KREJČÍ, Tomáš. *Analýza jízdy vozidla řízeného Trimble AutoPilot* [online]. Pardubice, 2014 [vid. 2022-11-27]. Univerzita Pardubice Fakulta elektrotechniky a informatiky. Dostupné z: https://dk.upce.cz/bitstream/handle/10195/56442/KrejciT_AnalyzaJizdy_ZN_2014.pdf?sequence=3&isAllowed=y

[15] What is RFID and how does it work? *IoT Agenda* [online]. [vid. 2022-11-27]. Dostupné z: https://www.techtarget.com/iotagenda/definition/RFID-radio-frequency-identification

[16] KUNHOTH, Jayakanth, AbdelGhani KARKAR, Somaya AL-MAADEED a Abdulla AL-ALI. Indoor positioning and wayfinding systems: a survey. *Human-centric Computing and Information Sciences* [online]. 2020, **10**(1), 18. ISSN 2192-1962. Dostupné z: doi:10.1186/s13673-020-00222-0

[17] ESCOBAR, Tech Talker Eric. How Does Wi-Fi Work? *Scientific American* [online]. [vid. 2022-11-29]. Dostupné z: https://www.scientificamerican.com/article/how-does-wi-fi-work/

[18] Bluetooth Technology Overview. *Bluetooth® Technology Website* [online]. [vid. 2022-11-30]. Dostupné z: https://www.bluetooth.com/learn-about-bluetooth/tech-overview/

[19] Table 2 -1. Bluetooth Device Classes of Power Management. *ResearchGate* [online]. [vid. 2022-11-29]. Dostupné z: https://www.researchgate.net/figure/1-Bluetooth-Device-Classes-of-Power-Management_tbl1_329973291

[20] Everything you need to know about UWB technology. *KINEXON* [online]. [vid. 2022-11-30]. Dostupné z: https://kinexon.com/uwb-technology/

[21] *Area Targets | VuforiaLibrary* [online]. [vid. 2023-02-22]. Dostupné z: https://library.vuforia.com/environments/area-targets

[22] *AR Foundation | AR Foundation | 5.0.3* [online]. [vid. 2023-02-22]. Dostupné z: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@5.0/manual/index.html

[23] UMEK, Anton a Anton KOS. Validation of smartphone gyroscopes for mobile biofeedback applications. *Personal and Ubiquitous Computing* [online]. 2016, **20**. Dostupné z: doi:10.1007/s00779-016-0946-4

[24] DEEMS, Jeffrey S., Thomas H. PAINTER a David C. FINNEGAN. Lidar measurement of snow depth: a review. *Journal of Glaciology* [online]. 2013, **59**(215), 467–479. ISSN 0022-1430, 1727-5652. Dostupné z: doi:10.3189/2013JoG12J154

[25] PUBLISHED, Ryan Goodrich. Accelerometers: What They Are & How They Work. *livescience.com* [online]. 1. říjen 2013 [vid. 2023-02-23]. Dostupné z: https://www.livescience.com/40102-accelerometers.html

[26] TECHNOLOGIES, Unity. *Unity - Manual: Unity User Manual 2021.3 (LTS)* [online]. [vid. 2023-03-27]. Dostupné z: https://docs.unity3d.com/Manual/index.html

[27] LANHAM, Micheal. *Learn ARCore - Fundamentals of Google ARCore: Learn to build augmented reality apps for Android, Unity, and the web with Google ARCore 1.0*. Birmingham: Packt Publishing, 2018. ISBN 978-1-78883-363-9.

[28] What's under the hood in ARCore. *Google Developers* [online]. [vid. 2023-03-27]. Dostupné z: https://developers.google.com/ar/develop/video

[29] 04 - How does Situm work? *Situm* [online]. [vid. 2022-12-01]. Dostupné z: https://situm.com/docs/how-does-situm-work/

[30] KOSTAK, Jan. Indoor Navigation Platform Applications. *Sewio RTLS* [online]. [vid. 2022-12-01]. Dostupné z: https://www.sewio.net/indoor-navigation/

[31] XRGO AR Navigation - Augmented Reality Indoor / Outdoor Navigation. *XRGO - We connect the industry with X-Reality (AR, MR, VR)* [online]. [vid. 2022-12-02]. Dostupné z: https://xrgo.io/en/product/xrgo-ar-navigation-augmented-reality-indoor-outdoor-navigation/

[32] Indoor-Navigation: Diese Technologien könnten für einen Durchbruch sorgen. *Fraunhofer IAO – BLOG* [online]. 12. listopad 2020 [vid. 2022-12-02]. Dostupné z: https://blog.iao.fraunhofer.de/indoor-navigation-diese-technologien-koennten-fuer-einen-durchbruch-sorgen/

[33] *QR Code Generator | Create QR Code for free* [online]. [vid. 2023-04-10]. Dostupné z: https://qrplanet.com/

[34] KALLE (JAG). *Unity Converters for Newtonsoft.Json* [online]. C#. 4. duben 2023 [vid. 2023-04-08]. Dostupné z: https://github.com/jilleJr/Newtonsoft.Json-for-Unity.Converters

[35] *ZXing Decoder Online* [online]. [vid. 2023-04-08]. Dostupné z: https://zxing.org/w/decode.jspx

[36] FIREDRAGONGAMESTUDIO. *Barcode and QR Code handling in Unity with ZXing.NET* [online]. ShaderLab. 29. leden 2023 [vid. 2023-03-28]. Dostupné z: https://github.com/FireDragonGameStudio/Unity-ZXing-BarQrCodeHandling/blob/23366b4544c12038212a7a7829da5d51f954628e/Assets/Scripts/Android/AndroidCodeReaderSample.cs