
**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**


# UŽIVATELSKÉ ROZHRANÍ PRO ŘÍZENÍ DRONU S VYUŽITÍM ROZŠÍŘENÉ VIRTUALITY

**USER INTERFACE FOR DRONE CONTROL USING AUGMENTED VIRTUALITY**

**DIPLOMOVÁ PRÁCE**
**MASTER'S THESIS**


**AUTOR PRÁCE** **Bc. KAMIL SEDLMAJER**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Ing. VÍTĚZSLAV BERAN, PhD.**
**SUPERVISOR**



**BRNO 2019**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

# Zadání diplomové práce


Student: **Sedlmajer Kamil, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Uživatelské rozhraní pro řízení dronu s využitím rozšířené virtuality**
**User interface for drone control using augmented virtuality**
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte aktuální technologie řízení dronů a vizualizaci 3D virtuálních scén. Seznamte se s možnostmi on-line snímání reálných dat z dronu (poloha, video, hloubková data, letová data apod.) a dostupnými off-line datovými podklady třetích stran (výškové mapy, 3D modely budov apod.).
2. Navrhněte vizualizační interaktivní aplikaci, která bude integrovat on-line data z dronu a off-line data v jedné 3D scéně. Rozšiřte o UI prvky pro efektivnější řízení: navigační prvky, bezpečnostní prvky, ruční pozicování kamery apod.
3. Implementujte navrženou funkci 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:

- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision*, CL-Engineering, ISBN-13: 978-0495082521, 2007.
- 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.

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2, 3 a částečně 4.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Beran Vítězslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 6. května 2019

# Abstrakt

Tato práce hodnotí stávající možnosti ovládání dronů, jejich úskalí a navrhuje možná řešení pro efektivnější a snadnější řízení dronů. Výsledný systém je založen na pohledu třetí osoby a technologii rozšířená virtualita, kdy jsou do virtuálního 3D modelu prostředí integrována reálná data z dronu (video-stream, lokalizační informace). Model prostředí je vytvořen s využitím volně dostupných dat. Aplikace pilotovi nabízí prostředky pro snadnější orientaci v prostředí, navigaci k cílovým místům a možnost v průběhu plánování mise definovat oblasti s různým potenciálním bezpečnostním rizikem, které budou v průběhu mise využívány k navigaci v těchto zónách a vizualizaci celkové situace ve virtuální scéně rozšířené o on-line reálná data.



# Abstract

The thesis evaluates the current possibilities and problems of drone control and suggests possible solutions. The aim is to control drones more efficiently and easily. The final system is based on third person view and Augmented Virtuality technology where real data from the drone (video-stream, localization information) has been integrated into the virtual 3D model of the surroundings. The model of the surroundings has been created using free data. The application provides the pilot with the means to navigate in the surroundings and to navigate to destinations. It also offers the possibility to define areas with various potential security risks during mission planning, which will be used to navigate in the mission zones, and to visualize the overall situation in the virtual scene extended with online real data.



# Klíčová slova

dron, drony, ovládání dronů, pohled třetí osoby, rozšířená virtualita, virtuální scéna, mapová data, online data z dronu, navigační prvky

# Keywords

drone, drones, drones control, third person view, augmented virtuality, virtual scene, map data, on-line data from drone, navigation elements


# Citace

SEDLMAJER, Kamil. *Uživatelské rozhraní pro řízení dronu s využitím rozšířené virtuality.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, PhD.

# Uživatelské rozhraní pro řízení dronu s využitím rozšířené virtuality

## Prohlášení

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

.....
Kamil Sedlmajer
20. května 2019


## Poděkování

Chtěl bych poděkovat vedoucímu diplomové práce, Ing. Vítězslavu Beranovi, Ph.D, za cenné rady a především podnětné konzultace plné nápadů a vizí, které jsou v této práci představeny. Dále pak dopravnímu pilotovi Vladimíru Mixovi a kolektivu pilotů Aeroklubu Vyškov za konzultaci a vysvětlení leteckých předpisů.

# Obsah

# Kapitola 1

# Úvod

Tato práce se zabývá návrhem nové vizualizační aplikace pro snadnější a efektivnější ovládání dronů s důrazem na zlepšení prostorové orientace pilota při řízení prostřednictvím video přenosu. Pilotáž dronu je totiž náročná disciplína a méně trénovaný pilot může snadno ztratit představu o jeho aktuální pozici a prostorovou orientaci. Ztráta orientace pak může vést ke zničení nebo ztrátě dronu, neúmyslnému vlétnutí do zakázané oblasti, poškození majetku na zemi a nebo v krajním případě i zranění osob.

Navržené řešení využívá **pohled třetí osoby** a **rozšířenou virtualitu** pro zobrazení 3D scény vytvořené s využitím veřejně dostupných mapových dat a do této scény přidává stream z kamery na dronu, data ze senzorů a další letové informace.

Cílem není snaha o překonání již existujících programů pro pozemní stanice (obvykle notebook nebo mobilní telefon), ale naopak vyzkoušet nové přístupy k této problematice, ze kterých by v případě, že se osvědčí, mohla v budoucnosti vzniknout aplikace určená pro reálné využití.

První kapitola se věnuje ovládání koptéry, řízení pohledem ze země a využití přenosu videa. Dále pak vysílačkami, letovými režimy a především způsobem zobrazení letových dat a streamu z kamery na pozemní stanici. Také jsou zde uvedeny stručné informace o konstrukci dronů, hardwaru a senzorech. Krátká podkapitola je pak věnována české legislativě a pravidlům létání s bezpilotními letouny s důrazem na možná rizika, která se mohou při provozu dronu objevit.

V hlavní části práce je rozebrán problém ztráty orientace během řízení dronu a navrženo několik řešení, jak zlepšit pilotovu prostorovou orientaci tak, aby vždy věděl, kde se dron nachází, jak je otočen, kterým směrem chce letět a jaké překážky se nacházejí v jeho okolí. Řešení využívající právě rozšířenou virtualitu pak dále rozvíjí.

Na základě tohoto návrhu vznikla vizualizační aplikace, která byla dále testována jak v simulátoru, tak s reálným dronem. Závěrečná kapitola tedy popisuje průběh testování, vlastnosti tohoto konceptu, zjištěně nedostatky, navrhuje možnosti jejich odstranění a nastiňuje směr dalšího vývoje aplikace.

# Kapitola 2

# Drony a jejich řízení

## 2.1 Způsoby řízení dronu

Pro řízení dronů se používají dva základní přístupy. Řízení pohledem ze země a řízení za pomoci videa. V obou případech pilot drží v ruce RC vysílačku a řídí multikoptéru pomocí dvou joysticků. Dále může mít k dispozici pozemní stanici (mobil, tablet, notebook), kde má zobrazeny informace o letu, stavu baterie, případně mapu s vyznačenou polohou dronu nebo video z kamery na koptéře. Moderní drony navíc umí část letu provádět v autonomním režimu.

### Řízení pohledem ze země

Při řízení pohledem ze země pilot může efektivně řídit pouze na takovou vzdálenost, na kterou bezpečně vidí, kde dron je a kam míří jeho příď. Většina pilotů je tak schopna tímto způsobem letoun ovládat pouze na několik desítek nebo maximálně stovek metrů. Pokud pilot ztratí představu o tom, kam míří příď letounu, může dojít velmi snadno vlivem nesprávného zásahu do řízení ke střetu s překážkou. Moderní drony však umožňují přepnout režim letu do tzv. režimu „Return to home“ a dron se sám vrátí na místo startu v přednastavené výšce. Když je letoun dostatečně blízko, pilot může opět převzít kontrolu a v letu bezpečně pokračovat.

### Řízení přes video

Při tomto způsobu řízení pilot na letoun vůbec nekouká. Sleduje pouze obraz z kamery a řídí podle něj. V tomto případě je maximální vzdálenost letu omezena především dosahem signálu. V případě, že dron signál ztratí, opět nastupuje autonomní režim, ve kterém se dron vrací na místo startu. Pilot při tomto způsobu řízení přesně vidí, kam míří příď letounu, pokud však delší dobu nevidí žádné jemu známé orientační body, může dojít ke ztrátě orientace a pilot ztrácí čas rozhlížením po okolí ve snaze zjistit, kde se dron nachází, případně se musí opět spolehnout na autonomní návrat. Problému ztráty orientace se podrobněji věnuje kapitola 3. Při tomto způsobu řízení pilot může s dronem v malé výšce letět pouze dopředu. Do stran a za sebe totiž vůbec nevidí a případnou překážku by tedy vůbec neviděl.

Tyto dva způsoby řízení je samozřejmě možné také kombinovat, pokud ale pilot řídí pomocí videa, není často lehké dron rychle najít na obloze. To se samozřejmě netýká případu, kdy je dron v bezprostřední blízkosti pilota a nebo alespoň natolik blízko, že ho lze snadno najít podle sluchu.

## 2.2 Letové režimy

Letové režimy v této kapitole jsou popisovány co možná nejobecněji – tedy nezávisle na řídícím softwaru. Z toho důvodu je u některých uvedeno několik různých názvů. Ne všechny režimy jsou dostupné na všech platformách a mohou se také mírně lišit. Zde uvedené názvy odpovídají především platformám ArduCopter [4] a INAV [2]. U navigačních režimů navíc také záleží na množství připojených senzorů. Bez těch by totiž tyto režimy vůbec nemohly fungovat.

Vzhledem k tomu, že způsob ovládání, letové režimy a veškeré elektronické vybavení je velmi podobné i v dalších typech bezpilotních letounů jako jsou letadla, samokřídla a různé typy VTOL (letadel s kolmým startem), není nutné brát informace obsažené v příštích kapitolách pouze v kontextu multikoptér.

Režimy jsou popisovány od těch s nejmenší asistencí autopilota po ty, kde už autopilot vykonává většinu práce sám.

**Manual** — Režim bez jakékoliv stabilizace — na multikoptérách se prakticky nepoužívá. Používá se však na letadlech a modelech vrtulníků.

**Acro** — Režim pro akrobatické manévry, vyžaduje extrémně náročnou pilotáž. Při vycentrovaných joysticích drží UAV (bezpilotní letoun) svůj aktuální náklon. Stabilizace působí jen proti změnám způsobeným externími vlivy (vítr).

**Rattitude/Horizon** — UAV drží svůj náklon při vycentrovaných joysticích, dovoluje však neomezené náklony ve všech osách. Po opětovném vycentrování řízení se UAV opět sám vyrovná.

**Stabilized/Angle** — Dron drží nulový náklon při vycentrovaných joysticích. I tak ale může samovolně někam pomalu cestovat, protože pozice není nijak korigována pomocí senzorů polohy (např. GPS). Při vychýlení páky řízení se letoun nakloní a začne pohybovat v daném směru, náklon však nikdy nepřesáhne určitou mez (např. 30°).

**Altitude/Althold** — Totéž jako režim stabilized, jen se UAV snaží udržovat při vycentrované páce plynu kromě náklonu také výšku.

**Position/Poshold** — Totéž jako režim althold, UAV zde ale udržuje navíc pozici s využitím senzorů polohy. Při vychýlení řízení se začne pohybovat daným směrem.

Na tomto přehledu je vidět, že letových režimů je poměrně dost a orientace v nich je pro uživatele značně složitá. Při nechtěném zapnutí méně stabilizovaného režimu je navíc obrovské riziko zničení dronu. Z tohoto důvodu někteří výrobci v poslední době množství letových režimů značně omezují. Mnohdy **zůstávají přítomny pouze 2 režimy**. Prvním je režim plné stabilizace s GPS (`Position/Poshold`), kdy dron využívá všech svých senzorů, aby byl co nejméně ovlivňován pohybem vzduchu. Druhý režim (`Stabilized/Angle`) poskytuje také plnou stabilizaci, nevyužívá ale GPS a další senzory polohy. Dron tedy sám pomalu cestuje a je ovlivňován větrem. Odstranění ostatních režimů výrazně zjednodušuje ovládání, na druhou stranu pak není možné s dronem létat plnou rychlostí ani provádět akrobatické prvky.

V dalších kapitolách budeme předpokládat vždy ovládání plně stabilizovaného dronu s využitím GPS a všech dalších dostupných senzorů pro korekci polohy i výšky.

## 2.3 Autonomní let

Moderní drony umějí létat částečně autonomně v tzv. navigačních letových režimem. Během těchto režimů (s výjimkou režimu Cruise) UAV nereaguje na řízení a letí čistě v režimu autopilota.

**Cruise** — Letoun drží nastavený směr, při kombinaci s režimem `altitude` drží také stálou výšku. Obvyklý spíše pro letadla, pro multikoptéry se příliš nepoužívá.

**Return/RTH** — UAV přiletí zpět na místo startu (nebo jiné místo definované pomocí pozemní stanice) a přistane.

**Follow me** — UAV automaticky sleduje uživatele s pozemní stanicí (telefon nebo tablet se zapnutou GPS a příslušným softwarem). Ve výjimečných případech dron dokáže sledovat také uživatele bez GPS lokátoru pouze na základě počítačového vidění.

**Mission** — UAV automanomně postupně prolétává nastavenými body (tzv. WayPointy). Po dosažení posledního bodu letoun provede přednastavenou akci – obvykle přechod do režimu `Return/RTH`, přistání nebo zůstane na aktuální pozici.

