VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

APLIKACE PRO EFEKTIVNÍ ŘÍZENÍ DRONU S VYUŽITÍM ROZŠÍŘENÉ VIRTUALITY

APPLICATION FOR EFFICIENT DRONE CONTROL USING AUGMENTED VIRTUALITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **RÓBERT HUBINÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÍTĚZSLAV BERAN, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Hubinák Róbert**
Program: Informační technologie
Název: **Aplikace pro efektivní řízení dronu s využitím rozšířené virtuality**
Application for Efficient Drone Control Using Augmented Virtuality
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s aktuálními technologiemi pro řízení dronů a vizualizací letových informací. Prostudujte využívané navigační prvky podobných aplikací.
2. Navrhněte vizualizační aplikaci a sadu grafických prvků, které budou zobrazovat prostorové informace naplánované mise, letové informace, bezpečnostní prvky apod.
3. Implementujte navrženou aplikaci s využitím relevantních dostupných technologií a knihoven.
4. Vyhodnoťte vlastnosti výsledného řešení na základě experimentů v reálném prostředí.
5. Prezentujte klíčové vlastnosti řešení formou plakátu a krátkého videa.

Literatura:

- Dieter Schmalstieg, Tobias Hollerer. *Augmented Reality: Principles and Practice.* Addison-Wesley, 2016. ISBN: 978-0321883575.
- H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Boston, 2005.
- Dále dle pokynu vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a částečně bod 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Beran Vítězslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 1. listopadu 2019

Abstrakt

Cieľom tejto práce je identifikovať problematické situácie, ktoré môžu nastať pri pilotovaní dronu a na ich základe navrhnúť prvky užívateľského rozhrania, ktoré by tieto problémy eliminovali. Navrhnuté prvky boli implementované do už existujúcej aplikácie na ovládanie dronu. Výsledné riešenie využíva technológiu rozšírenej virtuality a rozšírenej virtuality, kedy sú podporne vizualizačné prvky vkladané priamo do 3D scény a reálnych dát z dronu. V mojej práci som vytvoril systém, ktorý umožňuje pilotovi definovať záchytné body v scéne formou misie, rozširuje aplikáciu o navigačný systém k týmto bodom a pridáva do aplikácie prvky, ktoré pilotovi ukazujú smer a vzdialenosť blížiacich sa prekážok.

Abstract

The aim of this work is to identify problematic situations that may occur when piloting a drone and based on them design elements of the user interface that would eliminate these problems. The proposed elements were implemented in an existing drone control application. The resulting solution uses the technology of augmented virtuality and augmented reality, where supporting visualization elements are inserted directly into the 3D scene and real data from the drone. In my work, I created a system that allows the pilot to define clues in the scene in the form of a mission, extends the application with a navigation system to these points and adds to the application elements that show the pilot the direction and distance of approaching obstacles.

Klíčová slova

dron, kooperácia dronov, ovládanie dronov, rozšírená virtualita, virtuálna scéna, misia, navigačné prvky, mapa, zobrazenie nebezpečenstva

Keywords

drone, drone cooperation, drone control, augmented virtuality, virtual scene, mission, navigation elements, map, danger display

Citace

HUBINÁK, Róbert. *Aplikace pro efektivní řízení dronu s využitím rozšířené virtuality.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, Ph.D.

Aplikace pro efektivní řízení dronu s využitím rozšířené virtuality

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Róbert Hubinák
30. července 2020

Poděkování

Chcel by som sa poďakovať vedúcemu práce doktorovi Beranovi za cenné rady na spoločných konzultáciach a pomoc pri dokončovaní práce. Zároveň sa chcem poďakovať výskumníkovi Alfredovi za dodanie testovacích dát z dronu. Na záver by som sa ešte chce poďakovať mojim kamarátom, spolužiakom a rodine, ktorý ma podporovali pri písaní práce.

Obsah

Kapitola 1

Úvod

Táto práca sa zaoberá návrhom navigačných prvkov do už existujúcej aplikácie, ktorá využíva technológiu rozšírenej virtuality pre uľahčenie pilotáže dronu. Hlavným cieľom tejto práce je implementovať prvky, ktoré by pilotovi uľahčili orientáciu v 3D scéne. 3D scéna tejto aplikácie je totižto pomerne ťažko čitateľná a pilot, ktorý nepozná dobre okolie, v ktorom s dronom lieta sa pomerne ľahko stratí prehľad o tom, kde sa jeho dron nachádza a o tom, čo sa nachádza okolo dronu.

V mojej práci som definoval problematické situácie, kedy by pilot vyžadoval dodatočné informácie, ktoré táto aplikácia neponúka. Sú to informácie ako vzdialenosť a smer blížiacich sa prekážok, ale aj priestorové informácie o tom, kde sa nachádzajú prvky misie, ktorú pilot práve plní. Výsledné riešenie obsahuje sadu prvkov užívateľského rozhrania, ktoré umožňujú vizualizáciu týchto dôležitých informácii.

V prvej kapitole stručne popíšem základnú teóriu týkajúcu sa dronov, popíšem ako drony fungujú, ako sa ovládajú a aké úskalia prináša pilotáž týchto zariadení. V závere kapitoly sa v krátkosti povenujem opisu aktuálne používaných aplikácií na ovládanie dronu.

Nasledujúca kapitola sa venuje architektúre aplikácie, na ktorej budem pracovať, popíšem v nej jej najdôležitejšie súčasti a poukážem na nedostatky, na ktorých by som chcel v mojej práci zapracovať a plynule prejdem k návrhu riešenia.

V návrhu uvediem akými prvkami by sa dali jednotlivé problémy riešiť a rozoberiem, akým spôsobom by sa dali tieto prvky implementovať a vizualizovať do aplikácie. Menšiu podkapitolu potom venujem revízií užívateľského rozhrania na dotykové zariadenia.

V poslednej časti práce opíšem implementáciu navrhnutých prvkov, popíšem najdôležitejšie súčasti a niektoré úskalia s nimi spojené. Prácu zakončím popisom overovania funkčnosti aplikácie.

Kapitola 2

Ovládanie dronu

2.1 Dron

Bezpilotné lietajúce zariadenie, bežne nazývané ako dron, je akékoľvek zariadenie prispôsobené na let bez prítomnosti posádky. Bezpilotné lietadlá sú súčasťou takzvaných bezpilotných leteckých systémov, ktoré zahrňujú samotné lietadlo, stanicu pre ovládanie lietadla, systém komunikácie medzi stanicou a lietadlom a ďalšie komponenty potrebné pre let. Dron môže byť ovládaný buď pilotom na diaľku, alebo autonómne za pomoci autopilota. Moderné drony využívajú kombináciu týchto dvoch prístupov, kedy je ovládanie ponechané primárne pilotovi, avšak v prípade potreby môže dron prejsť do autonómneho módu [6]. Podľa konštrukcie drony môžme rozdeliť na 2 základné skupiny:

Drony s pevnou nosnou plochou

Tieto drony sú konštrukčne podobné bežným lietadlám (obrázok 2.1). Na let teda využívajú namiesto vrtulí jedno, alebo viac krídiel. Hlavnou výhodou takejto konštrukcie je, že dron využíva efektívne vztlak vzduchu, vďaka čomu sa výrazne šetrí využitie batérie. Takéto drony teda vydržia vo vzduchu dlhšie a unesú väčšiu záťaž ako drony s rotujúcou nosnou plochou. Ďalšou výhodou tejto konštrukcie je schopnosť plachtenia. To môže byť výhodné v prípade, že dronu zlyhá motor, pretože pilot môže stále drona čiastočne ovládať a vďaka tomu núdzovo pristáť. Nevýhodou oproti bežným vrtuľovým dronom je fakt, že udržať takýto dron na jednom mieste je prakticky takmer nemožné.

Drony s rotujúcou nosnou plochou

Tieto drony sú konštrukčne podobné helikoptéram (obrázok 2.2). Oproti helikoptéram však majú tradične viacej rotorov a dynamika letu je odlišná. Na náklon dronu sa nevyužíva náklon listov na rotoroch, ako je tomu v prípade helikoptér, ale regulácia otáčok rotorov. Počet rotorov sa u týchto dronov môže výrazne líšiť. Najtypickejšie sú drony so štyrmi rotormi umiestnenými v tvare písmena X, nazývané quadrokoptéry. Existujú však aj trikoptéry, hexakoptéry a oktokoptéry. Špeciálnou skupinou sú koptéry s koaxiálnou konštrukciou. Táto konštrukcia využíva 2 protibežné rotory nad sebou na jednom rameni. Typicky platí, že čím má dron viacej rotorov, tým je stabilnejší. Hexakoptéry a octokoptéry sa teda využívajú primárne na filmárske účely, pretože sú stabilnejšie a zároveň unesú aj ťažšie vybavenie, ako napríklad profesionálne filmárske kamery. Na druhú stranu však platí, že čím menej rotorov dron má, tým je vo vzduchu obratnejší [9].

Obrázek 2.1: Dron s pevnou nosnou plochov MQ – Reaper využívaný U.S. Air Force.[1]

Obrázek 2.2: Quadrokoptéra Dron DJI Mavic 2 Pro.[2]

2.2 Využitie dronov

Prvým odvetvím ktoré využívalo drony vo väčšej miere bola armáda, ktorá tieto zariadenia využívala na prieskum bojiska a na bombardovanie. Postupom času sa použitie dronov rozšírilo do ďalších odvetví a v dnešnej dobe je práca s nimi bežná a stávajú sa čoraz. Najznámejším odvetvím komerčnej sféry, ktoré využíva drony je filmársky priemysel, kde sa využívajú najmä na natáčanie náročnejších vzdušných záberov (obrázok 2.3), alebo aj na tvorbu obsahu do virtuálnej reality Využitie dronov je však omnoho širšie, ako sa na

[1] Zdroj obr.: https://www.stripes.com/news/promotion-rates-improving-for-air-force-drone-pilots-gao-says-1.567839

[2] Zdroj obr.: https://www.dji.com/sk/products

prvý pohľad môže zdať. Dôležitú rolu hrajú napríklad v oblasti poľnohospodárstva, kde sú okrem iného využívané na sledovanie úrody v reálnom čase. Poľnohospodári tak majú rýchly prístup k dátam o úrode a v prípade potreby môžu okamžite zakročiť. Vo vyspelejších krajinách sa využívajú autonómne drony pre chemický postrek polí. V oblasti lesníctva sa drony využívajú na monitorovanie lesných požiarov. Drony využívajú taktiež záchranné zložky na hľadanie stratených osôb, alebo získavanie dát o ťažko dostupných lokáciach v prípade prírodných katastrof [8].

Obrázek 2.3: Záber z filmu The Circle, ktorý bol celý natočený dronom DJI Inspire 2. [3]

2.3 Autonómnosť dronu

Ako bolo spomenuté v kapitole 2.1, ovládanie dronu môže byť či už čiastočne, alebo aj úplne autonómne. Ovládanie dronu je teda odlišné pre rôzne stupne autonómnosti. V tomto ohľade rozlišujeme 4 rôzne spôsoby ovládania letu:

Direct position control – Tradičné ovládanie dronu pomocou joystick-u. Dron tak nepotrebuje žiadne dodatočné senzory a pilot ovláda drona spôsobmi, ktoré sú opísané v kapitole 2.4.

Absolute position control – S využitím pokročilejších systémov, ako je napríklad GPS môže pilot nastaviť požadovanú cielovú destináciu a dron dokáže sám zmeniť polohu bez nutnosti zásahu pilota.

Relative position control – Pokročilejšie monitorovacie systémy dokážu udržať drona v relatívnej vzdialenosti od určitého objektu. Pilot tak môže napríklad zadať príkaz kedy bude dron pilota nasledovať v určitej vzdialenosti.

Task-oriented control – Najpokročilejším spôsobom ovládania dronu je ovládanie pomocou úloh. Príkladom môže byť tohtoročná oslava nového roku v Shaghai, kde sa rozhodli namiesto tradičného ohňostroja využiť drony na svetelnú šou [7].

Na to, aby sa dron dokázal autonómne pohybovať, musí mať zabudované špeciálne senzory, ktoré snímajú jeho okolie. Typicky sa využívajú optické senzory umiestnené v prednej

[3] Zdroj obr.: https://nofilmschool.com/2017/07/watch-first-dji-inspire-2-short-film-here

časti dronu, ktoré dokážu s pomocou špeciálnych algoritmov vytvoriť 3D hĺbkovú mapu priestoru pred dronom v reálnom čase. Spolu s optickými senzormi sa využívajú aj ultrasonické senzory, ktoré slúžia na udržanie dronu v stálej výške nad zemou.

2.4 Spôsoby riadenia dronu

Pri riadení dronu pilotom sa využívajú 2 základné prístupy. Riadenie pohľadom zo zeme a riadenie pomocou kamery.

Riadenie pohľadom zo zeme

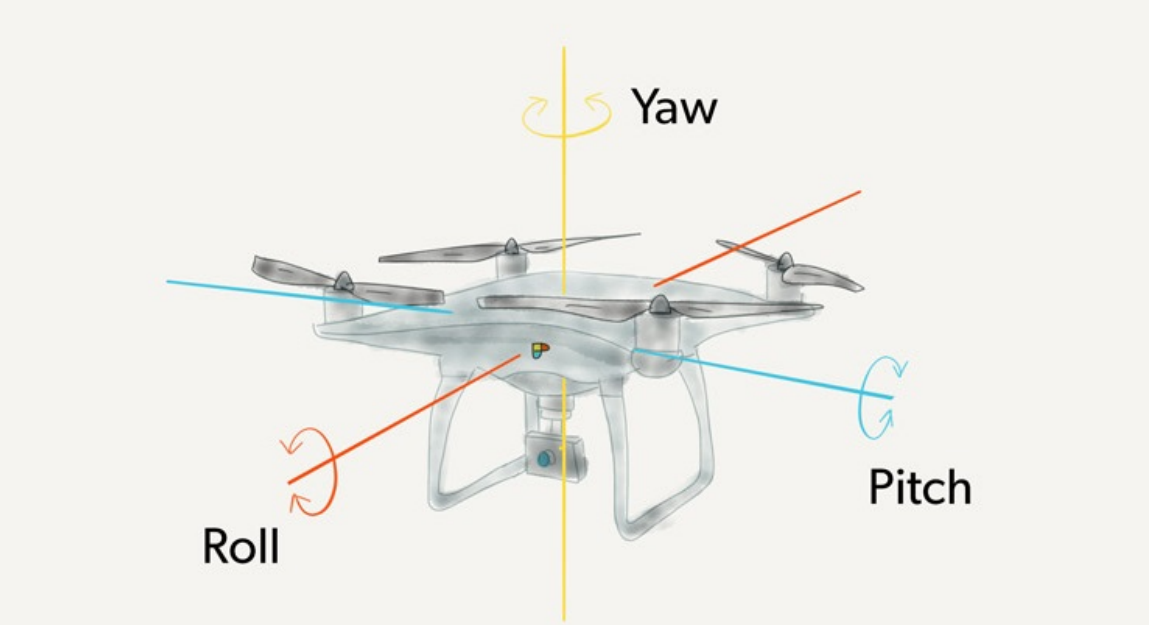
Pri tomto prístupe môže pilot ovládať zariadenie len pokiaľ je dron v jeho zornom poli. Pilot pri tomto prístupe jesne vidí dron a smer letu, vďaka čomu je možný aj bezpečný let pomedzi prekážky. Značnou nevýhodou je obmedzená oblasť, ktorú môže pilot dronom pokryť. Ďalším bežným problémom je strata orientácie. Bežne dostupné komerčné drony sú prevažne kvadrokoptéry a v prípade, že je dron ďalej od pilota je náročné určiť kam smeruje predok dronu. Pilot tak môže nechcene stočiť dronu do nesprávnej strany a naraziť na prekážku.

Riadenie pomocou kamery

Pri riadení pomocou kamery využíva pilot na orientáciu pohľad cez kameru, zabudovanú vo väčšine prípadov v prednej časti dronu. Maximálna vzdialenosť letu teda nie je obmedzená zorným polom pilota, ale iba dosahom signálu. Problém straty orientácie je teda eliminovaný nakoľko pilot presne vidí kam dron smeruje. Ak však pilot nemá dron v zornom poli, je vyhýbanie sa prekážkam omnoho náročnejšie, pretože pohľad z prvej osoby je značne obmedzujúci. Neskúsený pilot tak ťažko odhadne čo sa v okolí dronu nachádza a hrozí že v nepozornosti narazí na prekážku. Typickým príkladom môže byť prelet popod most, kedy pilot na kamere vidí, že dron preletel popod most, preto sa rozhodne stúpať, no zadná časť dronu ešte nie je za hranicou mosta a takto dôjde k havárií. Takisto môže nastať situácia, kedy na drobnú chvíľu vypadne kamera, čo môže mať pre dron fatálne následky. Problém so stratou orientácie môže čiastočne riešiť použitie 360 stupňovej kamery. Toto riešenie však nie je ideálne, pretože pilot stále nevidí čo sa nachádza pod a nad dronom a zároveň použitie takejto kamery zvyšuje nárok na prenos dát. Česká legislatíva bohužiaľ tento spôsob bežnému pilotovi nepovoľuje [11]. Je však vhodné sa tomuto spôsobu venovať s ohľadom na špecifické aplikácie, napríklad u záchranárov, alebo polície.

Základným spôsobom ovládania bežného dronu je ovládanie pomocou RC vysielača. Ovládať dron je možné náklonom v 3 smeroch (obrázok 2.4):

- **Náklon dopredu a dozadu** (pitch)
- **Bočný náklon** (roll)
- **Rotácia po vertikálnej osi** (yaw)



Obrázek 2.4: Osy náklonu drona. [4]

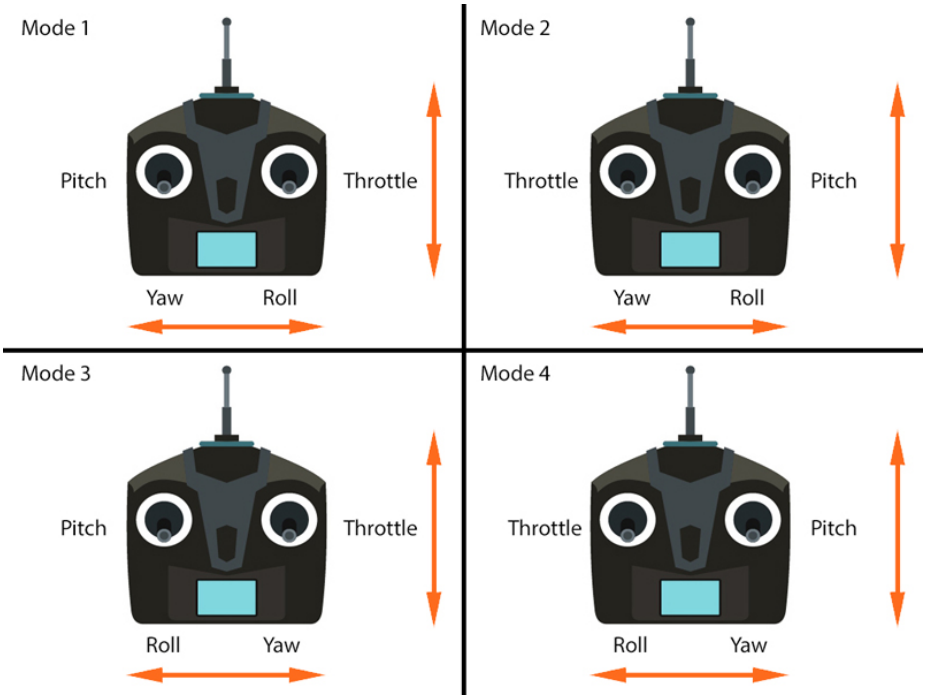
O tieto ovládacie prvky sa starajú 2 joysticky zabudované na vysielači ktorými je možné pohybovať po vertikálnej aj horizontálnej ose. Každý z joystickov ovláda pohyb dronu, buď náklonom v 2 z týchto smerov, alebo náklonom do jedného smeru a regulovaním plynu (stúpanie a klesanie dronu). Ak pilot pri ovládaní uvolní joystick, typicky sa joystick vráti do základnej pozície čím sa dron stabilizuje. Výnimkou je ovládanie plynu. To, za ktorý pohyb sú jednotlivé joysticky zodpovedné závisí od módu vysielača. Existujú 4 módy vysielača, pričom v drvivej väčšine sa môžeme stretnúť s módami 1 a 2, módy 3 a 4 sú využívané minimálne (obrázok 2.5). Tieto módy sú:

- **Mode 1**
  V tomto móde ovláda ľavý joystick predný náklon a rotáciu okolo vertikálnej osi. Pravý jotstick ovláda plyn a bočný náklon. Výhodou tohto módu je rozdelenie predného a bočného náklonu, čo umožňuje pilotovi presnejšie ovládanie dronu. Tento mód bol v minulosti bežnejší, postupom času však začal prevládať mód 1.

- **Mode 2**
  V tomto móde je ovládanie predného a bočného náklonu na pravom joysticku a ovládanie plynu a rotácie okolo vertikálnej osi na pravom. Nakoľko sú predný a bočný náklon hlavnými faktormi ovplyvňujúcimi smer letu, je výhodné mať tieto ovládacie prvky na jednom joysticku. Na tomto princípe funguje napríklad aj ovládanie skutočných lietadiel.

- **Mode 3 a 4**
  Mód 3 je opakom módu 2, čiže ovládanie pravého joysticku je presunuté na ľavý a naopak. Mód 4 je opakom módu 1. Tieto módy sú tu uvedené len pre úplnosť.

[4] Zdroj obr.: https://www.photopills.com/articles/drone-photography-guide/



Obrázek 2.5: Porovnanie jednotliých ovládacích módov.[5]

Niektoré pokročilejšie vysielače umožňujú medzi týmito módami ľubovoľne prepínať. Súčasťou vysielača môže byť aj display zobrazujúci dôležité informácie, ako kapacita batérie, stav signálu, výška a tak ďalej [4].

2.5 Aplikácie na ovládanie dronu

Moderné drony umožňujú sledovanie údajov o stave dronu spoločne so sledovaním pohľadu kamery dronu pomocou aplikácie v smartfóne, ktorý je možné upevniť priamo na vysielač. V minulosti tieto aplikácie slúžili primárne na nahrávanie videa a sledovanie kľúčových údajov z dronu akými sú signál, alebo percentá batérie. Postupom času sa s posunom technológie posúval aj ich vývoj a v dnešnej dobe sú nabité množstvom užitočných funkcií, ktoré výrazne uľahčujú let.

DJI GO 4

Najznámejšou aplikáciou, ktorá toto v súčasnej dobe umožňuje je aplikácia pre drony spoločnosti DJI, DJI GO 4 (obrázok 2.6).DJI GO 4 má veľké množstvo funkcií, ktoré umožňujú takmer úplne autonómny let. Okrem nastavení autonómnosti je táto aplikácia bohatá na navigačné prvky. Najvýraznejším prvkom je zabudovaná mapa, ktorá umožňuje užívateľovi definovať waypointy a takzvané points of interest, ktoré slúžia pilotovi na orientáciu počas jeho misie. Táto mapa je vždy dostupná na okraji obrazovky a po kliknutí sa otvorí na celú obrazovku, čím pilot získa omnoho lepší pohľad o tom, kde sa jeho dron nachádza. Ďalším veľmi zaujímavým prvkom je prvok indikujúci prítomnosť objektov okolo dronu. DJI drony

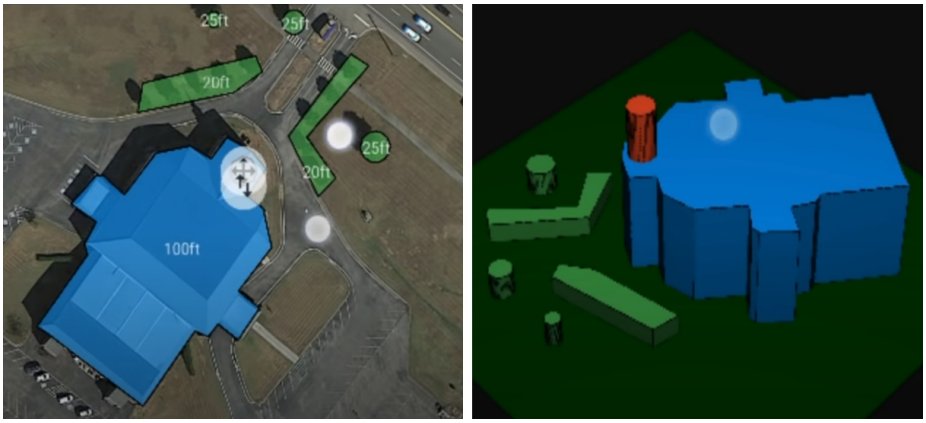
[5] Zdroj obr.: https://blog.studiosport.fr/comment-changer-le-mode-de-votre-radiocommande-frsky-mode-1-et-mode-2/

majú v sebe zabudovanú sadu senzorov, ktoré vďaka špeciálnym senzorom dokážu rozpoznať prekážky nachádzajúce sa v okolí dronu a umožňujú tak autonómny let. Okrem týchto prvkov obsahuje táto aplikácia aj dôležité letové informácie akými sú výška dronu, rýchlosť dronu, alebo vzdialenosť dronu od pilota.

Obrázek 2.6: Rozloženie ovládacích prvkov v aplikácií DJI GO 4.[6]

Drone Harmony

Skvelou alternatívou už spomínanej DJI GO je aplikácia Drone Harmony. Tá rovnako ako DJI GO poskytuje užívateľovi mapu, v ktorej môže plánovať misiu pomocou waypointov. Špecialitou tejto aplikácie sú jej vyspelé možnosti plánovania tejto misie. Aplikácia umožňuje užívateľovi dopredu modelovať štruktúry na mape, na základe ktorých sa bude dron pri lete pohybovať. Ak teda chce pilot nasnímať budovu, stačí v aplikácií definovať hranice budovy a výšku. Aplikácia potom vytvorí 3D model tejto budovy, na základe ktorého si pilot následne môže zvoliť trajektóriu letu. Tento postup je možno vidieť na obrázku 2.7.



Obrázek 2.7: Modelovanie 3D objektu v aplikácií Drone Harmony.[7]

[6] Zdroj obr.: https://store.dji.com/guides/dji-go-4-manual/

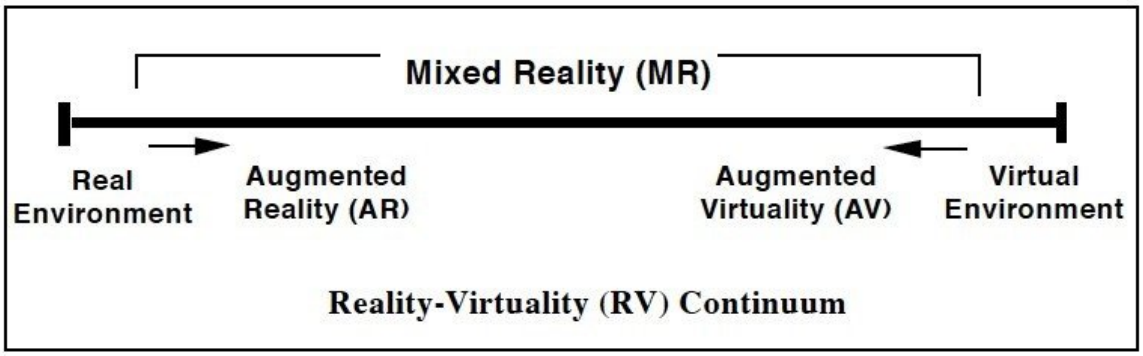
Kapitola 3

Vizualizačná aplikácia VSTool

Výskumná skupina Robo@FIT sa dlhodobo venuje mimo iného aj oblasti interakcie človeka so strojmi. V oblasti ovládania dronov vznikol v minulých rokoch koncept užívateľského rozhrania, ktoré využíva rozšírenú virtualitu a ďalšie GUI prvky s cieľom zlepšiť orientáciu pilota a znížiť tak jeho mentálnu záťaž. Ako realizácia tohto konceptu vznikla experimentálna vizualizačná aplikácia `VSTool`. Základnú verziu aplikácie, v spolupráci s výskumníkmi so skupiny Robo@FIT, spracoval vo svojej diplomovej práci Kamil Sedlmajer [7]. Čiastkové výskumné výsledky boli taktiež publikované v [12]. Aplikácia využíva volne dostupné mapové dáta na vytvorenie 3D scény. Do scény je potom umiestnený živý prenos z kamey dronu. V mojej práci budem nadväzovať na prácu pána Sedlmajera. V tejto kapitole teda stručne zhrniem základné princípy a technológie, na ktorých je aplikácia založená.