**Safe-mode** — Spouští se automaticky při ztrátě řídícího signálu, poruše nebo dosažení kritického stavu baterie. UAV provede přednastavenou akci, obvykle tedy buď ihned přistane nebo se vrátí na místo startu a přistane tam.

I přes to, že výrobci integrují do svých dronů stále více senzorů a vylepšují software tak, aby co nejvíce eliminovali možnost nárazu do překážky, navigační režimy jsou stále určeny převážně pro let ve volném prostoru.

## 2.4 RC vysílač a přijímač

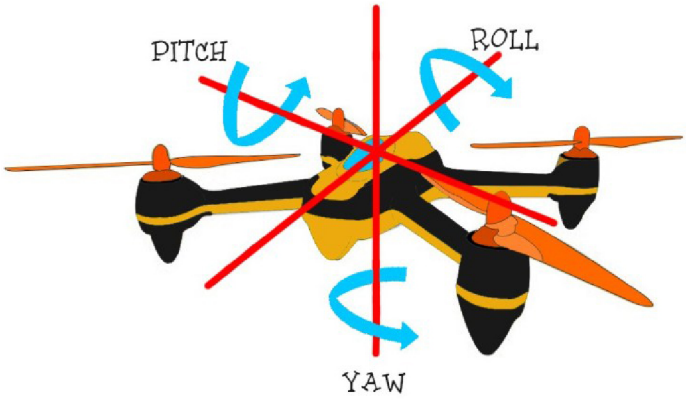
RC souprava slouží jako základní prostředek pro řízení bezpilotních letounů. Moderní RC vysílače pracují v pásmu 2,4GHz a běžný dosah se pohybuje kolem 1-2km v nezastavěné oblasti. Existuje mnoho různých komunikačních protokolů, proto je nutné vždy volit takový vysílač a přijímač, aby spolu dokázaly komunikovat – tedy zpravidla od jednoho výrobce. Některé RC soupravy navíc umožňují přenos telemetrických dat z letounu do vysílače a zobrazit tak na displeji informace o stavu baterie, rychlosti letu, výšce a podobně.

### Joysticky

Multikoptéru stejně jako kterýkoliv jiný bezpilotní letoun je možné ovládat náklonem ve 3 různých osách: Klopení (pitch) určuje náklon dronu vpřed a vzad a tedy i pohyb těmito směry, klonění (roll) náklon do stran a tedy pohyb vpravo a vlevo a bočení[3] (yaw) znamená zatáčení koptéry na místě.

[2]Zdroj obrázků: http://www.droneybee.com/how-to-fly-a-quadcopter

[3]Výrazy klopení, klonění a bočení vycházejí přímo z terminologie ÚCL (Úřad pro civilní letectví), vzhledem k tomu, že ale české názvy mohou být poměrně matoucí, budou zde uváděny spíše anglické výrazy, které jsou jednoznačnější a i v případě českých názvů místo pojmu bočení bude používáno slovo zatáčení, které nelze tak snadno zaměnit s pohybem koptéry do strany.




Obrázek 2.1: Multikoptéru je možné ovládat náklonem ve 3 různých osách: Klopení (pitch), klonění (roll) a bočení (yaw).[2]

### Módy vysílaček

Ovládání bohužel není vždy stejné a existuje více kombinací přiřazení ovládání náklonu jednotlivých os letounu joystickům vysílače. Významné jsou především módy 1 a 2. Existují také módy 3 a 4, ty už ale nejsou tolik běžné, proto zde popsány nebudou.

#### Mode 1

Mód 1 používají často letečtí modeláři především v Evropě a to hlavně z historických důvodů. V době, kdy nebyly programovatelné vysílačky ani elektronická stabilizace v modelech, potřebovali mít kontrolu nad řízením výškového kormidla při startu modelu z ruky a model obvykle hází silnější – tedy pravou rukou. Tento mód je ale často používán také pro ovládání dronů.


Obrázek 2.2: Mode 1 má plyn umístěný na pravém joysticku společně s kloněním (roll).

#### Mode 2

Výhoda druhého módu je v tom, že pravý joystick odpovídá kniplu ve skutečném letadle. Řídí se jím tedy náklon (a ted i pohyb) dopředu, dozadu (pitch) a do stran (roll). Levým joystickem se řídí plyn a zatáčení (obrázek 2.3). Piloti „velkých“ letadel se tedy snadněji naučí létat s modelem. Tento režim je u uživatelů dronů pravděpodobně o něco běžnější.


Obrázek 2.3: Mode 2 má plyn umístěný na levém joysticku společně se zatáčením (yaw).[4]

Obecně se nedá jednoznačně říci, který mód je lepší a záleží tak především na zvyku uživatele.

### Slidery a přepínače na vysílači

Na většině vysílačů se kromě joysticků nachází několik dalších dvoupolohových vypínačů, třípolohových přepínačů a také několik potenciometrů a sliderů, které umožňují plynulé ovládání dané funkce. Tyto ovládací prvky je pak možné přiřadit jednotlivým kanálům vysílače. U dronů pak přepínače obvykle slouží k volbě letových režimů, slidery a potenciometry k ovládání kamery.

## 2.5 Pozemní stanice

Pozemní stanice obvykle obsahuje tablet, počítač nebo mobilní telefon s připojenými přijímači telemetrie a videa. Na tomto zařízení je dále spuštěna aplikace, která zobrazuje video stream z kamery na dronu, mapu s vyznačenou pozicí dronu a letové údaje jako je např. výška a rychlost. Přijímače telemetrie a videa jsou u některých především komerčních dronů zabudovány přímo do RC vysílače, který se připojí k telefonu jediným USB kabelem. Pozemní stanice může obsahovat také otočnou směrovou anténu, která automaticky sleduje pohyb dronu. Díky tomu je možné dosáhnout většího dosahu a operovat tak na větší vzdálenosti i s menším vysílacím výkonem. Ergonomie tohoto řešení je však poměrně neuspokojivá.

### FPV brýle

Někteří piloti namísto dispeje telefonu používají brýle podobné těm pro virtuální realitu (obrázek 2.4). Video z kamery je doplněno o nějaká základní OSD (on screen data) – tedy základní data potřebná k pilotáži (stav baterie, výška, rychlost, počet GPS satelitů...) a takto upravené video je pak bezdrátově přenášelo do těchto brýlí (obvykle 2D, ale existují i 3D varianty). V brýlích může být také zabudovaný head-tracker – tedy zařízení, které snímá pohyb hlavy a otáčí podle něj kameru na dronu. Je tak možné podívat se do stran nebo dolů na zem pouhým otočením hlavy.

[4]Zdroj obr.: https://www.getfpv.com/learn/fpv-essentials/choosing-right-transmitter-mode/
[6]Zdroj obr.: https://www.engadget.com/2015/01/10/avegant-jellyfish-hands-on/

Obrázek 2.4: Někteří piloti pro řízení dronu přes video využívají brýle, do kterých je přenášen obraz z kamery na dronu.[6]

Teixeira a kolektiv [20] otestovali AR brýle Google Glass pro řízení dronu, obraz zde byl nejenom streamován do brýlí, pilot ale navíc řídil dron gesty a pohledem. Novější studie z roku 2018 autorů Erat a kolektiv [13] použila brýle Microsoft HoloLens k podobnému účelu. Operátor dron ovládal specifikováním míst, kam má přiletět. Dron byl tedy řízen do určité míry autopilotem. Demonstrační aplikace však zacházela ještě dál, protože ve vnitřních prostorech dokonce poskytovala pilotovi rentgenový pohled skrz zdi.

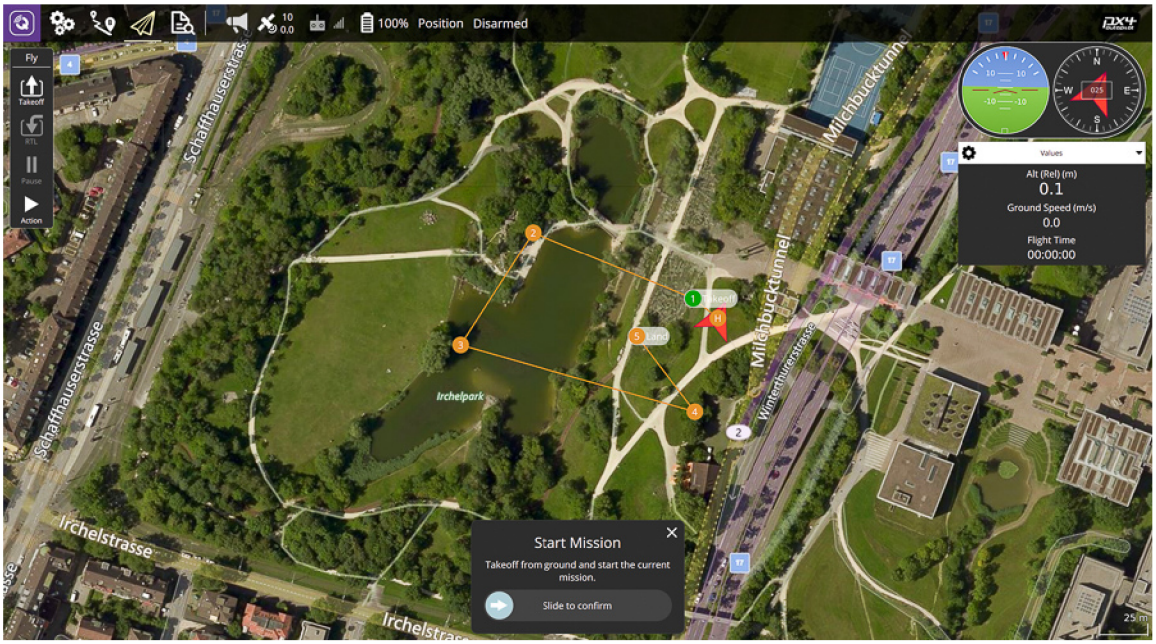
#### Data zobrazovaná na displeji pozemní stanice

Během letu je nesmírně důležité, aby měl pilot neustále přehled o pozici, výšce a orientaci letounu. Tento přehled může získat kromě přímého sledování letounu také pohledem do aplikace, která obvykle zobrazuje pozici dronu a HomePointu (místo startu) na mapě a také video přenos z kamery umístěné na koptéře (tzv. FPV – First person view). Dále jsou v aplikaci zobrazena aktuální data z koptéry – tedy stav baterie, výška, rychlost, počet GPS satelitů a podobně. V některých aplikacích jsou navíc zobrazeny přístroje podobné těm ze skutečného letadla (např. kompas a umělý horizont). Pozemní stanice dále umožňuje definovat mise pro autonomní let, přepínat letové režimy nebo provádět další nastavení dronu.

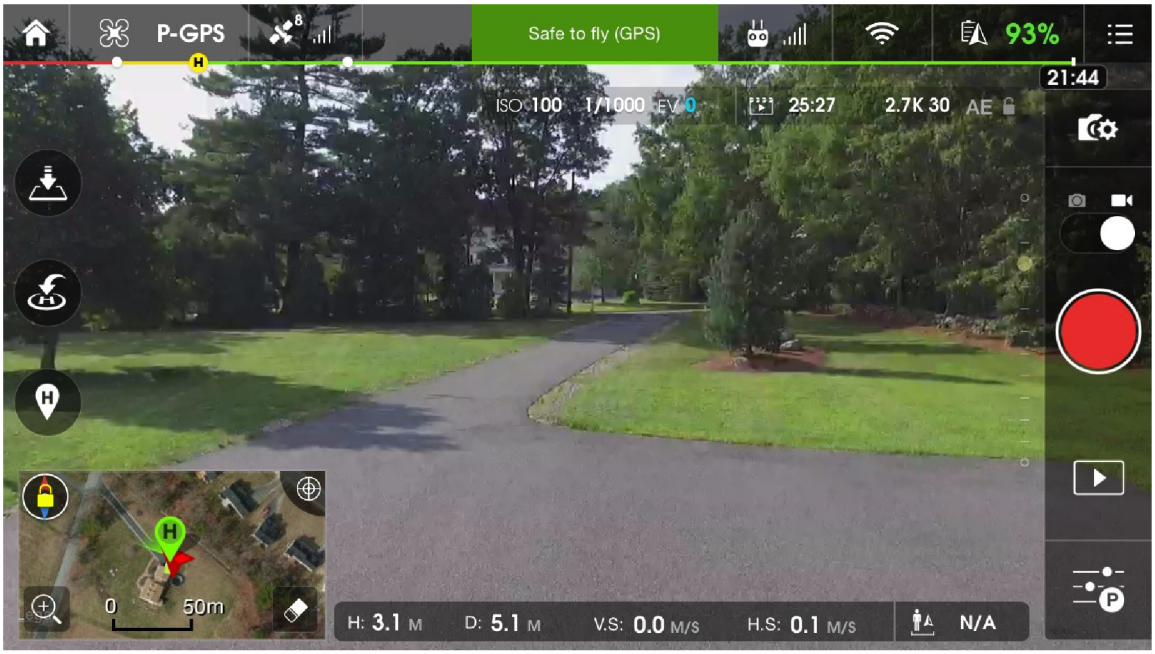
## 2.6 Další vybavení bezpilotního letounu

Přesto, že je tato práce zaměřena především na ovládání multikoptér a vizualizací informací z dronu, tato kapitola stručně popisuje jejich základní hardware a senzory. Pro pochopení celého textu není její přečtení bezpodmínečně nutné, kapitola však vysvětluje základní pojmy, které se objevují v dalších kapitolách.