3.1 Rozšírená virtualita a rozšírená realita

Hlavnou myšlienkou tejto aplikácie je využiť možnosti, ktoré rozšírená virtualita a rozšírená realita ponúkajú tak, aby pilotovi dronu čo najviac uľahčila prácu. Rozšírená virtualita je menej známy pojem, častejšie sa stretávame s pojmami ako rozšírená realita, alebo virtuálna realita. Tieto názvy môžu pre neznalého pôsobiť ako pomenovania jednej a tej istej veci, opak je však pravdou. Rozdeleniu týchto pojmov sa vo svojej práci venoval Paul Milgram [10]. Ten definuje pojmy rozšírená realita a rozšírená virtualita ako časti spektra zvané `Reality-Virtuality continuum` (obrazok 3.1). Na jednej strane spektra je reálne prostredie obmedzené zákonmi fyziky. Na druhej strane spektra je virtuálne prostredie, ktoré je tvorená čisto virtuálnymi prvkami a môže, ale nemusí pripomínať reálny svet a takisto sa nemusí riadiť zákonmi reálneho sveta. Rozšírená realita a virtualita sú teda len časti tohto spektra, pričom rozšírená virtualita má bližšie k virtuálnemu prostrediu a rozšírená realita bližšie k reálnemu prostrediu. Typicky platí, že pri rozšírenej realite sú do reálneho prostredia umiestnené virtuálne prvky a naopak u rozšírenej virtuality sú do virtuálneho prostredia umiestnené prvky reálneho sveta. Musí však platiť, že poloha virtuálnych prvkov vo virtuálnom prostredí je zhodná s polohou reálnych prvkov prostredia, ktoré to virtuálne reprezentuje.



Obrázek 3.1: Reprezentácia RV kontinua [10].

V aplikácií sa rozšírená virtualita využíva tým spôsobom, že do virtuálnej scény, ktorá je tvorená mapou vygenerovanou pomocou volne dostupných mapových dát je umiestnený model dronu, ku ktorému je pripnuté plátno, na ktorom sa premieta obraz z kamery (obrazok 3.2). Model sa v scéne pohybuje na základe GPS súradníc dronu, ktoré sú prepočítané tak, aby sa poloha modelu vo virtuálnej mape zhodovala s polohou reálneho dronu v reálnom svete. Aplikácia takisto simuluje náklon dronu pomocou dát z jeho elektronického kompasu. Pilot teda môže pomocou plátna ovládať dron ako v bežnej aplikácií z prvej osoby, no v prípade potreby môže kameru oddialiť a vidieť svoj dron, aj keď len virtuálne, z tretej osoby. Pilot teda vidí kde sa dron nachádza, ktorým smerom sa dron pohybuje a čo je najdôležitejšie, vidí prekážky nielen pred dronom, ale aj prekážky okolo dronu. Samozrejme, virtuálna mapa neodpovedá stopercentne realite. Aj napriek tomu však toto riešenie výrazne zlepšuje orientáciu pilota a znižuje riziko havárie. Využitie virtuálnej scény môže byť užitočné aj v prípade, kedy pilot nemá k dispozícií pohľad cez kameru a ovláda dron len pohľadom zo zeme. Pilot vďaka pohľadu do virtuálnej scény nestratí prehľad o tom kam smeruje predok dronu a zároveň môže letieť aj za hranice svojho zorného poľa, pretože polohu dronu môže sledovať priamo v aplikácii.

Obrázek 3.2: Využitie rozšírenej virtuality v aplikácií. Video z dronu plynule naväzuje na virtuálnu scénu [11].

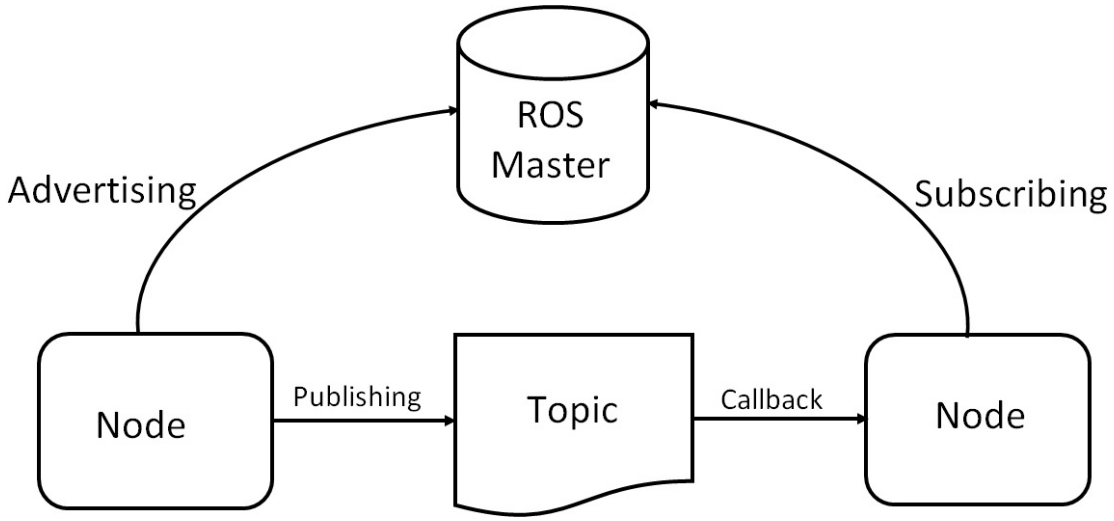
3.2 Unity

Cieľom vyvíjanej aplikácie je vytvoriť 3D virtuálnu scénu, ktorá by odpovedala realite, a za pomoci 2D prvkov tvoriacich užívateľské rozhranie umožniť užívateľovi interakciu s aplikáciou. Bolo teda nutné zvoliť vhodné prostredie, ktoré takúto prácu umožňuje. Vhodným prostredím sa ukázal byť herný engine Unity. Unity je multiplatformový engine vytvorený primárne pre tvorbu 2D a 3D hier na PC, konzole, mobilné zariadenia a web. Rovnako ako u väčšiny iných herných enginov, ako Unreal Engine alebo CryEngine, je používanie na nekomerčné účely zdarma. Na skriptovacie sa v Unity využíva jazyk C#. Unity si od svojho vzniku v roku 2005 vybudovalo pomerne veľkú komunitu, vďaka čomu je na internete veľké množstvo voľne dostupných knižníc a komponent ktoré značne uľahčujú vývoj. Súčasťou editoru je aj Unity Asset Store, kde môžu vývojári umiestniť svoje predpripravené komponenty, väčšina z nich je však platená. Aplikácia v Unity je tvorená takzvanými scénami. Tieto scény sú navzájom nezávislé a každá z nich tvorí hierarchiu objektov zvaných `GameObject`.U 3D aplikácie sa scéna skladá z 2 základných častí. Z 3D scény, ktorú tvoria 3D objekty, a z takzvaného plátna(Canvas), ktoré tvoria 2D objekty tvoriace užívateľské rozhranie, ako sú tlačidlá, panely, textové polia a tak ďalej. Každý 3D objekt má základný komponent zvaný `Transform`, ktorý určuje jeho polohu, rotáciu a mierku na osách X,Y a Z. U 2D objektov je komponent `Transform` nahradený komponentom `RectTransform`. Ten určuje polohu relatívnu k zvolenému stredu 2D súradnicovej sústavy, rozmery, rotáciu a mierku objektu. Unity obsahuje veľké množstvo už pripravených objektov. Vývojár si však tieto objekty môže ľubovoľne upravovať pridávaním, odoberaním a menením parametrov predprivených komponentov, ktoré sú k dispozícií. Rovnako ako predpripravené komponenty ide k jednotlivým objektom pridať aj skripty, ktoré dokážu meniť parametre objektu na ktorom je skript umiestnený, ale aj iných objektov, na ktoré sa môže skript

jednoducho odkazovať. Na to, aby skript bežal, musí byť pripojený k niektorému objektu v scéne. Každý Unity script implementuje svoju vlastnú triedu, ktorá dedí zo základnej triedy zvanej `MonoBehavior`. Táto trieda implementuje 2 najzákladnejšie metódy, ktoré sa využívajú takmer v každom Unity skripte. Sú to metódy `Start` a `Update`. Metóda `Start` sa automaticky vykoná pri spustení aplikácie a metóda `Update` sa vykonáva v každom jednom snímku [5].

3.3 Prepojenie s dronom - ROS

Po zvolení vhodného prostredia pre vývoj aplikácie, bolo potrebné vymyslieť spôsob, akým by aplikácia komunikovala s dronom. Bolo potrebné zaistiť, aby aplikácia mala v každom momente prehľad o tom, kde sa dron nachádza, jak je naklonený a mala prístup k videu z kamery dronu. Ako vhodný protokol na tento prenos týchto dát sa ukázal protokol RosBridge, ktorý je súčasťou ROSu – Robot Operating System. ROS nie je operačný systém, je to skôr sada open-source knižníc, ovládačov a nástrojov, ktoré majú užívateľovi uľahčiť vývoj v oblasti robotiky. Architektúra ROSu je založená na modulárnom princípe. Tvoria ho takzvané uzly. Uzol je v podstate software, ktorý má na starosti jednoduchý proces, ako je napríklad čítanie dát zo senzoru. Tieto uzly sú spolu prepojené pomocou takzvaného publish-subscribe protokolu. Znamená to, že ak má uzol dáta ktoré chce zdielať, zdieľa ich pomocou takzvaného topic-u (stáva sa z neho publisher) a pokiaľ iný uzol potrebuje tieto dáta, prihlási sa na odber tohto topic-u (stáva sa z neho subscriber) [1] [2]. V našom prípade sú uzly ktoré nás zaujímajú tie, ktoré poskytujú dáta z kamery dronu a dáta o lokácii a rotácii dronu. ROSbridge poskytuje aplikačné rozhranie pre prácu s ROSom pre aplikácie, ktoré nie sú priamo založené na ROSe. Komunikáciu medzi ROSbridge serverom a aplikáciou sprostredkuje knižnica ROSsharp. Zjednodušená architektúra ROSu je zobrazená na obrázku 3.3.



Obrázek 3.3: Zjednodušený model architektúry ROSu. [1]

[1] Zdroj obr.: https://www.designnews.com/gadget-freak/ros-101-intro-robot-operating-system

3.4 Virtuálna mapa

Posledným problémom, ktorý bolo potrebné riešiť, aby bola aplikácia kompletná, bolo získať mapové dáta a zobraziť ich v Unity. Najkvalitnejším zdrojom mapových dát sa ukázali mapy od spoločnosti Google. Tá bohužiaľ neposkytuje tieto dáta tretím stratám, preto bol zvolený voľne dostupný plugin Mapbox využívajúci dáta z OpenStreetMap. Súčasťou Mapboxu je objekt, ktorý automaticky vygeneruje Mapu do scény na základe parametrov, ktoré sú nastavené pri spustení aplikácie. Mapa je tvorená štvorcovými dlaždicami, každej o veľkosti 400x400m. Užívateľ si môže zvoliť stred a veľkosť vygenerovanej Mapy. Veľkosť predstavuje počet dlaždíc, ktoré sa vygenerujú v každom smere od nastaveného stredu. Objekt Mapy takisto umožňuje zvoliť textúru terénu. Na výber je satelitný pohľad alebo klasická mapa. Užívateľ môže medzi týmito dvoma textúrami v priebehu používania ľubovoľne prepínať. Zvyšné parametre, ako výška budov, alebo mierka mapy užívateľ meniť nemôže. Mierka mapy je nastavená tak, aby jeden meter predstavoval jednu jednotku vzdialenosti v Unity (1 Unity units), čo výrazne zľahčuje prácu so vzdialenosťami medzi jednotlivými objektami. Výška budov je takisto zmenená tak, aby aspoň čiastočne odpovedala realite, pretože dáta o budovách v Mapboxe nie sú presné a základná výška je príliš nízka [3] [11].

Kapitola 4

Návrh riešenia

4.1 Požiadavky na aplikáciu

Ako už bolo spomenuté v kapitole 2.2, drony sú často využívané špeciálnymi zložkami, ako sú polícia, alebo záchranné jednotky. Tie využívajú tieto zariadenia na pomerne náročné úlohy, ako napríklad monitorovanie náročne dostupných oblastí v prípade prírodných katastrof, alebo hľadanie stratených osôb. Pri takomto použití je ovládanie dronu omnoho náročnejšie, ako pri bežnom používaní, pretože v týchto prípadoch musí často pilot letieť mimo hranicu jeho zorného pola a nemá tak presný prehľad o tom, kde sa jeho zariadenie nachádza. Zároveň býva prostredie, v ktorom sa s dronom nachádza, omnoho náročnejšie na orientáciu než pri jednoduchom prelete nad lúkou, alebo menej zastavanou oblasťou. Typicky sa pri takýchto náročných úlohách využíva hneď niekoľko dronov, ktorých piloti musia medzi sebou spolupracovať. Pilot tak musí mať naraz prehľad nielen o tom, kde sa s dronom nachádza, ale aj kde sa nachádzajú drony, s ktorými spolupracuje a to, kde sa nachádza cieľ, ku ktorému sa má s dronom dostať. Požiadavkou na aplikáciu, ktorou pilot ovláda drona je teda, aby obsahovala čo najviac týchto informácií, čím sa výrazne zníži mentálna záťaž pilota. Súčasná verzia aplikácie VSTool odpovedá na tieto požiadavky len čiastočne. V tejto kapitole identifikujem problémy, ktoré by mohli vzniknúť pri použití aplikácie v takýchto prípadoch a ponúknem riešenia, ktoré by už spomínané požiadavky naplnili.