[8] Zdroj obr.: https://dronelife.com/2015/09/15/a-look-inside-the-dji-go-app/




Obrázek 2.5: Pozemní stanice QGroundControll (QGC) 3.5 zobrazuje mapu, pozici dronu a umožňuje definovat misi, tedy posloupnost WayPointů, které dron automaticky proletí. Dále aplikace zobrazuje kompas, náklon dronu, aktuální rychlost, stav baterie a letový režim.

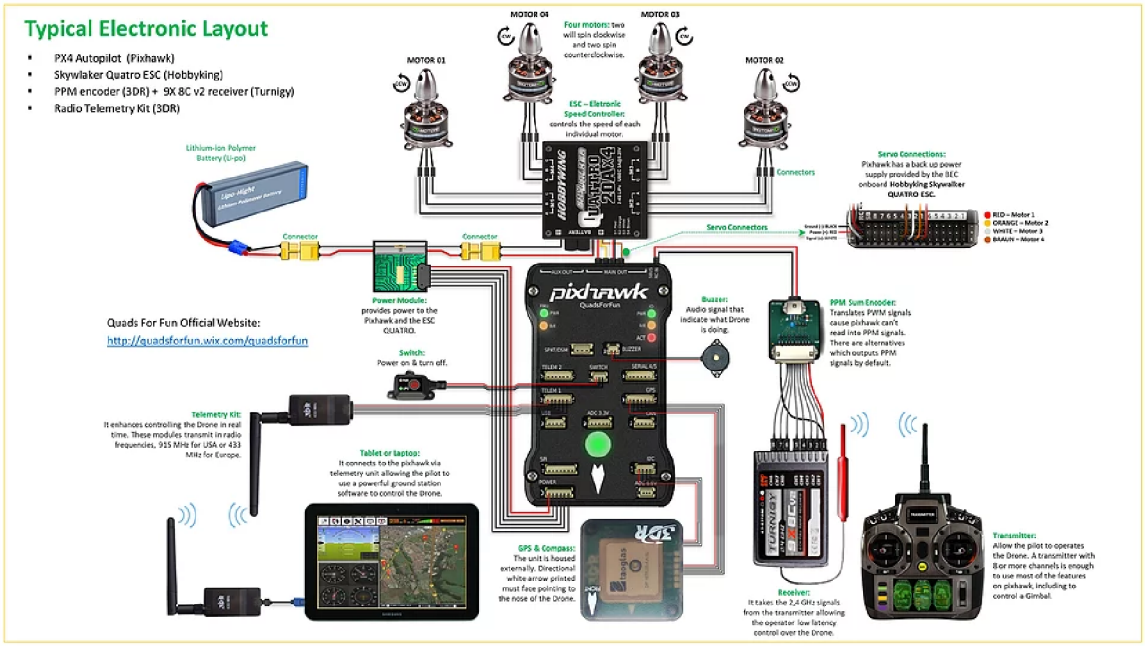



Obrázek 2.6: Pozemní stanice od DJI se na rozdíl od QGC zaměřuje v první řadě na zobrazení videa a možnosti jeho nahrávání. Zobrazení ostatních letových údajů už je ale velice podobné jako v ostatních aplikacích.[8]

### Řídící jednotka (Flight Controller)

Řídící jednotka je mozkem celého dronu. Obsahuje procesor, IMU a některé další senzory. Mnohdy je přímo vybavena barometrem a elektronickým kompasem. Naopak GPS modul je

téměř vždy připojován jako externí periferie. Dále řídící jednotka obsahuje velké množství vstupních a výstupních portů pro připojení veškerých dalších periferií. Mezi nejběžnější patří PWM výstupy pro připojení elektronických regulátorů a serv, několik sériových portů a sběrnice I2C. Pro připojení RC přijímače pak slouží obvykle sBus, iBus nebo PPM port. V procesoru řídící jednotky běží řídící software dronu.




Obrázek 2.7: Základní části multikoptéry.[9]

## Motory

V současnosti se používají pro veškeré drony a letecké modely téměř výhradně střídavé třífázové bezkartáčové (brushless) motory, které mají výrazně větší účinnost a poměr výkon/hmotnost než starší stejnosměrné motory. U dronů se pak nejčastěji vyskytují motory typu „outrunner“, které se vyznačují rotujícím pláštěm s permanentními magnety. Stator s cívkami se naopak nachází uvnitř a prochází jím pouze osa motoru.

Hlavními parametry pro výběr motoru je jeho maximální výkon (W) a počet otáček na volt (KV), který přímo souvisí s počtem závitů na cívkách motoru. Počet otáček motoru pak úzce souvisí s velikostí použité vrtule. Pokud má být dron schopný rychlé manipulace, volí se obvykle motory s vyššími otáčkami a menší vrtule, které nemají tak velkou setrvačnost. Pokud naopak vyžadujeme dlouhou dobu letu, používají se naopak vrtule co největší a nejpomalejší.

## Eletronické regulátory (ESC)

ESC je elektronické zařízení, které řídí rychlost elektromotoru v závislosti na vstupním signálu a také mění stejnosměrný proud z baterie na střídavý, který jde do motoru. Vyrábějí se buď jako samostatné moduly pro každý motor zvlášť, nebo v tzv. 4in1 provedení, kdy se na jediné desce nacházejí regulátory pro všechny 4 motory koptéry. Hlavními parametry

[9]Zdroj obr.: http://quadsforfun.wixsite.com/quadsforfun/typical-electronic-layout-pixhawk

pro výběr regulátoru jsou maximální přípustný proud (A) a podporované typy vstupního signálu. moderní ESC podporují kromě klasického PWM signálu také modernější protokoly jako Oneshot125, Oneshot42, MultiShot a DShot [2].

### Akumulátor

Jako zdroj elektrické energie pro drony se používají téměř výhradně Lithium-polymerové akumulátory (Li-pol, LiPo), které jako prakticky jediné dokáží poskytnout dostatečně velkou kapacitu při zachování přijatelné hmotnosti. Jmenovité napětí jednoho článku je 3,7V a pro drony se nejčastěji používají akumulátory s 3-6 články. Doba letu se pak nejčastěji pohybuje v rozmezí 15-35 minut u kvadrokoptér a přibližně dvojnásobku u letadel.

### Telemetrie

Telemetrií je obecně nazýván veškerý přenos informací z letounu na pozemní stanici k pilotovi. Těmito informacemi může být stav baterie, síla RC signálu, počet nalezených GPS satelitů, aktuální poloha, výška dronu, video z kamery a další.

Jistým otevřeným standardem pro přenos telemetrických dat z dronu je protokol MAVlink. Pro přenos videa se velmi často používá samostatný analogový přenos na pásmu 5,8GHz. Výrobci jako např. DJI však často používají své vlastní protokoly, které přenášejí současně telemetrická data i digitální video – zpravidla na frekvenci 2,4Ghz. To umožňuje přenos mnohem kvalitnějšího obrazu.

Pokud není vyžadován let na dlouhou vzdálenost, je možné využít také telemetrický přenos pomocí WiFi, která umožňuje přenos mnohem většího množství dat, bohužel však právě na úkor maximální vzdálenosti přenosu.

### Senzory

Pro určení polohy a náklonu dron využívá velké množství senzorů, zde je přehled těch nejzákladnějších:

**IMU** — Zpracovává data o poloze a zrychlení na základě jednoho nebo více gyroskopů a akcelerometrů. Některé IMU navíc obsahují integrovaný kompas.

**Kompas** — Elektromagnetické kompasy, podobně jako klasické kompasy, vycházejí z existence magnetického pole Země [17]. V dronech se používají nejčastěji tříosé magnetometry, které musí být umístěny co nejdál od motorů, antén a dalších zdrojů elektromagnetismu.

**Barometr** — Kvalitnější barometry dokážou určit výšku na základě tlaku vzduchu s přesností na cca na 20-50cm. Protože se však tlak vzduchu neustále mění, určují pouze výšku relativní — tedy rozdíl mezi výškou startu a výškou, kde se dron právě nachází.

**Sonar** — Měří vzdálenost od překážky na základě výpočtu z časového intervalu mezi vysláním zvukového signálu a ozvěnou [17].

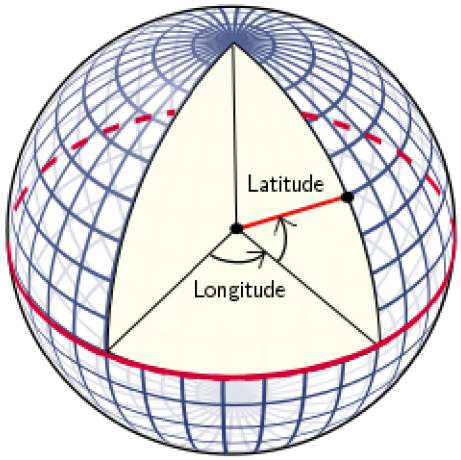
**Optical Flow Sensor** — Kamera s nízkým rozlišením, která pomáhá udržet dron na jednom místě při letu v malé výšce. Pracuje na podobném principu jako optická myš. Kamera sleduje objekty pod dronem a analyzuje směr jejich pohybu. Na základě těchto informací je let dronu upraven tak, aby se co nejméně pohyboval [4].

**GPS modul** — Určuje polohu na základě příjmu signálu z několika GPS satelitů. S počtem satelitů roste přesnost určení polohy. Moderní GPS jednotky proto dokážou přijímat signál také z konkurenčních sítí jako jako je evropský systém Galileo, ruský Glonass a čínské BeiDou a díky tomu mohou polohu dále zpřesňovat. Přesnost GPS se nejčastěji pohybuje v řádu nižších jednotek metrů. Satelitní systémy mají ale obvykle mnohem horší přesnost dat ve vertikálním směru, nadmořská výška je tedy ještě dále zpřesňována použitím barometru. Souřadnice jsou zpravidla vraceny ve formátu WSG84.

Data o poloze jsou vždy přesnější za letu ve větší výšce, kdy se dron nachází nad úrovní překážek, které brání příjmu signálu ze satelitů. Profesionální drony často disponují více samostatnými GPS přijímači umístěnými nad úrovní dronu, díky tomu je možné dosáhnout ještě větší přesnosti i spolehlivosti.

### Geografické souřadnice ve formátu WSG84

Zeměpisná šířka (latitude) se měří od rovníku k pólům a nabývá hodnot -90–90, na severní polokouli kladných a na jižní záporných. Spojnice bodů se stejnou zeměpisnou šířkou se nazývají rovnoběžky. Zeměpisná délka (longtitude) je úhel, který svírá rovina základního Greenwichského poledníku a místního poledníku (viz obrázek 2.8). Hodnota se pohybuje od 0 do 180, v kladných hodnotách na východní polokouli a v záporných hodnotách na západní. Kružnice spojující body se stejnou zeměpisnou délkou se nazývají poledníky. Pro tento souřadnicový systém se někdy používá označení WSG84 (nebo také EPSG:4326) [18].




Obrázek 2.8: Geografické souřadnice WSG84. Bod na povrchu je jednoznačně určen pomocí zeměpisné šířky (latitude) a zeměpisné délky (longtitude)[11].

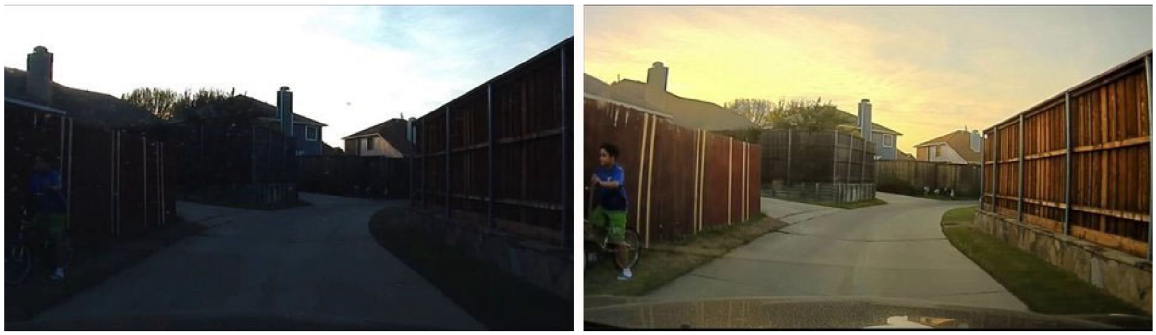
## Kamera a gimbal

Kamera je při řízení prostřednictvím videa hlavním prostředkem pro vnímání prostředí, ve kterém se dron pohybuje. Z toho důvodu je kvalita kamery velmi důležitá. Volí se kamery s vysokým dynamickým rozsahem obrazu (HDR) a vyšším kontrastem, aby pilot byl schopen snadněji rozlišit jednotlivé objekty na videu (obrázek 2.9) i za horších světelných podmínek.

[11] Zdroj obrázku: https://vvvv.org/blog/polar-spherical-and-geographic-coordinates

Použití kamery s horším dynamickým rozsahem je problém zejména při letu nad jednotvárnou krajinou. Příkladem může být let mezi poli na podzim, kdy je vše hnědé – pole, tráva i stromy. Pilot pak velmi snadno ztratí přehled, nad čím se letoun vlastně nachází nebo přehlédne nějakou překážku.

Dron se během letu neustále naklání a chvěje. Pro kvalitní záběry jsou proto drony vybaveny odpruženým gimbalem s tříosou stabilizací. Pohyb ve třech osách zajišťují malé motory. Gimbal je možné obvykle také dálkově ovládat a nastavit tak směr, kterým směrem má být kamera otočena.



Obrázek 2.9: Rozdíl v kvalitě obrazu při použití kamery s HDR technologií (vpravo) – Na první pohled je vidět, že podle druhého videa se bude dron řídit mnohem lépe[13].

## 2.7 Řídící software

Použitý řídící software silně ovlivňuje schopnosti a funkce dronu. Software je ale silně svázán s použitou řídící jednotkou, každý software má totiž vždy své podporované desky, u kterých je jistota, že bude fungovat správně.

### Ardupilot / Arducopter

Ardupilot je vyspělý, plnohodnotný a spolehlivý open-source autopilot software. Je vyvíjen už více než 5 let týmem různých odborných inženýrů, počítačových vědců i nadšenců. Je schopný řídit prakticky cokoliv, od konvenčních letounů, multikoptér a vrtulníků až po čluny a dokonce i ponorky. Existuje pro něj velké množství podporovaného hardware. Pro řízení dronů se používá verze Arducopter [4].

### PX4 autopilot

PX4[14] je open-source software vycházející z open-hardware projektu Pixhawk[15]. Řídící jednotky Pixhawk jsou velice oblíbené především ve výzkumných projektech a dražších dronech. Podobné oblibě se těší také software PX4. Ten podporuje obrovské množství různých typů dronů, letounů a VTOL. Určitou zvláštností je to, že pro všechny tyto rozdílné typy používá vždy stejný software — pouze jinak nakonfigurovaný. Podporuje robotický middleware ROS a vývojářům poskytuje kvalitní SDK a API, díky těmto vlastnostem je často využíván právě ve výzkumných projektech.

[13] Zdroj obrázku: https://watchguardvideo.com
[14] https://px4.io/
[15] http://pixhawk.org/

### Betaflight / CleanFlight / INAV

CleanFlight[16] nebo jeho klon Betaflight[17] je open-source řídící software především pro levnější řídící jednotky a hodí se hlavně pro závodní drony nebo letadla. Díky podpoře levných desek tento software často používají čínští výrobci ve svých levných dronech.

INAV vznikl jako jejich další klon s výrazně přepracovanými navigačními módy a lepší podporou GPS. V současné době u něj probíhá velice rychlý vývoj a nové verze se značnými vylepšeními vycházejí několikrát ročně. Rychle také roste seznam podporovaných řídících jednotek [2].

## 2.8 Česká legislativa

V ČR je létání bezpilotních letounů relativně přísně regulováno leteckým předpisem L2 a především jeho doplňkem X, který se zabývá právě bezpilotními systémy. Tyto předpisy vyžadují, aby všichni piloti s výjimkou rekreačního a sportovního využití byli evidováni u Úřadu pro civilní letectví (dále jen ÚCL) a získali tzv. povolení k létání letadla bez pilota – tedy něco jako pilotní průkaz pro bezpilotní systémy. Z tohoto důvodu jsem také já a další kolegové, kteří na tomto projektu spolupracují, musel absolvovat zkoušku a toto povolení získat.

### Doplněk X a získání povolení k létání

Doplněk X leteckého předpisu L2 [3] upravuje veškeré lety bezpilotních letounů především s ohledem na bezpečnost osob, majetku na zemi, dalšího leteckého provozu a životního prostředí. Z tohoto důvodu přesně definuje prostory ve kterých se létat smí a prostory, v nichž je provoz nějakým způsobem omezen nebo úplně zakázán. Dále jasně vymezuje, že odpovědnost za let má vždy osoba, která letoun dálkově řídí a to bez ohledu na úroveň automatizace tohoto letounu.

Dále s výjimkou, kdy ÚCL povolí jinak, stanovuje, že pilot, nebo kromě pilota i další poučená osoba, vždy musí mít letoun v přímém dohledu, aby mohl sledovat a vyhodnocovat dohlednost, překážky a další letový provoz. Z tohoto důvodu zatím **není možné legálně řídit dron pouze s využitím přenosu videa z dronu** bez přítomnosti druhé osoby, která dron sleduje a udržuje v přímém dohledu. Dá se však předpokládat, že se situace v tomto ohledu v budoucnosti změní, protože již s použitím současných technologií je možné drony poměrně bezpečně pilotovat pouze s využitím videa a s dalším vývojem technologií se situace bude pravděpodobně ještě dále zlepšovat a právě možnostmi vylepšení tohoto způsobu pilotáže se zabývají další kapitoly této práce.

Zkouška pro získání povolení k létání se skládá ze tří částí. První je ústní zkouška, která ověřuje, že pilot zná rozdělení vzdušného prostoru, dokáže se orientovat v letecké ICAO mapě a aplikaci AisView[18]. a ví, za jakých podmínek v daných prostorech smí dron provozovat. Dále je ověřováno, zda uchazeč zná minimální vzdálenosti, ve kterých se smí létat od osob, staveb a hustě obydlených oblastí a dále zná ochranná pásma silnic, dálnic a inženýrských sítí.

Druhou částí zkoušky je písemný test z doplňku X a třetí částí praktická zkouška, při které pilot prokazuje, že dokáže s letounem zaletět předepsané sestavy a neztratí při

[16] CleanFlight software: http://cleanflight.com/

[17] BetaFlight software: https://github.com/betaflight/betaflight

[18] Webová apliakce AisView od ŘÍZENÍ LETOVÉHO PROVOZU ČR pro zobrazení aktuálních vzdušných prostorů a omezení: http://aisview.rlp.cz/

tom přehled o orientaci dronu. Tyty sestavy je nutné zaletět jak v režimu plné stabilizace, tak také s vypnutou GPS, kdy je letoun značně ovlivňován větrem (o letových režimech pojednává kapitola 2.2). Nakonec je ještě nutné prokázat, že na dronu funguje „fail-safe systém“, který zaručí, že letoun v případě ztráty signálu sám bezpečně přistane. Tímto systémem musí být vybaveny všechny drony s hmotností nad 0,91kg.

Povolení k létání má však na rozdíl od pilotního průkazu na „velké letadlo“ nevýhodu v tom, že platí pouze pro jeden konkrétní typ bezpilotního letounu, nikoliv na celou kategorii (větroň, ultralight, vrtulník...). V tomto ohledu je tedy legislativa velice přísná.

### Minimální vzdálenosti a zakázané prostory

Dalším omezením provozu dronů jsou minimální horizontální vzdálenosti od osob, staveb a hustě osídlených prostor, které jsou stanoveny pro letouny těžší než 7kg od osob, prostředků a staveb minimálně na 50m během startu a přistání a 100m během letu. Minimální vzdálenost od husté zástavby je 150m. Pro lehčí letouny tato minima neplatí, je však stanovena tzv. „bezpečná vzdálenost“, která vyloučí možnost zranění osob a poškození majetku na zemi. ÚCL doporučuje jako bezpečnou vzdálenost poměr 1:2 při letu s dopřednou rychlostí (na 50m výšky poloměr 100 metrů) a 1:1 bez dopředné rychlosti (na 50m výšky, kruh o poloměru 50 metrů).

Dále je nutné dodržování minimální vzdálenosti 100m od dálnic, rychlostních komunikací a vysokorychlostních železnic, 60m od železnic, 50m od silnic I. třídy, 25m od silnic II. třídy a 15m od místních komunikací [7]. Z toho vyplývá, že ani **místní komunikaci**, kde se momentálně nepohybuje žádné vozidlo, **není možné legálně přeletět** bez dalších povolení.

Maximální výška letu je stanovena na horní hranici letového prostoru třídy G – tedy 300m nad zemí. Ve vyšších výškách je možné létat pouze v oblastech, kde je zajištěno poskytování informací o pohybu dronu ostatnímu leteckému provozu. Zpravidla tedy na neřízených letištích na základě koordinace s letištní a letovou informační službou AFIS nebo se stanovištěm poskytování informací známému provozu. V okruhu 5,5km od řízeného letiště se drony nad 0,91kg nemohou pohybovat vůbec. V ochranných pásmech letišť (dráhy a jejich okolí) a některých přírodních rezervacích navíc drony nesmějí létat vůbec a to ani ty nejmenší.

U naprosté většiny omezení je však možné požádat ÚCL o výjimku a profesionálové v oboru leteckých prací, kameramani nebo policisté, kteří drony využívají ke své práci o ni tedy musí pravidelně a opakovaně žádat.

Všechny tyto limity jsou pro pilota značně svazující, velká většina z nich je však s ohledem na bezpečnost nutná, protože i přes stále se zvyšující kvalitu dronů, zatím stále ještě k občasným poruchám dochází. Je tedy nutné si uvědomit, že v případě poruchy a pádu i malého dronu (1kg) z výšky např. 100m může dojít k velmi vážnému zranění nebo dokonce úmrtí osoby. V případě střetu s malým letadlem, které se obvykle pohybuje rychlostí 100-180km/h, může v krajním případě dojít i k pádu tohoto letadla. S většími a těžšími drony pak tato rizika dále narůstají. Je tedy možné, že v budoucnosti budou muset být profesionální drony certifikovány a pravidelně kontrolovány oprávněnými techniky podobně jako běžná letadla.

## 2.9 Simulátor

Pro testování ovládání koptéry a orientace pilota v prostoru je vhodné využít kromě reálné kvadrokoptéry také simulátor. Dostupných simulátorů je sice celá řada, jejich možnosti jsou však zatím stále poměrně omezené. Některé simulátory vyžadují přímé připojení hardwaru fyzické řídicí jednotky. Simulátor pak jednotce posílá data ze senzorů a ta zase naopak posílá simulátoru tah jednotlivých motorů, které simulátor zpracuje a převede na pohyb dronu prostorem. Některý řídicí software je možné simulovat pouze softwarově bez nutnosti mít připojený reálný hardware (např. Pixhawk). I tak je ale rozjetí takového simulátoru značně náročný úkol, protože je nutné správně nakonfigurovat a propojit několik spolupracujících aplikací a provézt prakticky stejnou konfiguraci softwaru jako u reálného dronu.

Zajímavými simulátory jsou především obecný robotický simulátor Gazebo a aplikace Airsim, která se specializuje pouze na simulaci dronů a automobilů. Vzhledem k tomu, že nebyl k dispozici reálný hardware podporované řídicí jednotky, vyzkoušel jsem simulátor Airsim, který nabízí přímo vestavěnou jednoduchou softwarovou simulaci kvadrokoptéry.

### AirSim

AirSim[19] je open-source simulátor od Microsoftu postavený na Unreal Engine, který ale ve své poslední verzi obsahuje také experimentální verzi pro Unity. Je vyvíjen především pro výzkum UI, počítačového vidění a posilovaného učení pro autonomní prostředky. Zpřístupňuje API pro ovládání autonomních funkcí pomocí jazyků C nebo Python obsahuje svůj vlastní jednoduchý simulátor řídicí jednotky, ale kromě něj umožňuje také pokročilejší simulaci typu hardware-in-loop s připojeným reálným flight-controllerem. Podobně je možné využít také softwarovou simulaci (software-in-loop) [19].

Obrázek 2.10: Simulátor Airsim: **vlevo:** výchozí prostředí simulátoru, bohužel však placené, **vpravo:** jedno z mála prostředí dostupných zdarma Landscape Mountains.

Airsim nemá žádné vestavěné možnosti nastavení a konfigurace se provádí pomocí úprav konfiguračního souboru ve formátu JSON. Možnosti konfigurace jsou však omezené, např. chybí důležité nastavení mapování os ovladače na osy kvadrokoptéry. Takže s připojením běžného gamepedu s 2 joysticky simulovanou koptéru nelze ovládat. Situace se paradoxně nezměnila ani nákupem oficiálně podporovaného ovladače FS-100, který převádí signál PPM ze standardní modelářské vysílačky do počítače. Ani s tímto ovladačem však simulátor nefungoval zcela správně.

[19] https://github.com/Microsoft/AirSim/

#### Kompilace

Pro řešení tohoto problému bylo nutné upravit soubor, který řeší zpracování dat z ovladačů a vše znovu zkompilovat. AirSim je vytvořen jako plugin pro Unreal Engine. Stačilo tedy vytvořit projekt v Unreal Engine, vložit do něj nějaké prostředí, následně vložit plugin Airsim, nastavit několik parametrů a startovací pozici dronu a vše zkompilovat pro cílovou platformu.

Vlastní kompilace má však jednu podstatnou nevýhodu. Zatímco již zkompilované binárky jsou dostupné s mnoha různými prostředími, při vlastní kompilaci se každý musí omezit na ta prostředí, která má koupená nebo jsou zdarma. Výběr je tedy velice omezený. Budoucí verze tohoto simulátoru už ale snad přinesou možnost mapování os ovladače pomocí konfiguračního souboru.