Nedostatky aktuálneho riešenia

Navigačné prvky

Hlavným nedostatkom súčastného riešenia sú značne obmedzené navigačné prvky. Aplikácia obsahuje 3 základné prvky, ktoré by mali pilotovi uľahčiť orientáciu a tými sú homepoint, waypoint a zóna. Homepoint predstavuje štartovaciu pozíciu dronu a umožňuje pilotovi jednoduchý návrat v prípade, že odletí príliš ďaleko a stratí orientáciu. Tento homepoint je pri spustení aplikácie spoločne s dronom umiestnený v strede vygenerovanej mapy, ale je možné ho premiestniť na aktuálnu pozíciu dronu, ak je potreba. Waypoint predstavuje dôležité miesto na mape, ku ktorému by sa mal pilot s dronom dostať. Implementované sú 2 typy waypointov, waypoint na zemi a waypoint vo vzduchu. Problémom sú veľmi obmedzené možnosti umiestnenia týchto waypointov. Aplikácia totiž neobsahuje zabudované užívateľské rozhranie pre ich umiestnenie. Waypoint vo vzduchu je možné umiestniť len na aktuálnu pozíciu dronu a waypoint na zemi je možné umiestniť kliknutím na zem. Pilot by

teda musel pri každom spustení aplikácie využiť zabudovaný simulátor a umiestniť tieto waypointy pred tým ako zaháji let. Okrem toho, že by takéto nastavovanie zabralo priveľa času a pilot by musel veľmi dobre poznať oblasť v ktorej lieta, aby vedel identifikovať, kde do scény má waypoint umiestniť, zároveň by musel mať vždy k dispozícii gamepad, pretože simulátor nie je možné ovládať pomocou klávesnice. Rovnako ako u waypointu na zemi aj hranice zóny je možné umiestniť len kliknutím na zem, ale na rozdiel od waypointov, ktorých môže užívateľ umiestniť ľubovoľne veľa, aplikácia umožňuje umiestniť len jednu zónu, čo je značne obmedzujúce. O navigáciu k waypointom sa stará dvojica šipiek spolu s meradlom vzdialenosti ktoré sú umiestnené pod modelom dronu. Jedna z nich naviguje pilota k zvolenému waypointu a druhá k homepointu. Tieto šípky dynamicky menia svoju polohu v priestore a smerujú vždy k zvolenému waypointu. Problém je, že v niektorých uhloch je pri nižšom rozlíšení ťažšie odhadnúť kam šípka ukazuje. Nakoľko je v pláne v budúcnosti aplikáciu preniesť na mobilné zariadenia, bude nutné upraviť aj tento navigačný prvok.

Podpora viacero dronov

Ďalším defektom, na ktorom by sa dalo zapracovať je fakt, že aplikácia podporuje vizualizáciu len jedného dronu. Ak by teda pilot chcel pomocou aplikácie sledovať 2 drony musel by aplikáciu spustiť 2krát, čo nie je úplne ideálne. Zároveň je pilot odkázaný len na dáta z ROSu. Pilot by tak musel pre každý dron, ktorý chce vizualizovať nastaviť prevod dát z flight controlleru na dáta ktoré dokáže ROS spracovať, čo je pre bežného v oblasti robotiky neskúseného užívateľa takmer nemožná úloha.

Užívateľské rozhranie

Aplikácia v súčasnej podobe obsahuje veľmi obmedzené užívateľské rozhranie. Tvorí ho len malá skupina tlačidiel z ktorých väčšina slúži na umiestnenie a navigáciu k implementovaným navigačným prvkom. Zvyšné tlačidlá slúžia na zmenu zdroju letových dát a na zobrazenie virtuálneho plátna. Všetky nastaviteľné parametre, ako adresa RosBridge serveru, nastavenia videa, alebo parametre vygenerovanej mapy umožňuje aplikácia užívateľovi nastaviť pri spustení. Problémom je, že toto nastavenie je umiestnené v odlišnej scéne ako hlavná časť aplikácie a užívateľ medzi týmito scénami nemá možnosť prepínať. Ak teda zadá niektorý z parametrov chybne, čo je pri parametroch ako url adresa, alebo zemepisná šírka bežné, musí aplikáciu reštartovať a tieto dáta zadať znovu.

4.2 Tvorba misie

Ako už bolo spomenuté v predchádzajúcej kapitole 4.1, navigačné prvky aplikácie neboli úplne dotiahnuté do konca a z toho dôvodu by bola vizualizácia aj jednoduchej misie typu prileť do zóny, preleť cez 3 waypointy a pristaň na určené miesto značne náročné. Prvým problémom, ktorý bolo teda potrebné vyriešiť, bolo navrhnúť dátovú štruktúru, ktorá by reprezentovala misiu tak, aby s ňou dokázali užívateľ aj aplikácia jednoducho pracovať. Základnými prvkami tejto štruktúry by mali byť časti, ktoré aplikácia už podporuje. V tomto prípade sú to:

- Dron - Model dronu, ktorý chceme vizualizovať
- Waypoint - Zaujímavý bod v scéne, ku ktorému by mal pilot s dronom smerovať

- Zóna - Môže predstavovať miesto, kam sa má pilot s dronom dostať, alebo naopak, ktorému by sa mal vyhnúť

Nakoľko pracujeme s mapovými dátami, drona a waypoint môžeme jednoducho reprezentovať pomocou trojice zemepisná šírka, zemepisná dĺžka a nadmorská výška (alebo výška nad zemou). Zónu potom reprezentujeme ako množinu týchto trojíc, kde výška predstavuje výšku hrany zóny. Konkrétna misia by mala byť tvorená postupnosťou dvojíc typu Dron-Waypoint, alebo Dron-Zóna, pričom musí platiť, že ku každému waypointu môže náležať ľubovoľný počet dronov, väčší ako nula. K zóne nemusí náležať ani jeden dron, v tom prípade môžeme túto zónu považovať za zónu, ktorej by sa mal pilot vyhnúť a náležite ju v aplikácii odlíšime od bežnej zóny. Využitie dronov je naozaj široké (viz. 2.2), predpokladá sa teda, že pilot nebude iba lietať z bodu A do bodu B, ale bude počas letu vykonávať aj určité úlohy. U waypointu teda budeme musieť rozlišovať 2 typy. Prvým typom bude klasický waypoint, ktorý je aktivovaný po tom, ako sa všetky drony, ktoré mu náležia dostanú do určitej vzdialenosti a slúži len ako navigačný bod v priestore. Vzdialenosť, na ktorú bude waypoint aktivovaný by mala byť nastaviteľná podľa druhu misie. V niektorých prípadoch môže stačiť vzdialenosť 100 metrov, v iných potrebujeme byť presní na jednotky metrov. Druhým typom je waypoint, ktorý bude musieť pilot ručne aktivovať a predstavuje určitý úkon, ktorý má pilot vykonať. Tento typ by mal mať v sebe zabudovaný aj určitý popis danej úlohy, ktorý bude aplikácia pilotovi zobrazovať.

4.3 Užívateľské rozhranie

K týmto prvkom bolo potrebné navrhnúť jednoduché užívateľské rozhranie, ktoré by užívateľovi zobrazovalo všetky dôležité informácie o stave misie a zároveň uľahčilo orientáciu v scéne. Užívateľ by mal mať prehľad o tom:

Kde sa jeho dron nachádza - užívateľ by mal prehľad o tom, kde sa jeho dron nachádza a čo sa nachádza v okolí dronu

Aká je jeho úloha v misií - ak má v priebehu misie užívateľ vykonať určitú úlohu mal by vedieť akú a po vykonaní ju možnosť potvrdiť

Kde sa nachádza bod ku ktorému smeruje - užívateľ musí vedieť kam má s dronom smerovať a ako ďaleko sa tento bod nachádza

Kde sa nachádzajú ďalšie drony podieľajúce sa na misií - v prípade potreby by mal užívateľ vedieť, ktorým smerom a ako ďaleko sa nachádzajú zvyšné drony. Zároveň by mal vedieť čo sa nachádza v ich okolí.

Kam smerujú ďalšie drony podieľajúce sa na misií a aká je ich úloha - užívateľ tak vie na ktoré drony sa má zamerať (tie ktoré majú rovnakú úlohu ako on) a ktoré môže ignorovať.

Keďže virtuálna scéna vytvorená pomocou Mapboxu obsahuje len zem a budovy, odhad vzdialeností bez iných záchytných bodov v scéne je takmer nemožný. Preto by bolo vhodné okrem rozhrania pre zobrazovanie dát o misií navrhnúť aj prvok, ktorý by dal pilotovi možnosť vidieť, ako ďaleko sa nachádza od jednotlivých prvkov v scéne a mohol sa tak efektívnejšie vyhnúť prípadnému nebezpečenstvu. Tento prvok by nemal byť prítomný počas celého letu, ale iba v prípade, že sa dron naozaj nachádza v blízkosti prekážky alebo

je veľmi nízko nad zemou. Zároveň je treba zaistiť, aby sa intenzita tohto prvku úmerne zväčšovala s blízkosťou od prekážky. Pilot tak ľahšie vyhodnotí, ktorej prekážke by mal venovať zvýšenú pozornosť a kam sa v prípade potreby môže s dronom bezpečne pohnúť. Ako však takéto niečo vizualizovať? Typickými aplikáciami ktoré využívajú podobné prvky sú moderné počítačové a konzolové hry, ktoré takto zobrazujú smer útoku nepriateľa. Tie na vizualizáciu nebezpečenstva z pravidla využívajú jeden z týchto spôsobov:

Indikátor na okraji obrazovky
Pri tomto spôsobe sa zvýrazňujú okrajové časti obrazovky podľa smeru útoku (obrázok 4.1). Typicky platí, že zvýraznenie horného okraju predstavuje útok spredu a zvýraznenie spodného okraju útok zozadu. Tento spôsob je pomerne nejasný a preto sa často využíva v kombinácii s inými spôsobmi.

Obrázek 4.1: Indikátor na okraji obrazovky v hre Halo. [1]

2D Indikátor
2d indikátor je v súčasnej dobe najpoužívanejším a zároveň najzrozumiteľnejším spôsobom ako zobrazovať smer útoku. Je to v podstate vylepšená verzia indikátoru na okraji obrazovky, ibaže indikátor je premiestnený viac do stredu obrazovky a zobrazuje aj uhol odkiaľ útok prichádza, nie len stranu (obrázok 4.2). Rovnaká konvencia platí aj pri tomto spôsobe.

[1] Zdroj obr.: https://www.reddit.com/r/halo/comments/ulyyc/halo_4_multiplayer_grenade_indicator_damage/

Obrázek 4.2: 2D indikátor v hre FarCry 5.

3D Indikátor

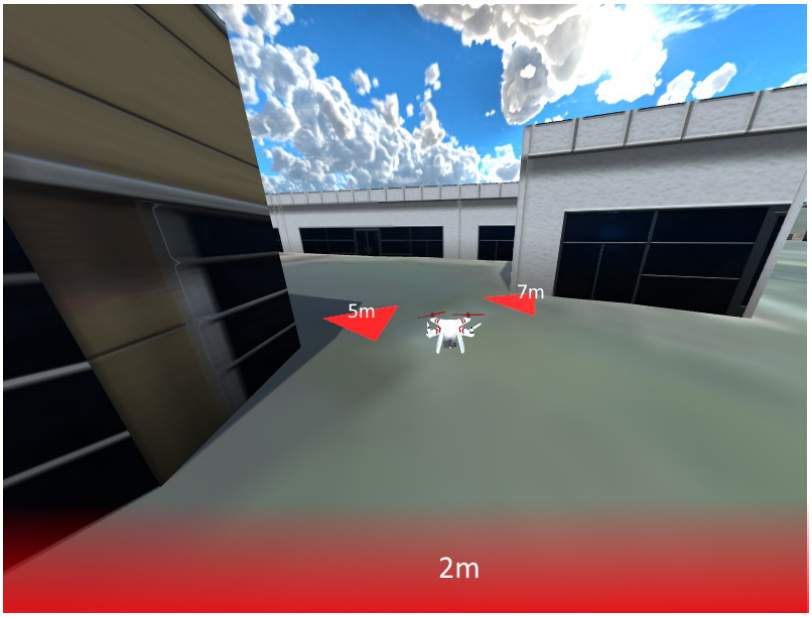
Niektoré hry využívajú indikátor umiestnený priamo v 3D scéne (obrázok 4.3). Indikátor tak priamo smeruje na nepriateľa, čo výrazne zlepšuje orientáciu hráča v scéne. Toto je jediné riešenie, ako zobraziť nebezpečenstvo zo všetkých 3 dimenzií. Problémom tohto riešenia je, že v niektorých uhloch je naozaj ťažké určiť kam vlastne indikátor smeruje [13].

Obrázek 4.3: 3D indikátor použitý v hre Destiny2. [2]

[2]Zdroj obr.: https://youtu.be/du7pJa4cqYE

4.4 Zobrazenie nebezpečenstva

Prvým spôsobom, ktorý sme skúšali využiť bol 2D indikátor. Problém s týmto riešením je, že nie je možné naraz vizualizovať nebezpečenstvo z vrchu a spredu ani zo zadu a zospodu. Tento problém by bol čiastočne eliminovaný pohľadom cez kameru, kedy by sme nemuseli zobrazovať nebezpečenstvo spredu, pretože pilot by dokázal sám vyhodnotiť, čo sa pred ním nachádza. Stále by sme však nedokázali vizualizovať nebezpečenstvo zo zadu. To môže byť problém v prípade, že predok dronu je kolmý na stenu budovy. Vtedy je kamera, ktorá sníma scénu v budove a nakoľko textúra budovy nie je vykreslená z vnútra, pilot by nemal ako vedieť, že sa za ním nachádza budova. Zároveň by aplikácia mala byť použiteľná aj v prípade, že pilot nemá k dispozícii video z kamery, preto tento spôsob nie je vhodný. Ako optimálne riešenie sa ukázalo použiť kombináciu 3D indikátoru a indikátoru na okraji obrazovky tak, že 3D indikátor zobrazuje nebezpečenstvo na horizontálnej osi, čiže vpredu, vzadu a po bokoch a o zobrazenie nebezpečenstva zospodu a zvrchu sa stará indikátor na vrchnom a spodnom okraji obrazovky (obrazok 4.4).



Obrázek 4.4: Návrh vizuálneho prvku na zobrazenie nebezpečenstva za pomoci kombináciu 3D indikátoru a indikátoru na okraji obrazovky.

4.5 Lokalizácia dronu

Prvou úlohou navrhnutého rozhrania teda bolo, aby mal užívateľ vždy prehľad o tom, kde presne sa jeho dron nachádza. Samozrejme, užívateľ vidí čo sa nachádza okolo dronu vďaka virtuálnej scéne a aplikácia mu umožňuje zapnúť navigáciu k bodu vzletu, čo by malo stačiť na základnú orientáciu. Problém však môže nastať ak pilot letí do oblasti, ktorú nepozná a nastane situácia, kedy stratí kontrolu nad dronom. To môže nastať napríklad, ak vypadne signál, alebo sa vybije batéria. V tomto prípade musí užívateľ okamžite vedieť, kde sa zariadenie nachádza, aby sa k nemu mohol dostať čo najskôr. Preto by bolo vhodné, aby užívateľ mohol vidieť väčšiu časť okolia dronu. Tento problém by veľmi elegantne vyriešila implementácia jednoduchej minimapy. Pilot by tak mal k dispozícií pohľad z vtáčej per-

spektívy na celú oblasť v ktorej sa jeho dron nachádza. Vďaka možnosti Mapboxu ľubovoľne prepínať medzi satelitnou a klasickou mapou, orientácia by bola jednoduchá aj v meste, kde by bolo vidieť okolité ulice, aj v divočine vďaka satelitným snímkam terénu. Pohľad cez minimapu by nemal byť statický. Užívateľ by mal mať možnosť pohľad ľubovoľne približovať a odďalovať. Návrh realizácie minimapy je zobrazený na obrázku 4.5.