## 2.10 Volně dostupné mapy

Navržená aplikace bude potřebovat volně dostupná data pro vytvoření modelu prostředí, ve kterém se dron pohybuje. Mezi taková data je možné zařadit mapy, satelitní snímky, výškové mapy, 3D modely budov a mnoho dalšího. V dalších kapitolách jsou všechna tato data nazývána souhrnně jako „mapová data“. Vzhledem k tomu, že vytvořit model světa z těchto jednotlivých částí by bylo velmi náročné a tedy i výrazně nad rozsah této práce, hledání vhodných dat se zaměřuje především na celé knihovny, které už mají samy o sobě implementovánu co největší část požadovaných funkcí. Tato kapitola tedy stručně shrnuje, jaké mapové knihovny jsou v současnosti zdarma k dispozici.

Společností, které disponují mapovými data existuje celá řada. Mezi nejznámější mapové aplikace patří Google Maps, Mapy.cz, Mapbox, Nokia Maps (HERE.com), Virtual Earth (Bing Maps), Arc-GIS a Open Street Maps. Mapy se liší jak svojí kvalitou (aktuálnost, rozlišení, různé typy map...), tak možnostmi a pravidly použití. Poskytovatelé obvykle provozují své vlastní mapové aplikace, kde si zachovávají některé exkluzivní funkce a v nějaké omezené podobě umožňují využití map také třetím stranám prostřednictvím API (Application Programming Interface). Díky tomu je možné tato mapová data využívat ve vlastní aplikaci. Každá taková společnost obvykle poskytuje API pro několik různých platforem (web, Android, iOS, Unity, Unreal Engine...).

API pro jednotlivé platformy se pak obvykle liší nejenom množstvím dostupných funkcí, ale mnohdy také licenčními podmínkami. Určitým standardem je možnost použití mapových dat do nějakého počtu požadavků denně zdarma. Při větším počtu uživatelů se platí právě počet načtení mapových dat ze serverů těchto společností, které je přímo úměrné počtu uživatelů aplikace.

Samostatnou skupinou jsou pak aplikace a API založené na otevřených mapách Open StreetMap. Zde existuje celá řada poskytovatelů, kteří obvykle využívají svoji vlastní variantu mapových dat, které se často mírně liší a obvykle ve snaze odlišit se od konkurence přidávají něco navíc. Vzhledem k tomu, že jejich data vycházejí ze stejného základu, jsou si ale velice podobné a liší se tedy především možnostmi API a podporou různých platforem.

### 2.10.1 Google Maps

Google nabízí mapové API pro různé platformy jako je JavaScript, Android i iOS. API pro jednotlivé platformy se liší jen minimálně a všechna tak poskytují velice podobná data. 3D modely budov jsou ale poměrně jednoduché a bez textur (obrázek 2.11). Hezké texturované

budovy Google zobrazuje pouze ve svých aplikacích a poskytuje jim tak určitou exkluzivitu. Nevýhoda všech těchto API je však v tom, že mají značně omezené ovládání a není např. možné libovolně manipulovat s kamerou. Nabízí pouze pohled kolmo shora nebo pod určitým pevným úhlem ze všech světových stran. Zcela volné možnosti manipulace s kamerou však zatím nenabízejí.


Obrázek 2.11: Mapová data společnosti Google **vlevo:** detailní 3D objekty přímo v aplikaci Google. **vpravo:** mapa poskytovaná třetím stranám prostřednictvím API už obsahuje výrazně chudší 3D modely.

Aby bylo možné ve výsledné aplikaci volně manipulovat s kamerou a pohledem do scény s mapou, musí tyto funkce podporovat mapové API. Některé mapové aplikace jsou distribuovány ve formě pluginu pro nějaký 3D engine, který právě umožňuje přidávání dalších 3D objektů a libovolnou manipulaci s kamerou. Další hledání tedy bylo omezené především na pluginy pro UnrealEngine a Unity.

Mapy společnosti Google mají ale jednoznačně nejkvalitnější 3D objekty včetně kvalitních textur (obrázek 2.11). Google je ale v této podobě zatím neposkytuje třetím stranám. Existuje sice plugin pro prostředí Unity, ale po komunikaci s obchodním zástupcem Googlu mi bylo sděleno, že jej poskytují pouze vývojářům her. Pro jakékoliv jiné typy aplikací ho tedy není v současné době možné použít.

Ve formě pluginu pro Unity jsou distribuovány ještě další dvě aplikace – Online Maps for Unity a MapBox for Unity. Online Maps for Unity[20] nabízí několik různých mapových podkladů (Google Maps, Mapbox, ArcGIS, Nokia Maps (HERE.com), Virtual Earth (Bing Maps), Open StreetMap). Aplikace ale není úplně zdarma — stojí $30, oproti tomu Mapbox for Unity, který nabízí podobné možnosti, je při dodržení určitých podmínek zcela zdarma.

### 2.10.2 MapBox for Unity

Plugin MapBox[21] pro Unity využívá především data z OpenStreetMap. Plugin automaticky načítá mapové podklady a výšková data a vytváří tak plastický terén. Dále je možné zvolit typ mapy. MapBox obsahuje několik klasických i satelitních map. Díky využití Unity je

[20] https://assetstore.unity.com/packages/tools/integration/online-maps-13261
[21] https://docs.mapbox.com/unity/maps/examples/replace-features/

možné libovolně manipulovat s kamerou a také přidávat do scény jakékoliv další objekty. Mělo by být tedy možné integrovat např. simulátor Airsim, který je sice primárně určen pro Unreal Engine, nová verze už ale obsahuje také betaverzi pluginu pro Unity.

MapBox obsahuje také několik mapových vrstev s 3D modely budov (obrázek 2.12), jejich kvalita je však nesrovnatelně horší než v případě Googlu. V českém prostředí na rozdíl od velkých amerických měst je navíc velmi málo modelů s reálnými texturami a skutečným tvarem střechy. Obvykle jsou dostupná velice podobná data jako v samotném Open Street Map, tedy hrubý obrys budovy a její přibližná výška.


Obrázek 2.12: Mapové aplikace založené na OpenStreetMap — **vlevo:** Mapový engine Mapbox pro Unity obsahuje jen jednoduché 3D modely, dokáže ale v Unity vymodelovat plastický terén, což je nejdůležitější vlastnost, kterou od aplikace požadujeme. **Vpravo:** Pro srovnání webová aplikace OSM Buildings[23], která využívá informace o budovách přímo z OpenStreetMap, je vidět, že budovy jsou velice podobné.

[23]OSM Buildings:https://osmbuildings.org/

# Kapitola 3

# Návrh aplikace a volba technologií

Při řízení přímým pohledem na letoun se pilotovi orientuje poměrně snadno v případě, že je letoun dostatečně blízko. Pilot totiž jednoznačně vidí, kam směřuje příď letounu a kam se letoun pohybuje. U koptéry je situace o něco horší než u letadla, protože zde nelze tak snadno určit, která část je přídí. Proto jsou obvykle přední rotory rozlišeny jinou barvou. Při létání na větší vzdálenosti je však jednoznačné určení směru natočení dronu obtížné. Cho a kolektiv [12] tento problém řešili zavedením egocentrického ovládání dronu, které ho automaticky rotuje tak, aby měl svou záď neustále natočenou směrem k pilotovi. Ten jej tak řídí ze své perspektivy a nemusí řešit, kterým směrem je v daný okamžik natočena příď letounu.

## 3.1 Problém ztráty vnímání pozice během řízení přes video

Při řízení skrze pohled z kamery (FPV) – což je v době psaní této práce v České republice stále ještě oficiálně zakázáno (viz kapitola 2.8), je situace značně odlišná. Pilot vidí k čemu míří příď a za předpokladu, že vidí v obraze nějaké jemu známé orientační body, zná přesnou orientaci dronu. Co však už vidí mnohem hůř, jsou překážky okolo dronu. Pilot totiž vidí jen před sebe, koptéra ale může letět také na stranu nebo dokonce vzad (mnohdy navíc plnou rychlostí). Zde už pilot ale překážky vůbec nemá šanci vidět.

Druhým problémem, který se při tomto způsobu ovládání často projevuje je fakt, že kamera zabírá jen určitou část prostoru a pokud pilot v obrazu nevidí žádný známý orientační bod, může ztratit přehled nejen o natočení dronu, ale také o jeho aktuální poloze. Pohled z kamery totiž zabírá pouze úzký úsek prostoru a pilot se tak může ztratit i v prostředí, které jinak dobře zná. Pohled z výšky navíc vypadá poněkud jinak než pohled ze země a méně trénovaný člověk s tím může mít problémy. Zde je situace horší především při větších rychlostech a prudkých obratech. Pokud člověk totiž obrat sleduje jen skrze video a nemá žádné vestibulární podněty a jiné fyzické pocity otáčení, může dojít k dezorientaci velmi snadno [14].

Netrénovaný člověk však ztratí přehled o pozici letounu velmi často i při relativně pomalém letu v malé výšce (pod 50m). Situaci pak dále ještě komplikuje pohyb v prostředí bez výrazných orientačních bodů (např. velké pole nebo větší množství podobných střech budov). Na ztrátu orientace má také velký vliv stabilizace kamery a kvalita video přenosu, pokud se obraz třepe, zrní a občas vypadává, což u dronů není žádnou výjimkou, může být tento způsob řízení pro člověka velice náročný. Pozici pak musí člověk hledat v mapě

zobrazené také na pozemní stanici, ale ani v tomto případě mnohdy není úplně snadné si tuto polohu rychle spojit s pozicí v prostoru.

Tyto nežádoucí efekty je možné částečně eliminovat použitím kamery s vysokým rozlišením, optikou s širokým zorným polem (často dokonce přes 170°), kvalitní stabilizací obrazu (gimbalem) a kvalitním přenosem. Zde se však naráží na dostupnost technologií, vysoké ceny a především omezení vysílacích výkonů. V některých případech také pilot může do určité míry otáčet kamerou a tím svůj pohled rozšířit. Především při letu ve vyšších výškách (nad 150m) je mnohem důležitější pohled směrem dolů než pohled před sebe. Při letu v malých výškách je naopak důležitý pohled přímo před sebe a kvalitní stabilizace, protože se dron velmi prudce naklání. V určitých případech tak může být dokonce výhodnější použít kameru s poměrem stran 4:3 než moderní širokoúhlé, pilot tak totiž vidí mnohem více ve vertikální rovině, tedy především před sebe a dolů, což je často velmi důležité. I přes všechny tyto možnosti však zůstává tento způsob pilotáže značně náročný a dal by se mnohdy přirovnat k jízdě autem v hustém městském provozu se začerněnými všemi skly a ponechanou pouhou polovinou čelního skla.

Při ztrátě orientace však roste riziko nárazu do překážky, případně vlétnutí do zakázané (viz kapitola 2.8) oblasti (např. hustá zástavba, ochranné pásmo liniových staveb...). Toto riziko se ještě zvyšuje při větších rychlostech nebo silném a nárazovém větru. Při ztrátě orientace pak netrénovaný pilot mnohdy rychlým zásahem do řízení ve snaze vyhnout se překážce, provede tento zásah přesně opačným směrem než zamýšlel a pokud dron neobsahuje senzory, které mu v tom zamezí, dojde k nárazu do překážky a poškození letounu.

**Pilot tedy musí za letu v každém okamžiku přesně vědět:**

**Kde se dron nachází** — pokud pilot neví, kde se letoun nachází, nemůže ho správně řídit.

**Jak je letoun otočen** — pilot pilot neví, musí zkoušet náhodné zásahy do řízení a sledovat, kam se dron pohne. Pokud je v okolí dronu překážka, může do ní pilot vrazit právě špatným zásahem do řízení ve snaze se jí vyhnout.

**Jaké překážky jsou v okolí dronu** — každá překážka představuje riziko kolize a zičení letounu, pilot tedy musí neustále vědět, na co si má v danou chvíli dávat pozor.

**Jakým směrem a jak rychle se pohybuje** — riziko kolize roste s rychlostí pohybu.

**Co se nachází pod dronem** — Dron může kdykoliv vlivem technické poruchy nebo např. srážky s ptákem spadnout a pilot je povinen s touto možností počítat, musí proto vědět, co se nachází pod dronem a v tzv. dopadové vzdálenosti a let upravit tak, aby tato rizika eliminoval.

**Kam nesmí letět** — Toto částečně souvisí s předchozím bodem. Existují však také oblasti, kde sice nemusí hrozit bezprostřední nebezpečí z pádu letounu, ale dron tam z nějakého důvodu stejně nesmí (viz kapitola 2.8).

**Směr, kam letět chce** — Pokud pilot musí kličkovat za letu mezi překážkami a nemá zrovna v zorném poli místo, ke kterému míří, snadno se odchýlí od původního směru a i v případě, že neztratí orientaci úplně, letí výrazně delší cestou, což zdržuje a zbytečně vybíjí baterii.

## 3.2 Možná řešení pro zlepšení orientace pilota

Při hledání řešení problému ztráty prostorové orientace během řízení bylo nutné brát ohled právě na tyto věci, které musí pilot vždy vědět. Je jasné, že současná technologie ještě nemůže v každém okamžiku dokonale splnit všechny tyto požadavky, je však možné pokusit se najít řešení, která by mohla alespoň částečně pomoci pilotovi cítit se tak, jako by se pohyboval s dronem a mohl se volněji rozhlížet po okolí a nebýt tolik svázaný tím, co v daném okamžiku kamera zabírá.

Obrázek 3.1: Aby mohl pilot dobře řídit, musí z kabiny perfektně vidět a konstruktéři to vědí. Cockpit moderního vrtulníku proto nabízí rozhled do všech stran a má prosklenou dokonce i podlahu. Nebylo by skvělé, kdyby i pilot dronu mohl mít podobný přehled o okolí? [2]

### 360° kamera

Pro zlepšení orientace pilotovi může pomoci např. všesměrová (360°) kamera, díky které uvidí nejenom dopředu, ale také za sebe. Ale ani to však nemusí být úplně vhodné. Přidání kamery totiž výrazně zvyšuje nároky na přenos dat. Většina na trhu dostupných 360° kamer navíc obsahuje pouze 2 objektivy umístěné v úhlu 180° (tedy zády k sobě). Jak už ale bylo řečeno. Pilot potřebuje vidět především dopředu a dolů — z tohoto pohledu už tedy běžná 360° kamera, která nahoru a dolů prakticky nevidí zase tolik vhodná není. Dalším problémem je využívání čoček typu „rybí oko“, které mají velké zkreslení. Objektivů by tedy pravděpodobně bylo potřeba ještě více a tím by se problém s přenosem dat ještě prohloubil.

[2] Zdroj obrázku: https://www.ainonline.com/

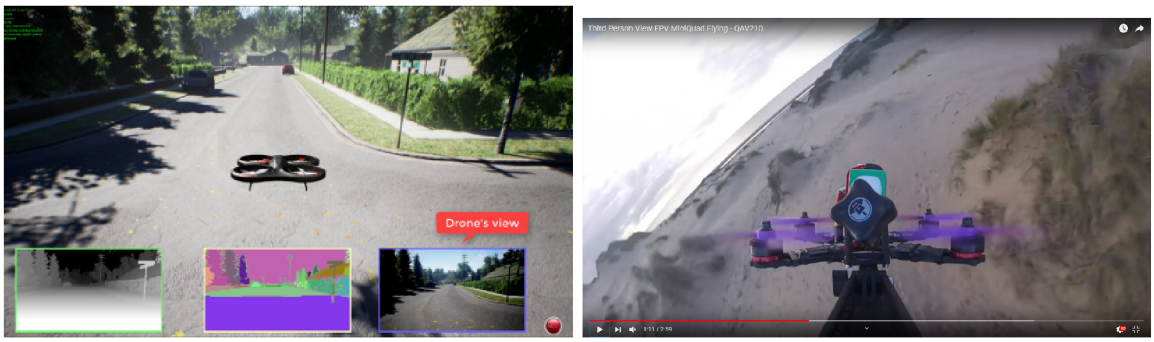
Další otázkou by bylo, jakým způsobem pilotovi video z 3D kamery zobrazit. To, že kamera vidí všemi směry totiž ještě automaticky neznamená, že všemi směry v každém okamžiku uvidí i pilot.

Proto se zkusme raději poohlédnout po řešení, které nebude tak náročné na přenos dat z dronu k pilotovi. Možnost použití více kamer a skládání jejich obrazů společně s vhodným způsobem zobrazení pilotovi ale určitě také stojí za prozkoumání a v budoucnosti by mohlo přinést velice zajímavé výsledky.

### Pohled třetí osoby

Řízení kvadrokoptéry v simulátoru (viz kapitola 2.9) má výhodu v tom, že je možné využít namísto pohledu 1. osoby (FPV) tzv. third person view – tedy pohled třetí osoby, kde je vidět také dron samotný a mnohem lépe jeho okolí. Pokud navíc ještě kamera zůstane do jisté míry zafixována tak, aby byla vždy umístěna za dronem, řízení se značně zjednoduší. Pilot totiž vidí přirozeným způsobem také to, jak se dron naklání vůči horizontu a náklon multikoptéry přímo souvisí s jejím pohybem. V určitých situacích a zejména pak během startu a přistání by se také hodil pohled shora na dron.

Získat krásný pohled na dron je však značně technicky složité, pokud kameru umístíme na tyč, zvyšuje se tím zátěž (dron s sebou nese další tyč a kameru). Tyč také může být jen poměrně krátká (převažování dronu dozadu) a z důvodu vibrací také musí být z relativně tuhého materiálu, který je obvykle těžký. Problém s vibracemi s délkou tyče narůstá a byla by tak nutná kamera s mnohem kvalitnější stabilizací. Přesto bylo toto řešení několika uživateli úspěšně vyzkoušeno a na dostupných videích je vidět, že tento koncept funguje[3] (obrázek 3.2).




Obrázek 3.2: Pohled třetí osoby — **Vlevo:** Screenshot ze simulátoru AirSim. S pohledem třetí osoby a několika dalšími typy pohledů, v podobně vravo dole FPV pohled. **Vpravo:** Pohled třetí osoby s využitím kamery na „selfie tyči“.

Ryosuke Murata a kolektiv [16] zase využili pro získání pohledu třetí osoby tzv. past image record system (SPIR), tedy systém ve kterém je obraz vizuálně generován přidáním 3D modelu robota do obrazu pořízeného kamerou na tomto robotu v minulém okamžiku – tedy v době, kdy robot viděl na místo, kde se momentálně nachází. Díky tomuto systému byl operátor tohoto robotického manipulátoru schopen zvládnout daný úkol výrazně rychleji a přesněji. Tento systém má však nevýhodu v tom, že je takto možné získat pohled pouze z místa, kde se robot v minulosti již nacházel, to však nemusí být vždy ten nejvhodnější pohled.

[3] Pohled třetí osoby na videu: https://www.youtube.com/watch?v=F7G7uH36y-c

Existuje však ještě jedna cesta, jak pohled třetí osoby získat – využít možností rozšířené virtuality a právě touto možností se zabývají další kapitoly této práce.

## 3.3 Využití rozšířené virtuality

### Rozšířená virtualita

Rozšířená virtualita je určitým protipólem známějšího pojmu rozšířená realita. U rozšířené reality je obraz z kamery rozšiřován o určité virtuální prvky. Rozšířená virtualita naopak vkládá do virtuální scény prvky z reálného světa.

V tomto případě tedy bude z volně dostupných mapových dat vytvořena 3D scéna, do které bude doplněn model dronu, který se bude pohybovat a naklánět na základě dat přenášených z dronu reálného. K tvorbě této scény je možné využít volně dostupná mapová data. Do této scény následně může být doplněn obraz z kamery, který obsahuje na rozdíl od statického modelu také pohybující se objekty a především je u něj vždy jistota, že je aktuální. Návrh scény je zobrazen na obrázku 3.3.


Obrázek 3.3: Návrh aplikace, využívající rozšířenou virtualitu. Scéna obsahuje 3D model města z aplikace Google Maps. Uprostřed se nachází online obraz z kamery (záměrně výrazně kontrastnější) a model dronu, který poměrně přirozenou cestou uživateli poskytne představu o okolí letounu a také jeho náklonu a pohybu.

V navržené aplikaci využívající rozšířenou virtualitu bude pilot moci řídit stejně jako doposud - tedy pomocí FPV (first person view), kdykoliv ale bude mít možnost kameru odzoomovat a plynule tak přejít na TPV (third person view) a uvidí kromě obrazu z kamery také dron samotný a jeho okolí, které už kamera nevidí (obrázek 3.3). Díky tomu je možné výrazně rozšířit pilotovo zorné pole (field of view) a umožní mu to rozhlížet se po

okolí. Podobný koncept rozšíření zorného pole pro piloty vojenských bezpilotních letounů představil už v roce 2005 Calhoun a kolektiv [11].

Kamery na dražších dronech poměrně často možnost zoomu mají a pilot má na ovladači slider pro jeho ovládání. V tomto případě však bude moci pilot zoomem obraz nejenom přiblížit, ale také oddálit a to až tak, že uvidí dron samotný a tedy i jeho náklon vůči horizontu i ostatním objektům.

Tato aplikace tedy bude kombinovat telemetrická data přijímaná z koptéry s video streamem z její kamery a mapovými daty načítanými online z internetu, případně offline z vlastního úložiště (obrázek 3.4). Tato data navíc ještě dále zkombinuje s dalšími informacemi získanými od uživatele.

### Data z dronu

Data, která bude nutné v reálném čase přenášet z dronu na pozemní stanici:

**Poloha** — Dvojice souřadnic ve formátu WSG84 (viz kapitola 2.6).

**Orientace dronu** — Úhel ve stupních získaný z elektronického kompasu na dronu.

**Klopení (pitch) a klonění (roll)** — Dvojice úhlů ve stupních nebo jako trojice i s úhlem z kompasu.

**Video z kamery** — online stream z kamery ve vhodném formátu. Video by mělo být komprimované.

**Úhel otočení kamery vůči dronu** — Trojice úhlů, říkající, jak je natočený gimbal – pokud je na dronu nainstalován.

**Další data z dronu** — stav baterie, rychlost letu, letový režim...

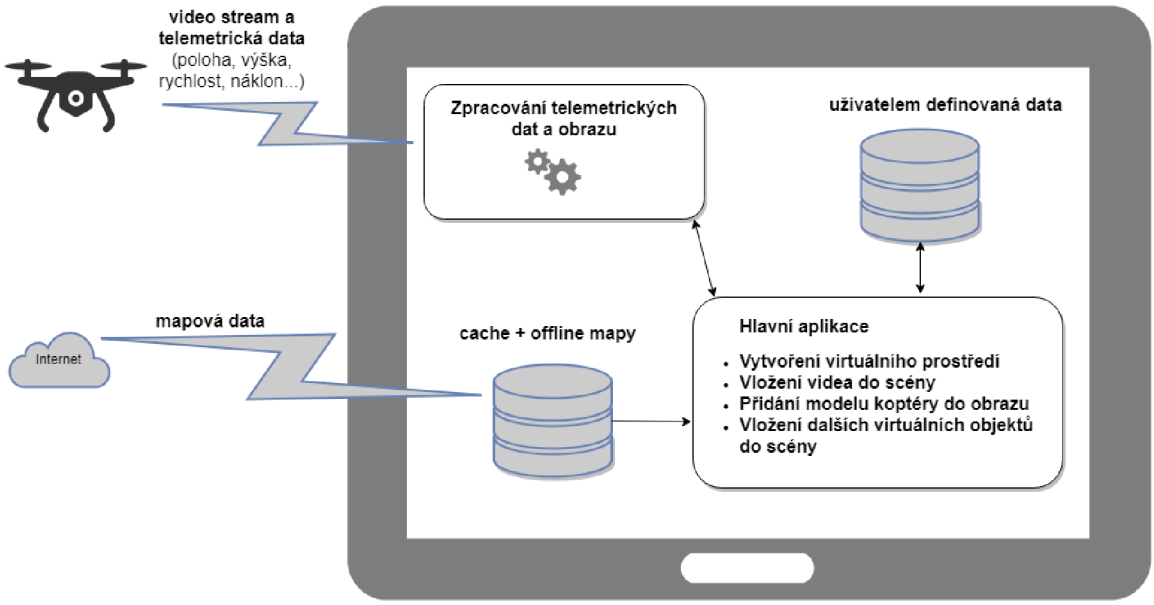
## 3.4 Návrh prostředí a ovládání

### Obraz z kamery

Obraz z kamery je možné ve scéně zobrazit několika různými způsoby. Prvním z nich je jeho promítnutí na obrovskou „virtuální obrazovku“. Tou může být buď rovina nebo vhodným způsobem zakřivená plocha. Velikost obrazovky se odvíjí od šířky zorného pole (FOV) fyzické kamery. Tato obrazovka se bude pohybovat vždy v určité fixní vzdálenosti před dronem a bude se otáčet a naklánět v závislosti na pohybu dronu a pokud je dron vybaven gimbalem, tak také podle pohybu jeho kamery. Druhou možností je promítání obrazu přímo na objekty ve scéně podobně, jako by na 3D modelu dronu byl umístěn video projektor.

### Pohyb po scéně

Aplikace by měla obsahovat jednoduchý vestavěný simulátor, který bude pilot moci využít při přípravě k letu, kdy je mnohdy vhodné si prostor proletět nejprve v simulátoru. Ten se může dále hodit při definování WayPointů a bezpečných oblastí, kde uživateli umožní pohybovat se ve 3D scéně přirozeným způsobem – tedy ovládat virtuální dron pomocí dvojice joysticků stejným způsobem jako ten skutečný.




Obrázek 3.4: Základní koncept aplikace pro pozemní stanici, která bude využívat rozšířenou virtualitu. Aplikace kombinuje telemetrická data z dronu s mapovými daty z internetu, které použije pro vytvoření 3D scény, ve které tato data z dronu budou zobrazena.

### Pohled do scény

Pohled do scény by měl být v základním režimu skrze kameru umístěnou vždy za dronem. Při tomto pohledu na dron pilot vždy ví, kde se nachází jeho příď. Podobně je možné využít také pohled z vrchu, kdy se kamera otáčí spolu s dronem a příď je tak vždy na horní straně obrazu, pilot ale velmi dobře vidí, co se nachází pod dronem. Mezi těmito pohledy by mělo jít plynule přecházet otáčením kamery a využít tak libovolné mezipolohy, kdy dron bude vidět např. z úhlu 45°. Spolu s kamerou virtuální by bylo dobré otáčet také kameru skutečnou tak, aby mířila podobným směrem a pilot měl tak co možná nejlepší přehled o aktuální situaci. To je samozřejmě možné pouze u dronů vybavených gimbalem.