4.6 Navigácia k prvkom misie

Prvky misie ako drony a waypointy sú v scéne veľmi malé a pri prechode scénou sú viditeľné až pri veľmi nízkej vzdialenosti. Preto by bolo vhodné, aby tieto prvky boli špeciálnym spôsobom vyznačené, aby ich užívateľ dokázal v scéne jednoducho lokalizovať. V drvivej väčšine 3D aplikácií sa na zvýraznenie vzdialených objektov využíva prístup, kde k objektu je akoby pripnutá 2D ikona, ktorej pozícia na pláne sa mení v závislosti od polohy sledovaného objektu a rotácie kamery snímajúcej scénu. V našom prípade by bolo vhodné, aby sa táto ikona dynamicky menila napríklad na šípku v závislosti od toho, či sa objekt nachádza v zornom poli kamery. Pilot by tak v každom momente dokázal určiť smer kam má s dronom smerovať. Keďže veľkosť ikony by sa nemala meniť, je dôležité, aby ikona obsahovala informáciu o vzdialenosti objektu od sledovaného dronu, aby mal pilot predstavu o tom, kde sa daný objekt v scéne naozaj nachádza. Ikona sledovaného waypointu by mala byť vždy dostupná a vizuálne odlíšená od ostatných. Čo sa týka ikon dronov, tieto sa neoplatí zobrazovať vždy, pretože pri prítomnosti veľkého množstva dronov by mohlo dojsť k dezorientácií. Preto je potreba si určiť parameter, na základe ktorého budeme tieto ikony zobrazovať. Môže ním byť napríklad vzdialenosť od ovládaného dronu, alebo úloha, ktorú práve daný dron vykonáva. Koncept takejto navigácie je ilustrovaný na obrázku 4.5.

4.7 Zobrazenie dát o misií

Dáta o misií, ako popis úlohy, alebo to kam smerujú zvyšné drony, prirodzene musíme oddeliť od hlavnej časti aplikácie, pretože by zaberali príliš veľa miesta, čo by zhoršovalo orientáciu pilota. Ideálne by teda bolo, aby sme vyhradili časť plátna práve pre tieto informácie. Týmto riešením sa mierne zmenší zorné pole kamery, ale nakoľko sa takto posunie pohľad cez kameru, táto strata sa rovnomerne rozloží na obe strany obrazu, čo v konečnom dôsledku nespôsobí až takú výraznú zmenu. Na toto uvolnené miesto môžme potom umiestniť zoznam dronov, ktoré sa v scéne nachádzajú spoločne s informáciou o ich aktuálnej úlohe v misií (obrázok 4.5). Pilot tak bude vedieť ktoré drony sú pre jeho úlohu podstatné a na tie sa môže zamerať. Ďalšou užitočnou funkciou, ktorú by rozhranie mohlo implementovať, je pohľad cez kameru druhého dronu. Užívateľ by tak mal okrem smeru letu a vzdialenosti prehľad aj o tom kde presne sa dron o ktorý sa zaujíma nachádza, čo môže byť v mnohých situáciach výhodné.

Obrázek 4.5: Mockup navrhnutého riešenia zobrazujúci informácie o misií spoločne s polohou jednotlivých prvkov misie.

4.8 Sprístupnenie funkcií využívajúcich klávesnicu

Keďže aplikácia by mala byť použiteľná aj na dotykových zariadeniach, je potrebné užívateľovi umožniť prístup ku všetkým funkciám, ktoré boli v pôvodnej verzií dostupné len pomocou klávesnice. Menovite sú to:

- Rotácia kamery (WSAD alebo šípky)
- Priblíženie a oddialenie kamery (Kolečko myši)

Niektoré drony je možné ovládať pomocou takzvaného FPV headsetu. Je to zariadenie podobné VR headsetu, vďaka ktorému môže pilot riadiť dron z prvej osoby. Toto zariadenie dokáže sledovať pohyb hlavy pilota a na základe jeho pohybu meniť rotáciu kamery dronu. Tento koncept sme chceli aspoň čiastočne preniesť do našej aplikácie. To by sme čiastočne dosiahli tým, že by sa rotácia kamery menila s rotáciou tabletu na ktorom by aplikácia bežala. Toto riešenie však nie je úplne praktické. Predpokladá sa, že pilot sa bude prevažne sústrediť na ovládanie dronu pomocou RC vysielača a aplikáciu na tablete tak bude ovládať maximálne jednou rukou. Preto som od tohto riešenia upustili a pozornosť sa upriamila práve na to, aby bola aplikácia použiteľná jednou rukou. Na takéto jednoduché ovládanie kamery potom stačí len štvorica tlačidiel na pohyb kamera po horizontálnej a vertikálnej ose a dvojica, ktorá by sa starala o priblíženie a oddialenie obrazu. Dôležité však je, aby mal užívateľ možnosť meniť polohu tohto prvku. Tým docielime, že aplikácia ovládateľná aj pravou, aj ľavou rukou.

Kapitola 5

Implementácia

Navrhnuté prvky užívateľského rozhrania boli začlenené do už existujúcej aplikácie VSTool. Základná architektúra aplikácie je popísaná v kapitole 3. Táto kapitola obsahuje vybrané implementačné detaily navrhnutých prvkov a spôsob ich začlenenia do aplikácie.

5.1 Parametre misie

Prvou úlohou, ktorú bolo potrebné vyriešiť, je určiť formát, v ktorom bude môcť užívateľ definovať parametre misie, ktorú má aplikácia vizualizovať. Keďže v budúcnosti má aplikácia pracovať na architektúre klient-server, bolo nutné zvoliť jeden z bežne používaných serializačných formátov, aby bolo prípadne možné načítať misiu priamo z dát zo serveru. Z tohto dôvodu som sa rozhodol pre formát JSON, ktorý je v súčasnej dobe veľmi rozšírený a zároveň je jednoduchý na pochopenie aj pre užívateľa, ktorý sa nevenuje informatike.

```
{
    "checkpoints":[
        {
            "drones":["Drone1"],
            "type":"confirm",
            "name":"Point 1",
            "points":[
                {
                    "height": 80,
                    "latitude": 49.228634,
                    "longitude": 16.597110
                }
            ],
            "description":"Take a photo"
        }
    ],
    "drones":[
        {
            "name":"Drone1",
            "latitude": 49.226020,
            "longitude": 16.597149,
            "url":
```

```
        }
    ]
}
```

Misia má 2 základné polia parametrov, pole `checkpoints`, ktoré reprezentuje waypointy a zóny v misii, a pole `drones`, ktoré ako, už z názvu vyplýva, reprezentuje drony podieľajúce sa na misií. Jednotlivé checkpointy sú definované piatimi parametrami:

name – Jednoznačný identifikátor checkpointu.

type – Určuje typ checkpointu. Podporované sú 3 základné typy:

- `regular` – Jednoduchý waypoint, ktorý je považovaný za splnený po tom, čo sa k nemu dostanú všetky pridelené drony.
- `confirm` – Waypoint s úlohou, ktorú musí pilot ručne potvrdiť. Je aktivovaný po tom, ako sa k nemu dostanú všetky pridelené drony a pilot potvrdí splnenie úlohy.
- `zone` – Geo-Zóna, reprezentuje zónu, do ktorej sa majú drony dostať, alebo ktorej sa majú vyhnúť. Je aktivovaná po tom, ako sa do nej dostanú všetky pridelené drony.

points – Pole bodov reprezentovaných zemepisnou šírkou `latitude`, dĺžkou `longitude` a výškou nad zemou, ktoré určujú polohu checkpointu na mape. Pri typoch `regular` a `confirm` sa berie prvý prvok ako poloha waypointu, pri `zone` tieto body reprezentujú hrany zóny.

drones – Pole identifikátorov dronov, ktorým bol pridelený daný checkpoint. Pri type `zone` môže byť toto pole prázdne, vtedy sa zóna považuje za tú, ktorej sa majú drony vyhnúť.

description – Reprezentuje popis úlohy pre typ `confirm`. Pre zvyšné typy je tento parameter voliteľný.

Jednotlivé drony podieľajúce sa na misií sú reprezentované štvoricou `name`, `latitude`, `longitude`,`url`, kde `longitude` je jednoznačný identifikátor dronu a dvojica `latitude`, `longitude` sú súradnice bodu, kde sa bude dron pri štarte misie nachádzať. Parameter `url` je nepovinný a umožňuje zadať webovú adresu, ktorá obsahuje aktuálnu polohu dronu v rovnokom formáte, ako je definovaný waypoint teda trojica latitude,longitude a height. Súbor, ktorý obsahuje túto konfiguráciu sa nazýva `mission.json` a je umiestnený v priečinku `StreamingAssets`. `StreamingAssets` je špeciálny priečinok, ktorého súbory nie sú pri preklade aplikácie zahrnuté do projektu a sú tak prístupné aj po zostavení aplikácie v tomto priečinku. Umiestnenie tohto priečinku vracia premenná `Application.streamingAssetsPath` a líši sa v závislosti od operačného systému. O spracovanie a beh misie sa stará trieda `MissionHandler`. Tá pri spustení aplikácie načíta parametre misie z `mission.json`. Na prevod JSON-u na objekt, s ktorým dokáže Unity pracovať, slúži metóda `FromJson` triedy `JsonUtility`. Táto metoda berie ako parameter reťazec vo formáte JSON a vracia triedu, ktorej parametre odpovedajú dátam z tohto reťazca. Trieda na ktorú sa JSON prevádza musí byť dopredu definovaná a názvy a typ jej parametrov sa musia zhodovať. Ak niektorý z parametrov nie je zadaný, nastaví sa mu základná hodnota daná jej konštruktorom [5]. Táto trieda teda môže obsahovať aj parametre, ktoré JSON nedefinuje, čo sa hodí napríklad na ukladanie referencií na GameObject-y ktoré dané parametre reprezentujú.

5.2 Generovanie objektov

Získanie polohových dát

Po tom, čo sú dáta z konfiguračného súboru načítané, je nutné umiestniť jednotlivé prvky misie do scény. Na to, aby sme mohli určiť, kde sa majú jednotlivé prvky v scéne nachádzať, potrebujeme konvertovať zemepisné súradnice na súradnice, ktorým Unity rozumie. Na tento prevod má Mapbox k dispozícií metódu `GeoToWorldPosition`. Tá berie ako parameter zemepisneú šírky a dĺžky a vracia polohu prekonvertovanú do Unity súradníc. Táto metóda však nepracuje s nadmorskou výškou, preto musíme použiť aj Mapbox metódu `QueryElevationInUnityUnitsAt`, ktorá vracia nadmorskú výšku terénu na zadanej polohe. Keďže je mierka Mapy nastavená tak, aby jeden meter odpovedal jednotke vzdialenosti v Unity, pre prevod výšky nad zemou stačí k výške terénu prirátať zadanú výšku. Obe tieto triedy sú implementované v triede `AbstractMap`, ktorá je použitá ako komponent v objekte `Map`, ktorý sa stará o generovanie mapy a je priamo súčaťou pluginu Mapbox. Pri testovaní sa občas stávalo, že tieto funkcie vracali chybné údaje a objekty tak boli rozmiestnené náhodne v scéne. Problémom bolo, že pri spustení aplikácie sa najprv vykonala metóda Start triedy, ktorá generovala objekty a až potom sa vykonala metóda Start triedy `AbstractMap`. Mapa teda ešte nebola nakonfigurovaná v momente kedy nastalo konvertovanie zemepisných súradníc na Unity súradnice a Mapbox metódy `GeoToWorldPosition` a `QueryElevationInUnityUnitsAt` tak vracali náhodné dáta. Našťastie v Unity je možnosť nastaviť poradie v akom budú jednotlivé skripty vykonané pri štarte aplikácie v ponuke `Edit > Project Settings > Script Execution Order`.

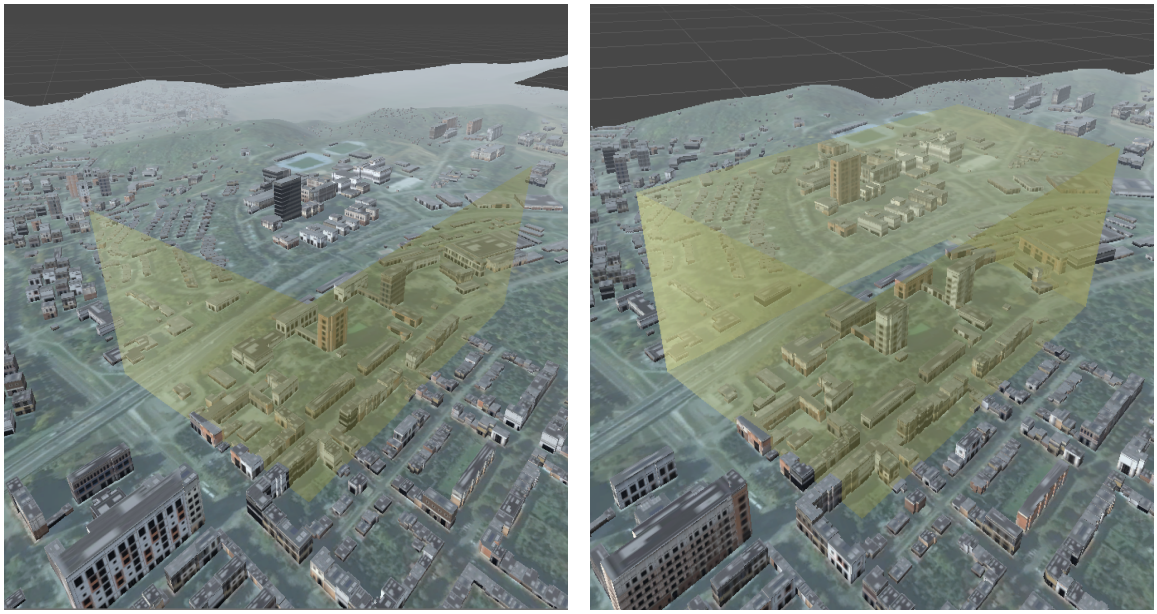
Generovanie jednoduchých objektov

Pre generovanie dronov, ktoré užívateľ priamo neovláda, bolo potreba vytvoriť takzvaný prefab z pôvodného objektu dronu, teda predpripravený objekt, ktorý dokáže Unity instanciovať. Z tohto prefabu bolo nutné odstrániť plátno, na ktorom sa premieta video z kamery, a skript starajúci sa o pohyb dronu `DroneController`, pretože pri použití simulátoru reagovali na použitie gamepadu všetky drony v scéne, čo nechceme. Keďže waypointy už boli implementované v pôvodnej verzií, pre tieto už boli prefaby k dispozícií. O umiestňovanie dronov sa stará metóda `SetupDrones` a o umiestňovanie jednoduchých waypointov metóda `GeneratePoint`. Obe tieto metódy najskôr instanciujú objekt do scény a následne mu priradia polohu vypočítanú za pomoci už spomenutých metód `GeoToWorldPosition` a `QueryElevationInUnityUnitsAt`.

Generovanie zóny

Na rozdiel od generovania waypointov a dronov, ktoré stačilo jednoducho instanciovať a umiestniť na zodpovedajúcu polohu, generovanie zóny v podobe n-uholníku bola výrazne zložitejšia úloha. Základné 3D objekty ktoré umožňuje pridať sú kocka, guľa, kapsula a valec. Prvotným plánom teda bolo využiť objekty, ktoré už Unity ponúka, ako steny n-uholníku, ktorý chceme vygenerovať. Keďže Unity umožňuje ľubovoľne meniť mierku objektov, mohli by sme použiť ako hranicu kocku, ktorú by sme umiestnili medzi 2 hrany n-uholníka a jej mierku nastavili tak, aby sa jej šírka zhodovala s dĺžkou steny medzi týmito hranami a výška s výškou zadanou v konfiguračnom súbore. Výhodou by tak bolo, že každú stenu zóny reprezentoval jeden GameObject a tak by sa ku každej stene dalo v prípade potreby samostatne pristúpiť. Po implementácií tohto mechanizmu však vznikli problémy, kvôli ktorým toto rie-

šenie nebolo vyhovujúce. Problém nastával pri zónach s väčším počtom stien, kde posledné generované steny neboli vygenerované správne a v zóne tak vznikali diery. To bolo zrejme zapríčinené tým, že pri mierka je v Unity reprezentovaná trojicou premenných typu float a tak pri výpočtoch polohy posledných stien vznikali nepresnosti. Našťastie Unity umožňuje generovať vlastné objekty zvané `Mesh`. Na to, aby objekt mohol vygenerovať Mesh musí obsahovať komponenty `MeshFilter`, ktorý ukladá štruktúru naprogramovaného objektu a `MeshRenderer`, ktorý sa stará o vykreslovanie a umožňuje okrem iného nastaviť textúru výsledného objektu. Každý mesh je reprezentovaný polom trojuholníkov `triangles` a polom `vertices`, ktoré reprezentuje jednotlivé body z ktorých sa tieto trojuholníky skladajú. V našom prípade je teda jedna stena zóny reprezentovaná štyrmi bodmi a štyrmi trojuholníkmi. Štyrmi trojuholníkmi z toho dôvodu, že Unity využíva takzvaný `backface culling`, čo znamená, že zadné strany trojuholníkov nie sú vykresľované. To znamená, že ak je trojulník otočený na zlú stranu Unity ho nevykreslí (obrazok 5.1). Ak teda chceme, aby boli steny zóny viditeľné z oboch strán, musíme definovať každý trojuholník dva krát. Jeden trojuholník, ktorý bude otočený do vnútra zóny a druhý otočený na vonkajšiu časť zóny. O celý tento proces generovania sa stará metóda `GenerateZone`. Tá najprv vypočíta pozície jednotlivých bodov, ktoré budú reprezentovať vykreslené trojuholníky. Keďže terén mapy nie je rovina, môže sa stať, že stena zóny nebude vykreslená presne horizontálne, alebo že spodná časť bude akoby levitovať. Aby sme sa tomuto vyhli, bolo treba nájsť najnižší a najvýšší bod zóny a ich výšky nastaviť všetkým spodným a vrchným bodom vykresľovaných trojuholníkov. Kvôli tomu, aby bolo možné pilota správne navigovať do zóny, musíme okrem stien vygenerovať aj objekt, ktorý bude reprezentovať stred zóny. O nájdenie stredu sa stará metóda `GenerateMiddlePoint`. Nájdenie stredu medzi bodmi je pomerne jednoduché, stačí vypočítať priemer vektorov pozícií jednotlivých bodov.



Obrázek 5.1: Textúra zóny reprezentujúcej Purkyňovi koleje **vľavo:** Je zapnutý backface culling, vnútorné steny nie sú vykreslené **vpravo:** Backface culling je vypnutý, vnútorné steny sú vykreslené.

5.3 Beh misie

Poslednou úlohou, ktorú bolo potrebné vyriešiť, je implementácia mechanizmu, ktorý bude riadiť celý chod misie. Tento mechanizmus sa musí postarať o to aby:

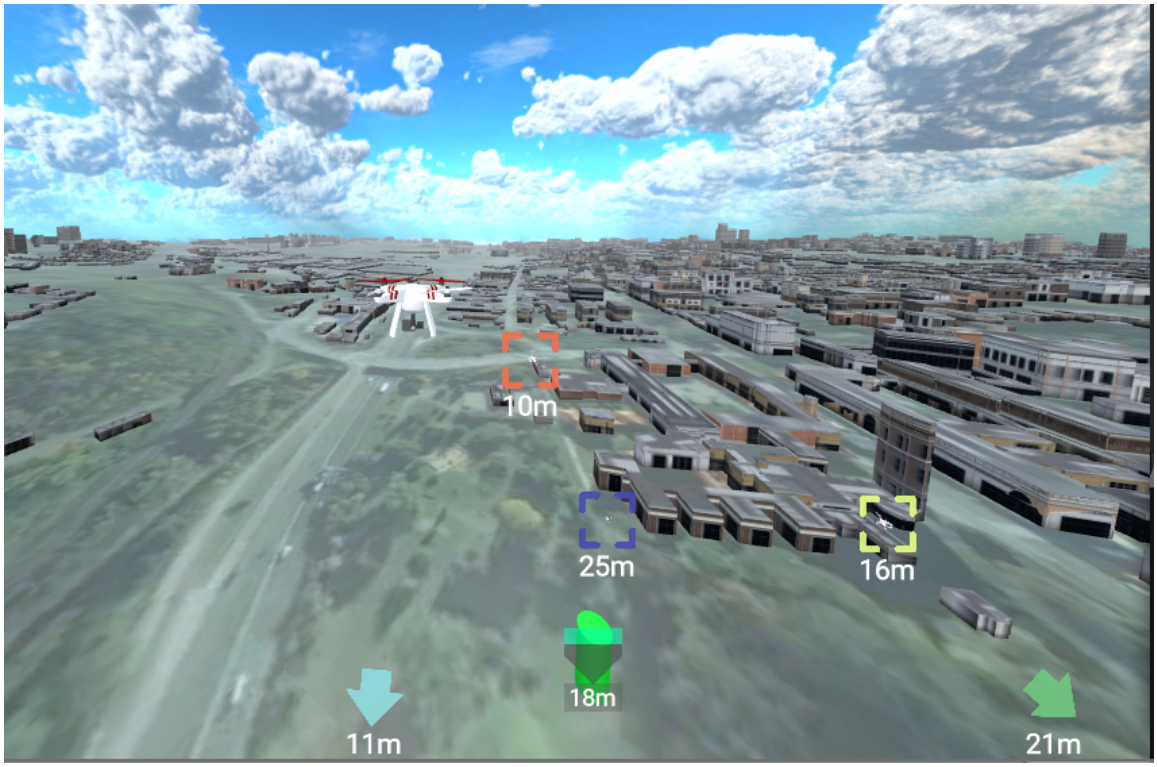
- Každý dron smeroval ku svojmu aktuálnemu checkpointu
- Checkpoint bol aktivovaný len v prípade, že ho splnili všetky drony, ktoré mu boli pridelené.
- Ak je checkpoint splnený, každému dronu musí byť správne pridelený nový checkpoint

Z toho vyplýva, že potrebujeme v každom momente poznať polohu všetkých dronov a zároveň polohu im priradených checkpointov. Ako som už spomenul v predchádzajúcej sekcií, metóda ktorá prevádza JSON na C# objekt umožňuje do jednotlivých tried pridávať aj parametre ktoré JSON nedefinuje. To sa ukázalo ako veľká výhoda práve pri riešení tohto problému. Do triedy, ktorá reprezentovala jednotlivé drony, som tak mohol pridať referenciu na ich polohu v scéne a zároveň usporiadaný zoznam checkpointov, ku ktorým sa majú počas misie dostať. Mechanizmus má tak v každom momente prehľad o tom, kde sa jednotlivé drony nachádzajú a zároveň to, ku ktorým checkpointom smerujú. Ak je potom checkpoint aktivovaný, stačí u každého dronu, ktorý sa na ňom podieľa, odstrániť prvý prvok z jeho zoznamu. Misia tak definitívne končí v momente, kedy má každý dron prázdny zoznam checkpointov. Teraz už stačí len vyriešiť to, kedy majú byť jednotlivé checkpointy aktivované. Keďže trieda, ktorá reprezentuje checkpoint, už obsahuje zoznam dronov, ktoré mu boli pridelené, môžme aplikovať rovnaký proces a checkpoint budeme považovať za splnený v momente, kedy bude jeho zoznam priradených dronov prázdny.

5.4 Navigácia

O navigáciu pilota k jednotlivým prvkom misie sa stará trieda `IconManager`. Pri štarte aplikácie sa vytvorí pre každý dron v scéne 2D ikona, ktorá je umiestnená na hlavné plátno aplikácie. Keďže každá ikona je inštancia toho istého prefabu, je potrebné, aby boli farebne odlíšené. Každý dron má teda pridelenú farbu, ktorá ho reprezentuje. Ikona zároveň obsahuje informáciu o tom, ako ďaleko sa jednotlivé drony nachádzajú od toho užívateľovho. Pred tým, ako však umiestnime tieto ikony na plátno, musíme sa ubezpečiť, že jednotlivé drony už boli umiestnené do scény. Z tohto dôvodu nie je vytváranie v metóde `Start`, ale v statickej metóde `DroneAdded`, ktorá je volaná triedov `MissionHandler` pri generovaní dronov. Keďže waypoint je v jednej chvíli aktivovaný vždy len jeden, jeho ikona je už umiestnená na plátne. O prevod 3D súradníc konkrétneho objektu na 2D súradnice ikony na plátne sa stará metóda `WorldToScreenPoint`. Táto metóda je implementovaná priamo v Unity v komponente kamery. Tento jednoduchý prevod však nie je dostačujúci. Prvým problémom je, že ak sa nachádza dron mimo zorného poľa kamery, jeho ikona nebude na obrazovke vidieť. Tento problém rieši metóda `Clamp`, ktorá obmedzí polohu ikony do zadaného intervalu. Súradnice plátna v Unity začínajú v nule a ich maximálna hodnota je daná výškou a šírkou plátna, ktoré sú dané premennými `Screen.height` a `Screen.width`. Do zvoleného intervalu musíme ale započítať aj dĺžku a širku ikony, pretože poloha objektov sa počíta od ich stredu a z ikony by tak na kraji obrazovky bola viditeľná len polovica. Druhým problémom je, že ak prevádzame 3D súradnice na 2D, zanedbávame tretiu súradnicu, z čoho vyplýva, že jednej 2D súradnici odpovedajú dve 3D súradnice, jedna pred a druhá

za kamerou. Ikona by tak bola viditeľná aj v prípade, že sa dron nachádza za nami. Preto musíme ikonu zobraziť v strede plátna len v prípade, že sa dron nachádza v zornom poli, na čo slúži metóda `Dot`. Ak dron nie je v zornom poli, jeho ikona je umiestnená na kraji obrazovky a zmení svoju podobu na šípku, aby mal pilot prehľad o tom, ktorým smerom sa sledovaný dron, alebo waypoint nachádza (obrázok 5.2). Ikona dronu je plne viditeľná od vzdialenosti 50 metrov a do vzdialenosti 100 metrov sa postupne zmenšuje, dokým nie je viditeľná vôbec. Ak by tak bolo v scéne veľa dronov, pilot nebude mať plnú obrazovku ikon, ale bude vidieť len tie drony, ktoré sú dostatočne blízko jeho dronu.



Obrázek 5.2: Ikony dronov(farebné hrany) a waypointu(tyrkysový trojuholník). Ikony dronov, ktoré nie sú v zornom poli kamery sú zmenené na šípky na okraji obrazovky.

5.5 Zobrazenie nebezpečenstva

O spracovanie informácií o blížiacom sa nebezpečenstve sa stará trieda `RayCastHandler`. Na získanie informácie o tom, či sa nachádza prekážka v okolí dronu, využíva táto trieda takzvané `RayCasty`. `RayCast` je neviditeľný lúč, ktorý dokáže zistiť či sa v jeho smere nachádza objekt a zároveň má informáciu o tom, ako ďaleko sa tento objekt nachádza od definovaného začiatku lúču. Metóda používa špeciálnu verziu `RayCastu` zvanú `SphereCast`, ktorá namiesto jednoduchého lúču používa lúč v tvare gule. Tým, že je požitý guľový lúč môžme detekovať budovu aj v prípade, že sa dron nachádza na jej okraji. Použitých je 10 lúčov, 2 ktoré sa starajú o detekciu nebezpečenstva nad a pod dronom a zvyšných 8 detekuje prekážky okolo dronu v horizontálnej rovine. Konkrétny prvok, ktorý zobrazuje nebezpečenstvo okolo dronu, je v umiestnený priamo v 3D scéne a tvorí osemuholník okolo dronu. Nebezpečenstvo z vrchu a zospodu je zobrazené na krajných častiach plátna (obrázok 5.3). Intenzita prvku má 3 úrovne. Úroveň závisí od vzdialenosti prekážky ku dronu. Prvá

úroveň je aktivovaná, ak je dron vo vzdialenosti menšej ako 10 metrov od prekážky, druhá, ak je menej ako 5 a tretia, ak je menej ako 3 metre od prekážky.

Obrázek 5.3: Zobrazenie nebezpečenstva okolo dronu pomocou dvojice 3D indikátoru (pred a naľavo od dronu) a indikátoru na okraji obrazovky (nebezpečenstvo zo spodu).