#### Volná kamera

Existence virtuálního modelu prostředí, ve kterém se dron pohybuje, nabízí spoustu možností. Jednou z nich je možnost, podívat se do scény a na dron z úplně jiného pohledu a s kamerou tak manipulovat zcela volně. Mohlo by tedy být možné např. dron zastavit na bezpečném místě (dostatečně daleko od překážek), přepnout režim ovládání na vysílačce a pomocí joysticků stejným způsobem jako při řízení dronu odletět s virtuální kamerou a podívat se tak na dron a situaci kolem něj z libovolného místa, následně přepnout režim ovládání z kamery zpět na dron a pokračovat v letu s ním. Při letu jej však sledovat z nově zvoleného místa.

Na rozdíl od předchozího přístupu, kdy se kamera pohybuje vždy za (případně nad) dronem, v případě volné kamery už neplatí to, že se letoun nachází vždy v zorném poli kamery a to, že pilot vždy ví, kde se nachází příď. V tomto případě jde tedy o velmi podobnou situaci jako při řízení pohledem ze země se všemi negativy, která ho provázejí a to včetně situace, kdy dron vyletí ze zorného pole kamery a pilot jej tedy vůbec neuvidí. Kameru je sice možné automaticky natáčet tak, aby dron vždy zůstal v jejím zorném poli

podobně, jako se otáčí člověk, když sleduje letoun ze země. Pokud se však mezi dronem a virtuální kamerou bude nacházet překážka (např. model budovy), pilot letoun neuvidí. Dalším problém nastane v případě, že dron bude příliš daleko od kamery. V takovém případě ho pilot buď neuvidí vůbec a nebo ho uvidí pouze jako malý bod v dálce a nepozná tedy, jak je otočen.

Tento problém by šel vyřešit např. tím, že se kamera po dosažení určité maximální vzdálenosti začne pohybovat směrem k dronu a díky tomu tato vzdálenost nikdy nebude překročena. Pokud se dron naopak začne vracet, kamera se opět postupně vrátí na původní místo. Aby pilot vždy dobře viděl orientaci dronu, bylo by možné také zobrazit malý 3D model letounu dole na kraji obrazu, podle otočení tohoto modelu by pilot snadno poznal, jak je otočený dron, který jinak už vidí velmi malý.

### Využití FPV nebo VR brýlí

Dron je možné řídit pomocí FPV brýlí nebo brýlí pro virtuální realitu. Někteří piloti dokonce využívají head tracker, tedy zařízení, které sleduje pohyb hlavy a podle toho natočí kameru na koptéře. Pilot se tak může snadno a hlavně přirozeně rozhlížet po okolí. Zde je ale obrovským problémem latence - tedy zpoždění od otočení hlavy po otočení kamery, ze kterého se mnohým lidem dělá po chvíli nevolno.

Při využití v navrhované aplikace ale problém téměř zmizí. Když pilot otočí hlavu, pohled se okamžitě otočí a zobrazí tak uživateli příslušnou část 3D scény, obraz kamery se posune do středu zorného pole až dodatečně s tím jak se kamera skutečně otočí. Zajímavou možností by mohla být také možnost funkci otáčení kamery na chvíli vypnout rozhlížet se tak pouze po virtuální scéně (tedy trojrozměrné mapě) bez otáčení fyzické kamery.

Při využití VR brýlí se tedy pilot může virtuálně „pohybovat“ v určité vzdálenosti za dronem a mít možnost se plně rozhlížet po scéně. Rychlým pohybem hlavy směrem dolů tak např. může okamžitě zjistit, nad čím právě letí. Je tam opravdu stále ještě louka nebo už se nebezpečně přibližuje k dálnici nad kterou nesmí?

### Vizualizace překážek

Speciální oblastí je pak vizualizace dat ze senzorů. Některé drony jsou vybaveny hned několika sonary a dalšími senzory pro měření vzdálenosti od jiných objektů. V případě detekce objektu některým ze senzorů by bylo určitě vhodné tento objekt také ve scéně nějak zobrazit. Způsob zobrazení už samozřejmě záleží na typu senzoru, obecně se však dá říct, že známe alespoň přibližný směr a vzdálenost překážky. Přesný tvar obvykle neznáme. Jednou z možností, které se nabízejí, by tedy mohlo být zobrazení např. skupiny malých červených průhledných kuliček, které budou symbolizovat nějakou překážku a upozorní tak pilota, aby si dal pozor a případně tuto oblast zkontroloval pomocí kamery.

Dá se navíc předpokládat, že množství a kvalita senzorů v blízké budoucnosti neustále poroste, možnost vizualizace jejich dat tedy určitě stojí za prozkoumání.

## 3.5 Navigační prvky

### Hranice oblastí

Do scény také bude možné přidat jakákoliv další data. Např. pomocí poloprůhledných „virtuální zdí“ je možné zobrazit hranice zakázaných oblastí nebo prostorů letišť, kam dron v žádném případě nesmí vletět. Tyto hranice jsou často až od nebo do určité výšky a

proto tento způsob zobrazení nabízí daleko širší možnosti než prosté zobrazení ve 2D mapě. Podobným způsobem je možné vizualizovat také uživatelem definované bezpečné zóny, kam může letět bez obav ze střetu s překážkou.

Hranice mohou být ukládány ve formě polygonů geografických souřadnic ve formátu WSG84 doplněnými informacemi o minimální bezpečné nebo naopak maximální povolené výšce.

Tyto „virtuální zdi“ je možné také barevně rozlišit, aby pilot vždy viděl, do jaké oblasti se blíží. Hranice bezpečné zóny, kde se může pohybovat bez obav ze střetu s překážkou, mohou být např. zelené, zakázané prostory zase červené. Vhodná by také mohla být změna barvy při průletu touto hranicí. Pokud se tedy letoun nachází před vnější hranicí bezpečné oblasti, bude zobrazena zelenou barvou, pokud naopak vletí dovnitř, měl by hranici už vidět neutrální oranžovou barvou, protože za touto hranicí už bezpečná zóna není. Obdobně také hranice zakázané zóny by měla být červená, pokud už ale dron do této oblasti vletí, barva hranice by se měla změnit na oranžovou, za hranicí se už totiž zakázaná zóna nenachází. Současně by bylo také vhodné pilota výrazně upozornit (např. zvukovou výstrahou), že je v oblasti, kde být nemá, aby tuto oblast co nejrychleji opustil.

### Navigace v prostoru

Drony běžně umí létat tzv. mise – tedy autonomní let mezi několika předem naprogramovanými body (viz kapitola 2.3). Pokud je však nutné letět ve složitějším prostředí, ve kterém se nacházejí překážky, autonomní let není možný. Je však možné do scény přidat např.navigační šipky, které budou k těmto místům ukazovat, díky tomu pilot vždy bude vědět, kterým směrem má letět a jeho úkolem bude pouze se vyhnout překážkám. Tyto WayPointy mohou být umístěny nejen na zemi, ale také v určité konkrétní výšce ve vzduchu. Aplikace by také měla umět navigovat zpět k místu startu a pilot by měl mít možnost zobrazit obě navigační šipky najednou. Jedna bude ukazovat k místu startu, druhá k aktuálnímu WayPointu. Kromě šipek by měla být vždy zobrazena také vzdálenost k danému bodu.

## 3.6 Volba technologií

### 3D engine a vývojové prostředí

Jako první bylo nutné zvolit vhodné vývojové prostředí a engine, ve kterém bude aplikace vytvořena. Vzhledem k tomu, že bylo nutné zajistit, aby bylo možné co nejjednodušší propojení s mapami a případně také se simulátorem, bylo zvoleno prostředí Unity.

Unity je multiplatformní herní engine od společnosti Unity Technologies. Aplikaci vytvořené v Unity je možné přeložit pro PC, Linux, Mac, herní konzole, mobilní zařízení a dokonce i web [6]. Velkou výhodou je také kvalitní dokumentace a možnost využití zdarma pro nekomerční účely. Pro platformu Windows pak existuje editor, ve kterém je možné aplikace vyvářet z připravených komponent, případně využít Unity Asset Store a přidat tak do aplikace další komponenty. K dispozici je jich obrovské množství, některé jsou dostupné zdarma, velká část jich je však placená.

Kromě grafického prostředí pro tvorbu Unity podporuje také tvorbu scriptů v jazyce C#, díky kterému je možné vytvářet další komponenty.

### Zdroj mapových dat

Dalším krokem bylo nalezení vhodných volně dostupných mapových dat. Výsledná mapa musí obsahovat satelitní i běžnou mapu a především 3D terén. Pro každé místo na zemi musí být možné snadno zjistit jeho nadmořskou výšku. Dále by mapa měla obsahovat alespoň nějaké hrubé 3D modely budov. Vzhledem k důležitosti mapových dat byl částečně přizpůsoben výběr platformy a použitého 3D enginu právě dostupným mapám, tak, aby bylo co nejvíc vyžadovaných funkcí obsažených v API těchto map a nebylo nutné je dodělávat zvlášť, což by bylo velice časově náročné.

Jak již bylo řečeno v kapitole 2.10, Google Maps mají jednoznačně nejkvalitnější 3D budovy včetně velmi kvalitních textur. Existuje dokonce plugin pro prostředí Unity, Google jej ale zatím neposkytuje třetím stranám – alespoň zatím. V době dokončování této práce se velmi zkvalitnily také 3D mapy u českých map Mapy.cz, ty ale poskytují jen značně omezené API pro vývojáře.

Vzhledem k tomu, že pravděpodobně žádná jiná firma takto kvalitní 3D model celého světa nemá, bude se naše aplikace muset omezit na poněkud horší řešení založené na datech, která k dispozici jsou. Tedy na satelitní mapu, relativně kvalitní terén, ale jen velmi hrubé modely budov bez reálných textur. Je však pravděpodobné, že Google tato svá data časem uvolní i pro jiné účely než jsou hry a pokud se tento model ovládání osvědčí, 3D mapy z Google Maps nebo případně Mapy.cz mohou tuto aplikaci v budoucnosti posunout na úplně jinou úroveň.

Pro naši aplikaci tedy budou v současné době nejvhodnější mapová data dostupná ve formě pluginu pro nějaký 3D engine, který umožňuje přidávání jakýchkoliv dalších 3D objektů do scény a libovolnou manipulaci s kamerou. Další hledání tedy bylo omezené především na pluginy pro UnrealEngine a Unity. Výběr se tedy zmenšil na dva hlavní kandidáty Online Maps for Unity a MapBox for Unity (viz kapitola 2.10.2). Vzhledem k tomu, že požadavky nejlépe splňoval **MapBox for Unity**, byla nakonec zvolena právě tato aplikace. Tím byl také definitivně potvrzen výběr platformy Unity.

### Vestavěný simulátor

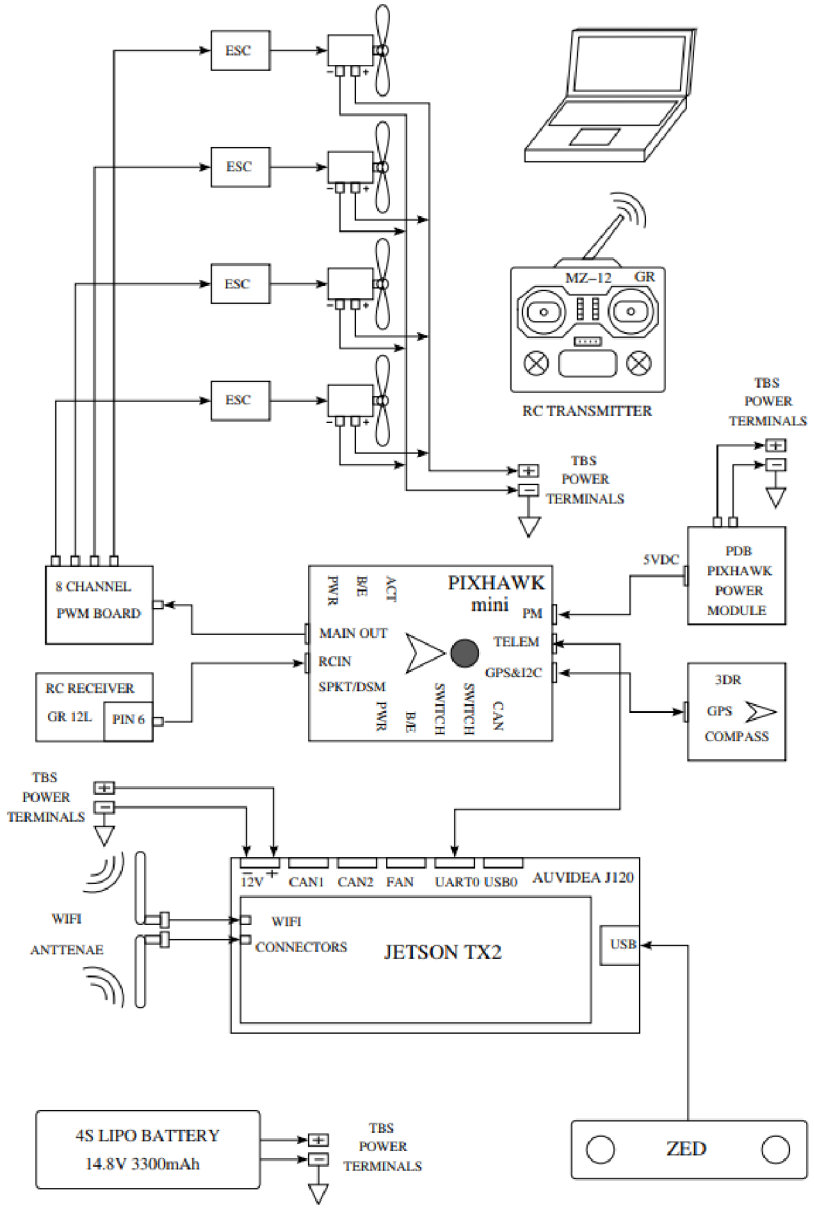
Po podrobném nastudování a vyzkoušení pokročilejších simulátorů (viz kapitola 2.9) se ukázalo, že takto reálný simulátor vlastně vůbec není potřeba. V aplikaci bude simulace používána především pro pohyb 3D prostorem stejným způsobem jako s reálným dronem. Aplikace je však určena pro ovládání velice stabilního dronu (reálný samozřejmě bude o něco horší), který sám nikam necestuje, neovlivňuje ho vítr, setrvačnost ani žádné další vnější vlivy, na ovládání reaguje velice přesně a nepotřebuje umět žádné další letové režimy. Takový dron se simuluje velice snadno. Proto bylo rozhodnuto vytvořit vlastní jednoduchý simulátor, který pouze převede vstup z herního ovladače se 2 joysticky na pohyb dronu prostorem.