Aby bol objekt lúčom definovaný, musí obsahovať komponent `Collider`. Mapbox umožňuje v základnom nastavení pridať `Collider` iba terénu. Pridanie collideru budovám bolo treba naprogramovať ručne a stará sa o to metóda `AddMeshCollider` triedy `MapController`. Táto metóda je nastavená tak, aby sa vykonala až sekundu po spustení aplikácie. To je z dôvodu, že Mapboxu chvíľu trvá, kým načíta všetky budovy do scény. Zmeniť postupnosť vykonávania skriptov však nemôžme, pretože trieda `MapConroller` sa stará o počiatočné nastavenie parametrov mapy.

5.6 Užívateľské rozhranie

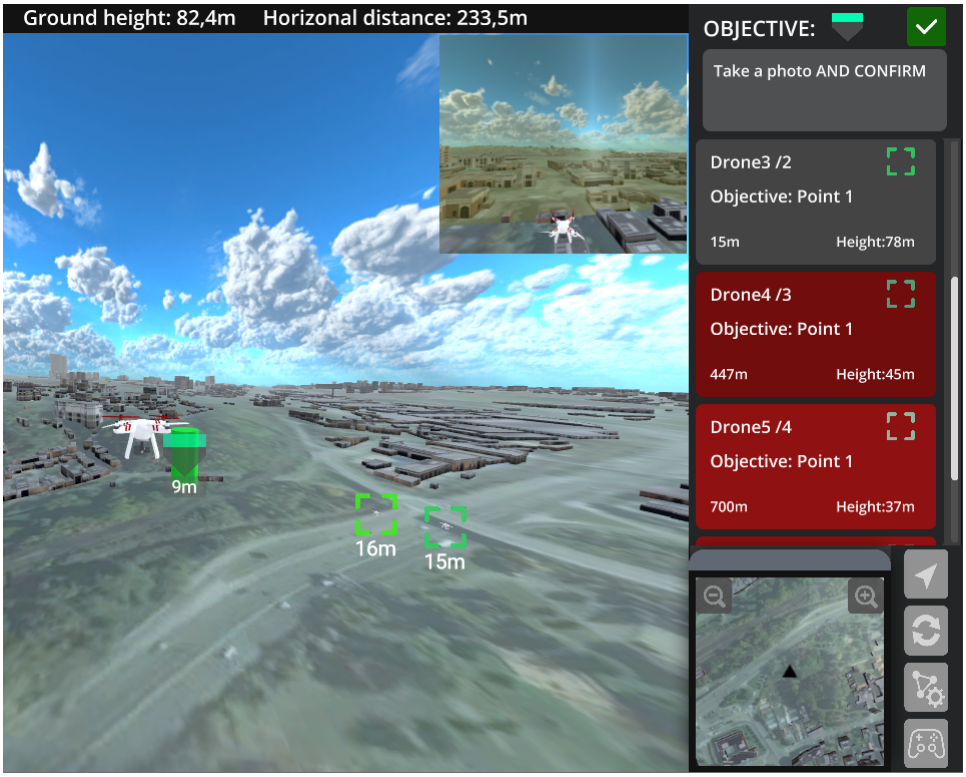
Užívateľské rozhranie aplikácie sa skladá zo štyroch základných častí. Z minimapy, bočného panelu s informáciami o misií, panelu s nastaveniami a tlačidiel na ovládanie kamery.

Minimapa

Základným prvkom minimapy je kamera `MinimapCamera`, ktorá sníma scénu z vtáčej perspektívy. O pohyb a rotáciu kamery spoločne s dronom sa stará skript `MinimapController`. Na to, aby bolo možné na plátne zobraziť pohľad z druhej kamery, je potrebné vytvoriť špeciálny typ textúry zvaný `RenderTexture`, na ktorú sa bude tento pohľad premietať. Na plátno potom môžme umiestniť 2D objekt `RawImage`, ktorý dokáže na rozdiel od obyčajného `Image` zobrazovať dynamické textúry. Minimapa má 2 tlačidlá: jedno na priblíženie a druhé na oddialenie pohľadu. U kamery je možné nastaviť vlastnosť `Culling Mask`, vďaka ktorej je možné zvoliť, ktoré vrstvy bude kamera vidieť. Kamera minimapy je nastavená tak, aby zaznamenávala iba terén, budovy a zóny. Bez tohto nastavenia sa stávalo, že dron bol príliš blízko kamery a terén spolu s budovami nebol cez drona vidieť. V základnom stave je minimapa umiestnená v pravom dolnom rohu obrazovky, uživateľ však môže dynamicky meniť polohu minimapy počas behu aplikácie.

Informačný panel

Na rýchle získanie informácií o stave misie slúži panel umiestnený na pravej strane obrazovky (obrázok 5.4). Hlavnými časťami tohto panelu sú pole zobrazujúce aktuálnu úlohu spolu s tlačidlom na prípadné potvrdenie splnenia úlohy a skrolovateľný zoznam dronov podieľajúcich sa na misií. Každú položku tohto zoznamu tvorí názov dronu, jeho aktuálna úloha v misií, výška nad zemou, vzdialenosť od užívateľovho dronu a pridelená farba. V niektorých situáciach je však vhodné, aby mal pilot prehľad aj o tom, kde presne sa jednotlivé drony nachádzajú. Z tohto dôvodu, každá z týchto položiek navyše slúži ako tlačidlo pre zobrazenie pohľadu z kamery zvoleného dronu. Tento pohľad je dostupný v pravom hornom rohu obrazovky a na jeho vytvorenie bol použitý rovnaký postup ako pri minimape. Zoznam navyše upozorňuje pilota, na ktoré drony musí počkať pred pokračovaním na ďalší checkpoint. Položky týchto dronov sú zobrazené v červenej farbe, pokiaľ sa nedostanú k užívateľovmu checkpointu. Unity bohužiaľ neobsahuje vstavané rozhranie pre zobrazovanie takýchto zoznamov. Jednotlivé položky zoznamu sú teda, rovnako ako tomu bolo u dronov a waypointov, len instancované, dopredu vytvorené objekty. O vytvorenie týchto položiek a zobrazenie správnych informácií o jednotlivých dronoch sa stará trieda `Drones`. Nakoľko informácií, ktoré užívateľ potrebuje vedieť o misií, je pomerne veľa, tento panel zaberá až jednu pätinu šírky obrazu. Z tohto dôvodu je obraz z kamery zmenšený na štyri pätiny a posunutý do ľavej časti obrazovky. O túto transformáciu sa stará trieda `CameraScaler`, ktorá upravuje parameter `rect` hlavnej trojice kamier. Poslednou súčasťou bočného panelu je malý panel s tlačidlami na pripojenie k ROSu a umiestnenie pôvodných navigačných prvkov.



Obrázek 5.4: Bočný panel zobrazujúci informácie o misií.

Panel s nastaveniami

Najdôležitejšou súčasťou užívateľského rozhrania je panel s nastaveniami umiestnený v ľavej časti obrazovky. Pri spustení aplikácie tento panel nie je viditeľný, užívateľ ho môže zapnúť kliknutím na tlačidlo v ľavom hornom rohu obrazovky. V pôvodnej verzií aplikácia obsahovala 2 scény, začiatočnú scénu s užívateľskými nastaveniami a hlavnú scénu, kde užívateľ ovláda drona. Problémom bolo, že medzi týmito scénami sa nedalo spätne prepínať, pilot tak nemohol meniť počas behu parametre, ako url ROSu, veľkosť mapy, alebo parametre kamery. Úlohou tohto panelu tak bolo spojiť tieto dve scény do jednej a uľahčiť tak užívateľovi prácu s aplikáciou. Pri prvom spustení aplikácie sú všetky parametre nastavené na prednastavené hodnoty triedou `SetupPlayerPrefs`. Ak potom užívateľ zmení tieto parametre, pri ďalšom spustení budú tieto zmeny automaticky aplikované. Na to slúži Unity trieda `PlayerPrefs`, ktorá umožňuje ukladať užívateľské nastavenia medzi spusteniami aplikácie. Prvou skupinou nastavení, ktoré tento panel obsahuje, sú nastavenia spojené s pripojením k ROSu. Konkrétne sú to url RosBridge serveru a ROS topic s videom z kamery. Na spracovanie užívateľského vstupu slúži v Unity prvok `InputField`. Ten umožňuje nastaviť metódu pre udalosť `On End Edit`, ktorá nastane v momente, keď užívateľ prestane upravovať zadaný text. Parameter je tak nastavený hneď po tom, čo ho užívateľ zadá. Na zmenu generovanej mapy slúžia parametre `MapCenter` (stred mapy) a `MapSize` (veľkosť mapy). Na zmenu veľkosti mapy je použitý slider, ktorý umožňuje nastaviť veľkosť mapy od 1 po 10. Keďže pred prvým spustení nemá užívateľ možnosť ovplyvniť, kde sa bude nachádzať stred mapy, načítanie misie (5.1) neprebieha pri štarte aplikácie, ale po stlačení tlačidla `LoadMission`. Nenastane tak problém, kedy by boli navigačné prvky misie umiestnené mimo vygenerovanej mapy. Slidery sú použité aj na nastavenie parametrov plátna s

videom, ktorými sú výška a šírka videa, vzdialenosť plátna od kamery a veľkosť zorného poľa. Pilot si tak môže počas letu prispôsobiť plátno tak, aby záznam z videa čo najlepšie zapadol do virtuálnej scény. Keďže pri nastavení niektorých parametrov používa užívateľ klávesnicu bolo potrebné deaktivovať všetky funkcie, ktoré využívali vstup z klávesnice a následne ich zakomponovať do užívateľského rozhrania. Tým zároveň zaistíme, že aplikácia bude plne funkčná aj na dotykových zariadeniach, čo bolo tiež jedným z cieľov tejto práce. Sú to funkcie ako vypnutie budov, zmena textúry mapy, alebo rotácia a zoom kamery. Väčšina týchto funkcií bola teda umiestnená do panelu s nastaveniami. Rotácia a zoom sú dôležité pri ovládaní dronu, preto by mali byť užívateľovi vždy k dispozícií. Všetky funkcie pracujúce s kamerou, tak boli umiestnené do jednoduchého ovládacieho prvku, ktorého polohu na obrazovke môže užívateľ ľubovoľne meniť počas behu aplikácie. Veľkosť tohto prvku je zhodná s veľkosťou minimapy, užívateľ tak môže vymeniť polohy týchto dvoch prvkov a v prípade, že používa napríklad tablet, môže aplikáciu ovládať aj ľavou, aj pravou rukou.

5.7 Manuál k použitiu

V tejto kapitole súhrnne zhrniem znalosti z predchádzajúcich častí práce a doplním ich o informácie potrebné pre prácu s aplikáciou. Pri prvom použití majú všetky nastaviteľné parametre aplikácie predvolenú hodnotu. Na ich zmenu je k dispozícii panel s nastaveniami, ktorý sa užívateľov sprístupní po kliknutí na tlačidlo v ľavom hornom kraji obrazovky. Prvým parametrom, ktorý je potrebné nastaviť je stred vygenerovanej mapy v poli `MapCenter`. Veľkosť mapy je možné nastaviť sliderom `Map Size` a to od 1200x1200 metrov (úroveň 1) po 8400x8400 metrov (úroveň 10). Terén mapy je možné nastaviť tlačidlom `Change Map Source`. Na výber je satelitný snímok, alebo klasická mapa. Pre stiahnutie mapových dát vyžaduje aplikácia pripojenie k internetu Po tom, čo užívateľ nastaví tieto parametre, môže načítať misiu tlačidlom `Load Mission`. Misiu môže užívateľ definovať v súbore `Drone3rdPerson_Data\StreamingAssets\mission.json` (formát je popisaný v časti 5.1). Pre pripojenie k dronu je potrebné správne vyplniť pole s URL RosBridge serveru, ktorý beží na testovacom drone `RosBridge IP` a topic, ktorý obsahuje video z kamery `Video Topic`. U videa je potom možné nastaviť rozlíšenie a zorné pole kamery (FOV). Testovací dron niekedy zle určuje nadmorskú výšku, z tohto dôvodu je možné po pripojení k dronu nastaviť dron do správnej výšky tlačidlom `Set Altitude Offset`. Ďalšie parametre potom umožňujú zapínať a vypínať niektoré funkcie aplikácie. Je možné zapnúť virtuálne plátno, zobrazenie budov a zobrazenie pôvodných navigačných prvkov ako 3D navigačné šípky, alebo pôvodné zóny, ktoré môže užívateľ definovať za behu aplikácie.

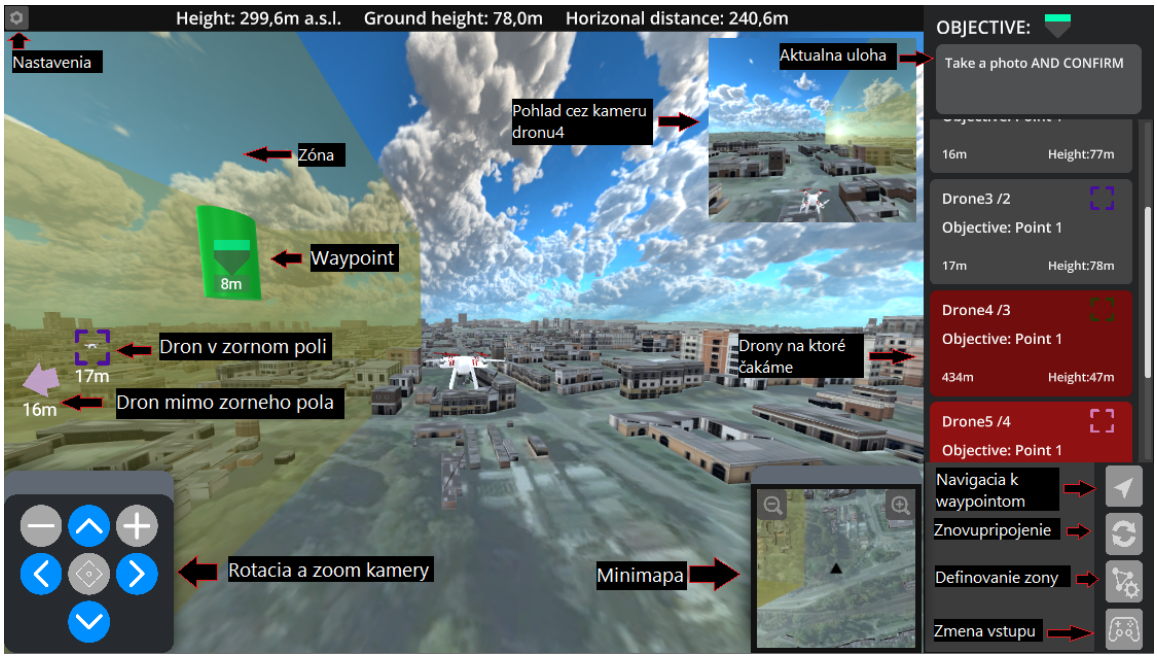
Funkcie, ktoré užívateľ potrebuje využívať za behu aplikácie sú dostupné pomocou tlačidiel v pravom dolnom rohu obrazovky. Spodné tlačidlo slúži na zmenu zdroju letových dát. Dostupné sú 3 zdroje: simulátor (ikona s gamepadom), živé dáta z dronu (ikona ROS) a náhodné dáta (ikona RAND). Ďalšie tlačidlo slúži na znovupripojenie k dronu v prípade výpadku, alebo zmeny IP adresy RosBridge. Tlačidlo s šípkou umožňuje užívateľovi definovať waypointy za behu aplikácie a to buď na miesto kde sa dron nachádza, alebo na miesto kam užívateľ klikne. Posledné tlačidlo slúži na editáciu zóny, ktorú môže užívateľ definovať za behu aplikácie. Je možné umiestniť len jednu takúto zónu a jej hranice je možné definovať kliknutím. Editácia sa vypne po opätovnom stlačení tohto tlačidla.

Na rotáciu kamery slúži skupina tlačidiel v ľavom dolnom rohu obrazovky. Polohu týchto tlačidiel je možné ľubovoľne meniť potiahnutím po obrazovke.

V pravej strane obrazovky sa nachádza informačný panel obsahujúci popis aktuálnej úlohy, zoznam dronov podielajúcich sa na misií a minimapu. Na prvky zoznamu dronov

sa dá kliknúť, čím sa zobrazí pohľad cez kameru zvoleného dronu. Ak je prvok červený, znamená to, že konkrétny dron sa podieľa na rovnakej úlohe ako užívateľ a ešte sa nedostal k zvolenému waypointu. Minimapu je možné približovať a odďalovať a rovnako ako prvok s rotáciou je jej polohu možné meniť posunutím po obrazovke.

Rozloženie jednotlivých prvkov rozhrania je ukázané na obrázku 5.5.



Obrázek 5.5: Rozloženie prvkov užívateľského rozhrania.

Kapitola 6

Overovanie funkčnosti navigačných prvkov

Overovanie funkčnosti prebiehalo v dvoch módoch. V simulovanom móde, kedy bolo cieľom odladiť funkčnosť nových prvkov aplikácie a uistiť sa, že je navrhnuté užívateľské rozhranie pochopiteľné aj pre nového nezaškoleného užívateľa. Druhým módom bolo overovania na základe reálnych dátach z dronu, ktorého cieľom bolo uistiť sa, že komunikácia s dronom bezchybne funguje a to, ako vplýva obraz z kamery na zabudované navigačné prvky.

6.1 Overovanie v simulátore

Overovanie funkčnosti navigačných prvkov prebiehalo z väčšej časti v zabudovanom simulátore. Aby sa zistilo, či navigačné prvky fungujú a skutočne pomáhajú s orientáciou v scéne v každej situácií, boli vytvorené 3 skúšobné scenáre. Toto overovanie prebiehalo počas vývoja, hlavne formou konzultácií s vedúcim práce a výskumníkom zo skupiny Robo@FIT.

Prvým skúšobným scenárom bola jednoduchá misia pre jeden dron. V tejto misií sa mal pilot dostať do dvoch zón, ktoré predstavovali Purkyňovi koleje a areál FITu. Po tom, čo sa užívateľ dostal do jednotlivých zón, mal za úlohu nafotiť areál z niekoľkých lokácií. Práve v tejto situácií sa ukázalo, že je nutné pridať do misie waypoint, ktorý musí užívateľ ručne potvrdiť. Základný waypoint sa totižto aktivoval hneď, ako sa k nemu užívateľ s dronom dostal a užívateľ tak stratil prehľad o tom, odkiaľ mal fotku zhotoviť. Tento scenár sme využívali primárne v počiatočnej fázy vývoja, kedy ešte nebola implementovaná podpora pre vizualizáciu viacerých dronov. V tejto fáze bolo primárnym cieľom vyladiť funkčnosť základných navigačných prvkov, ktorými sú waypointy a zóna. Výsledkom boli implementačné úpravy popísané v sekcii 5.2 a pridanie už spomínaného waypointu s potvrdením.

Druhý scenár obsahoval rovnaké prvky ako ten prvý, rozdielom bolo, že do scény bol umiestnený ďalší dron ovládaný simulátorom. Tento scenár tak slúžil na overenie funkčnosti vzájomnej kooperácie viacerých dronov. Práve pri tomto scenári sa ukázalo, že je nutné vizualizovať polohu jednotlivých objektov pomocou 2D ikony. V predchádzajúcom scenári bola na navigáciu k jednotlivým waypointom použitá 3D šípka z pôvodnej verzie aplikácie [11]. Po tom čo bol do scény pridaný ďalší dron však pochopiteľne táto šípka už nebola dostačujúca, pretože bolo potrebné zobraziť polohu viac ako jedného objektu.

Posledný scenár bol omnoho komplexnejší než predchádzajúce dva. Obsahoval veľké množstvo checkpointov a hneď šesť spolupracujúcich dronov (5 z nich ovládaných simulátorom, 1 užívateľom). Hlavným cieľom tohto scenáru bolo zistiť, či užívateľ nestratí orientáciu

pri väčšom počte objektov v scéne. Ukázalo sa, že nie je vhodné vždy zobrazovať ikony všetkých dronov, niekedy sa totižto stávalo, že navzájom splývali a bolo ťažké určiť kde sa ktorý dron nachádza. Z tohto dôvodu bola nastavená viditeľnosť týchto ikon na vzdialenosť 50 metrov a do bočného panelu bola pridaná možnosť náhľadu cez kameru druhého dronu, aby užívateľ mohol vidieť aj polohu dronov, ktorých ikony nie sú zobrazené.

6.2 Overovanie s dátami z dronu

Keďže aplikácia nemá slúžiť len ako simulátor, ale ako plnohodnotná aplikácia na ovládanie dronu, bolo potrebné funkčnosť aplikácie overiť aj na reálnych dátach z dronu. Keďže dron nebol vždy k dispozícií, bola vytvorená séria testovacích RosBag záznamov, ktoré obsahovali záznam videa spoločne s dátami o polohe a náklone dronu. Tieto záznamy následne bolo možné prehrať priamo v ROSe a simulovať tak reálny let z dronom. Na to aby dokázala aplikácia čítať dáta bolo potrebné spustiť RosBridge server, na ktorý sa aplikácia pripojila a následne spustiť RosBag. Tento proces bol pomerne náročný na hardware, pretože vývoj aplikácie prebiehal na platforme Windows a na spustenie ROSu bol potrebný operačný systém Linux, ktorý musel byť virtualizovaný. Z tohto dôvodu bol ROS spolu s RosBridge spúšťaný na druhom počítači, ktorý slúžil ako lokálny server. Hlavným cieľom tohto overovania bolo ubezpečiť sa, že všetky súčasti starajúce sa o získanie dát z ROSu fungujú správne a že navigačné prvky sú použiteľné aj pri použití plátna.

Ukázalo sa, že navigačné prvky celkom prirodzene zapadajú do obrazu na plátne. Problémom však bola náväznosť videozáznamu na virtuálnu scénu. Stávalo sa, že obraz z kamery bol v niektorých prípadoch posunutý vyššie a obraz tak nenadväzoval na budovy vo vygenerovanej mape. Ukázalo sa, že Mapbox zle prepočítava nadmorskú výšku dronu a dron tak bol posunutý vyššie ako by mal. Z tohto dôvodu bolo potrebné nastaviť offset na výšku dronu tak, aby sa jeho výška dronu v scéne zhodovala s tou v reálnom svete. Aby bol problém náväznosti vyriešení úplne, bolo ešte potrebné pridať do nastavení možnosť upravovať parametre kamery za behu aplikácie.

6.3 Ďalší vývoj

Ďalším cieľom vývoju aplikácie by mala byť vizualizácia viacerých reálnych dronov v scéne. Na fakulte je dostupný dron od spoločnosti DJI, preto by bolo vhodné vytvoriť program, ktorý by získaval letové dáta z DJI dronu a posielal ich na server, z ktorého by aplikácia dokázala tieto dáta čítať a na ich základe vizualizovať druhého drona. Využiteľnosť aplikácie by sa tak výrazne zlepšila, keďže drony od DJI sú najviac rozšírené. Ďalej by bolo vhodné viac využívať dáta z kamery dronu na možnú detekciu objektov reálneho sveta, ktoré by Unity dokázalo vykresliť. V závere overovania sa prišlo na niektoré drobné problémy, ktoré by takisto vyžadovali pozornosť. Prvým problémom je, že ak sa dron nemôže dostať k waypointu, misia nemôže pokračovať. Z tohto dôvodu by bolo vhodné, aby pilot mohol waypoint potvrdiť aj keď sa pri ňom nenachádza. Druhým problémom, ktorý sa nestihol vyriešiť je lepšie spracovanie metódy `AddMeshCollider` spomenutej v kapitole 5.5. Pridanie Colliderov na budovy je v tejto metóde nastavený na 1 sekundu od štartu aplikácie. To pri pomalšom hardvéri nemusí byť dostačujúce a mala by táto metóda realizovaná formou callbacku.

Kapitola 7

Záver

Cieľom tejto práce bolo navrhnúť navigačné prvky do už existujúcej aplikácie na ovládanie dronu, ktoré by pomohli užívateľovi s orientáciou pri pilotáži dronu.

Na začiatku vývoja som si naštudoval problematiku dronov, to ako sa tieto zariadenia ovládajú a aké najčastejšie problémy nastávajú pri ich pilotáži. Po získaní základných znalostí o problematike som prešiel na štúdium základnej architektúry systému, na ktorom som pracoval. Vyskúšal som si používanie dronu s touto aplikáciu a identifikoval problémy a nedostatky, na ktoré som následne v mojej práci navrhol riešenie.

Navrhol som niekoľko riešení, ktorých funkčnosť som počas vývoja overoval a upravoval tak, aby boli spracované čo najkvalitnejšie a boli jednoduché na použitie aj pre bežného užívateľa. Vytvoril som rozhranie, vďaka ktorému si užívateľ môže v aplikácií vizualizovať vlastnú misiu, s čím je spojená možnosť zobrazenia viacerých dronov. Táto možnosť je kľúčová hlavne pri špeciálnych misiách, ktoré s dronmi realizujú špeciálne jednotky ako polícia, alebo záchranári. K jednotlivým prvkom misie som vytvoril HUD slúžiaci na navigáciu a panel zobrazujúci všetky informácie o misií v reálnom čase. Keďže pri zložitejších úkonoch s dronom je veľmi dôležité, aby mal pilot jasný prehľad o tom, čo sa v okolí dronu nachádza, bol do aplikácie začlenený mechanizmus, ktorý ukazuje smer a vzdialenosť najbližších prekážok okolo dronu, spolu s minimapou. V závere vývoja som ešte zapracoval na vylepšení užívateľského rozhrania a umožnení použitia aplikácie na dotykových zariadeniach.

Ďalší vývoj aplikácie by sa mal teraz uberať k architektúre klient-server, ktorá by umožnila akýsi multiplayer medzi viacerými dronmi (nie len simulovaný, ako je tomu teraz). Zároveň by sa dali využiť dáta z kamery dronu na detekciu niektorých prvkov reálneho sveta, ktoré by následne Unity dokázalo vykresliť do virtuálnej scény.

Aplikácia má skutočne veľký potenciál a verím, že jej ďalší vývoj prinesie zaujímavé riešenia v oblasti vývoja aplikácií na ovládanie dronu.

Literatura

[1] *01 – ROS – Robot Operating System – úvod do ROS*. Dostupné z: https://www.dailyautomation.sk/uvod-do-ros/.

[2] *Dokumentace ROS wiki*. [Online]. Dostupné z: http://wiki.ros.org/.

[3] *MAPBOX DOCUMENTATION*. Dostupné z: https://docs.mapbox.com/unity/maps/overview/.

[4] *RADIO TRANSMITTER MODES*. Dostupné z: https://bmfa.org/Info/Getting-Started/Getting-Started-in-R-C/Radio-Modes.

[5] *Unity Documentation*. [Online]. Dostupné z: https://docs.unity3d.com/Manual/.

[6] Cir, I. 328: Unmanned Aircraft Systems (UAS). *International Civil Aviation Organization*. 2011.

[7] Funk, M. Human-Drone Interaction: Let's Get Ready for Flying User Interfaces! *Interactions*. New York, NY, USA: Association for Computing Machinery. duben 2018, roč. 25, č. 3, s. 78–81. Dostupné z: https://doi.org/10.1145/3194317. ISSN 1072-5520.

[8] JURKŮ, M. *Studie pro využití bezpilotních letadel-dron v komerčním prostředí*. 2016. Bakalářská práce. České vysoké učení technické v Praze. Vypočetní a informační centrum. Dostupné z: https://dspace.cvut.cz/handle/10467/66078.

[9] LASNOVSKÝ, L. *Sociální funkce dronů a jejich využití ve veřejném prostoru [online]*. 2016 [cit. 2020-07-04]. Bakalářská práce. Masarykova univerzita, Filozofická fakulta, Brno. Dostupné z: https://theses.cz/id/37414i/.

[10] Milgram, P., Takemura, H., Utsumi, A. a Kishino, F. Augmented reality: A class of displays on the reality-virtuality continuum. In: International Society for Optics and Photonics. *Telemanipulator and telepresence technologies*. 1995, s. 282–292.

[11] Sedlmajer, K. *Uživatelské rozhraní pro řízení dronu s využitím rozšířené virtuality*. Brno, CZ, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/16730/.

[12] Sedlmajer, K., Bambušek, D. a Beran, V. Effective Remote Drone Control Using Augmented Virtuality. In: *Proceedings of the 3rd International Conference on Computer-Human Interaction Research and Applications 2019*. SciTePress - Science and Technology Publications, 2019, s. 177–182. Dostupné z: https://www.fit.vut.cz/research/publication/12006. ISBN 978-989-758-376-6.

[13] Stephenson, J. *A UX Analysis of First-Person Shooter Damage Indicators.* Dostupné z: https://medium.com/@jasper.stephenson/a-ux-analysis-of-first-person-shooter-damage-indicators-59ac9d41caf8.