## 3.7 Propojení s dronem

Poslední velmi důležitou věcí byl výběr komunikačních protokolů pro propojení s dronem. Vzhledem k tomu, že ale dlouho nebylo jasné, na jakých technologiích bude založen testovací dron, nechával se výběr komunikačního protokolu až na pozdější fázi projektu. Ten je totiž přímo závislý na výrobci dronu, použité řídící jednotce a softwaru. Nakonec byl zvolen pře-

nos dat přes WiFi pomocí protokolu ROSbridge. Díky použití WiFi je však značně omezena maximální vzdálenost přenosu, výhodou je naopak relativně vysoká datová propustnost.

Samotný dron obsahuje řídící jednotku Pixhawk se softwarem PX4 Autopilot (viz kapitola 2.7), miniaturní superpočítač Nvidia Jetson TX2 s operačním systémem Linux Ubuntu, stereoskopickou kameru, GPS a kompas (obrázek 3.5).




Obrázek 3.5: Architektura testovacího dronu. Dron používá řídící jednotku Pixhawk se softwarem PX4, která je pomocí portu telemetrie propojena se superpočítačem Nvidia Jetson TX2. Dron je dále vybaven stereoskopickou kamerou ZED a dvojicí WiFi antén, GPS a kompasem [9].

Řídící software PX4 autopilot využívá komunikační protokol MAVlink. Tento point-to-point protokol byl navržen přímo pro komunikaci s bezpilotními systémy a pro vzájemnou komunikaci komponent na jejich palubě [8]. Protokol MAVlink také podporuje velká většina

pozemních stanic (viz kapitola 2.5). Na počítači Nvidia Jetson běží robotický operační systém ROS a data z flight controlleru jsou převáděna na MavRos – tedy protokol MAVlink v ROSu.

### Robotický operační systém – ROS

Robotický operační systém ROS není ani přes svůj název operačním systémem, ale jde o robotický middleware (kolekci softwarových nástrojů a knihoven pro vývoj robotů), jejichž cílem je zjednodušení vytváření komplexních robotů, Poskytuje služby určené pro heterogení počítačový cluster, abstrakci od hardwaru a předávání zpráv mezi procesy [1].

ROS poskytuje komunikační systém mezi jednotlivými distribuovanými uzly systému. Využívá architekturu publisher/subscriber. Hlavní uzlem je tzv. ROS-master. Uzly, které produkují nějaká data (např. senzor nebo kamera) se přihlásí jako publisher. Uzel, který má naopak o tato data zájem se přihlásí jako subscriber. ROS-master udržuje seznam dostupných topiců a zprostředkovává komunikaci mezi jednotlivými uzly.

#### ROS přes WebSocket — ROSbridge

Rosbridge poskytuje JSON API pro programy, které nejsou přímo založené na ROSu. Obvykle jde právě o frontend aplikace. ROS-bridge server v systému vystupuje jako standardní ROS node, který komunikaci předává přes protokol WebSocket dalším aplikacím ve formátu JSON. Jediným povinným polem zprávy je pole `op`, které určuje činnost této zprávy (nejčastěji tedy publish/subscibe) [5].

Přihlášení k odběru dat z kompasu ve formátu JSON:

```
{
    "op": "subscribe",
    "topic":"/mavros/global_position/compass_hdg"
}
```

Odpověď je zpráva typu `publish`, která obsahuje úhel ve stupních, zde tedy dron směřuje přibližně na VSV (východoseverovýchod):

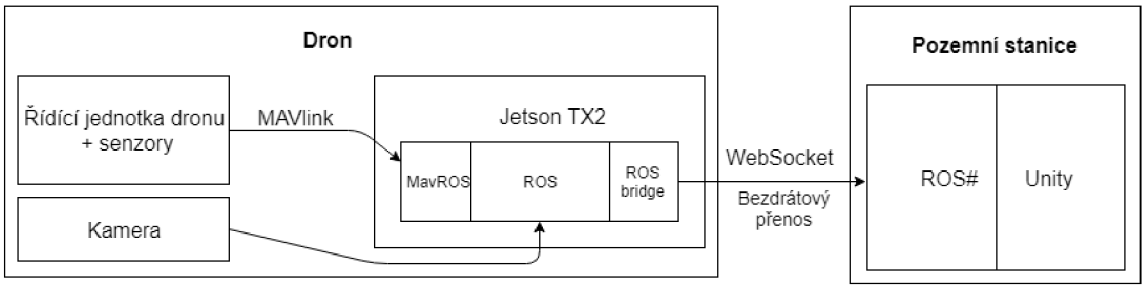
```
{
    "topic": "/mavros/global_position/compass_hdg",
    "msg":
    {
        "data": 57.29
    },
    "op": "publish"
}
```

### ROSbridge v Unity — RosSharp

Unity je herní engine a jeho využití pro jiné typy aplikací je tedy poměrně neobvyklé, při hledání knihovny pro propojení s ROSem byl tedy výběr značně omezený. Téměř jedinou možností bylo použití knihovny ROS# (RosSharp) a její následné doplnění o nové typy zpráv a další menší úpravy. ROS# je sada open source softwarových knihoven a nástrojů v C# pro komunikaci s ROS z .NET aplikací, zejména Unity [10]. Autorem knihovny ROS# je Dr. Martin Bischoffa. Knihovna je distribuována jako plugin pro Unity prostřednictvím

Unity Asset Store a nebo GitHubu projektu[4], kde je k dispozici vždy aktuální verze. Při první instalaci této knihovny (únor 2019) z Unity Asset Store docházelo k potížím s komunikací. Během práce na této DP ale na knihovně probíhal velmi intenzivní vývoj a po instalaci novější verze z GitHubu tyto problémy zmizely. Určitou nevýhodou této knihovny je nepříliš kvalitní dokumentace, zdokumentovány jsou pouze základní operace a nejběžnější použití komponent, knihovna však obsahuje i spoustu pokročilejších funkcí, jejichž přítomnost je však možné zjistit pouze studiem zdrojového kódu.

Schéma, jak probíhá komunikace skrze jednotlivé protokoly a knihovny je zobrazeno na obrázku 3.6.




Obrázek 3.6: Komunikační protokoly mezi dronem a aplikací. Na straně aplikace zprostředkovává komunikaci knihovna ROS#, která komunikuje s dronem přes ROSbridge, který je přenášený přes protokol WebSocket.

[4]GitHub projektu ROS# https://github.com/siemens/ros-sharp/

# Kapitola 4

# Tvorba aplikace

## 4.1 Mapy a terén

Pro instalaci MapBox for Unity je nutné si nejprve vytvořit účet na webových stránkách projektu[1], vygenerovat autentizační token a stáhnout balíček do počítače. Následně bylo možné pomocí nabídky `Assets > Import Package > Custom Package` přidat MapBox do projektu a zadat vygenerovaný autentizační token. Balíček je poměrně velký a obsahuje značné množství příkladů, ty ale nebylo nutné importovat. Základním objektem, který bylo nutné přidat do scény je prefab (předpřipravený objekt) `Map`. Ten ale obsahuje obrovské množství parametrů, které je nutné správně nastavit. Nastavení mapy je zobrazeno na obrázku 4.1. Mapová aplikace ale obsahuje značné množství chyb a ne vše funguje úplně správně. Aktualizace ale vycházejí jen velmi zřídka.

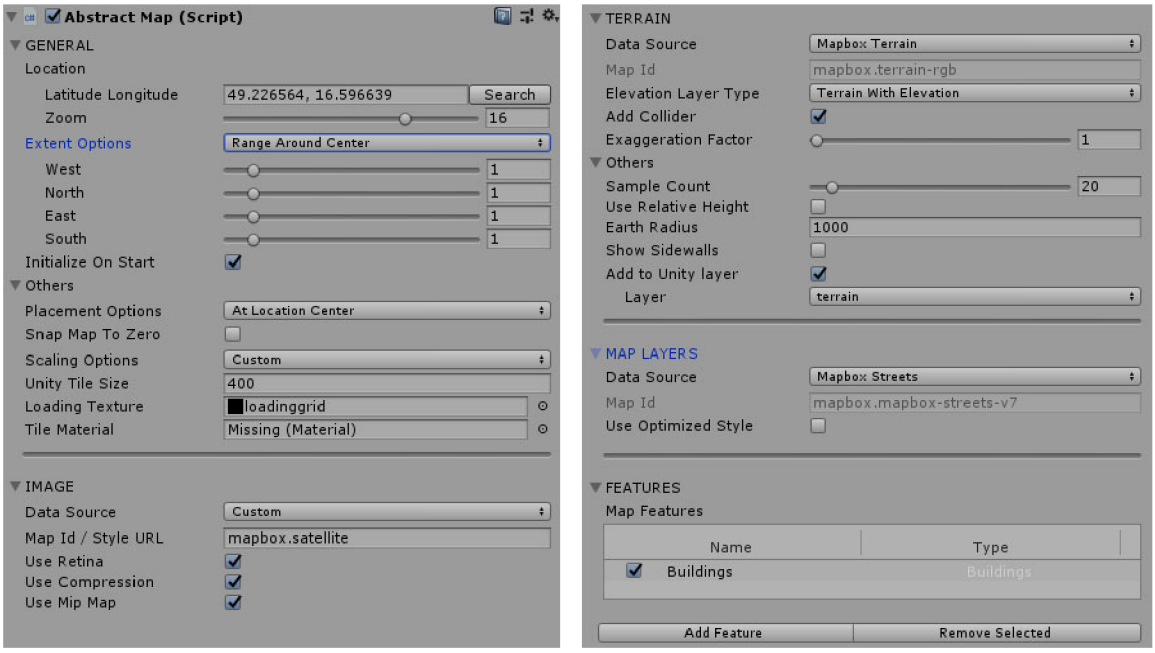
Jak je vidět na obrázku 4.2, mapa je složena ze čtvercových dlaždic, které jsou při maximálním dostupném rozlišení mapy veliké 400x400m. MapBox podle dokumentace umí načítat dlaždice v závislosti na pohybu kamery tak, aby vždy byla viditelná oblast pokryta dlaždicemi. Tato funkce však v MapBoxu při určitém náklonu kamery nefunguje zcela správně a mnohdy se tak stávalo, že i téměř 1/2 zorného pole nebyla pokryta. Druhým problémem tohoto průběžného načítání bylo to, že aplikace při pohybu kamery, který vyvolal potřebu načítat větší množství dlaždic, najednou na malou chvíli zamrzla, což u aplikace, která má sloužit pro řízení dronu v reálném čase není žádoucí.

Z tohoto důvodu bylo použito statické zobrazení určitého počtu dlaždic bez průběžného načítání. Dlaždice jsou načteny pouze jednou při startu aplikace. Načítání probíhá v první řadě z cache, pokud je ale daná oblast načítána poprvé, musí se načíst z internetu. Velikost mapy (počet dlaždic) je možné nastavit v konfigurační scéně (viz kapitola 4.6). Při nastavení velikosti 4 dlaždic na každou světovou stranu vznikne mapa velikosti 3600x3600m se středem v místě startu, což je pro pohyb běžných dronů naprosto dostatečná velikost.

### Souřadnice a nadmořská výška

Dalším úkolem, který bylo nutné vyřešit byl převod polohy z dronu do souřadnicového systému Unity, které používá kartézský souřadnicový systém, u kterého jsou souřadné osy vzájemně kolmé přímky, které se protínají v jednom bodě – počátku soustavy souřadnic. Jednotkou vzdálenosti je 1UU (Unity units). Dron oproti tomu používá polohu zadanou pomocí své a aktuální nadmořské výšky a geografických (zeměpisných) souřadnic ve formátu WSG84 – tedy zeměpisné šířky a délky (viz kapitola 2.6).

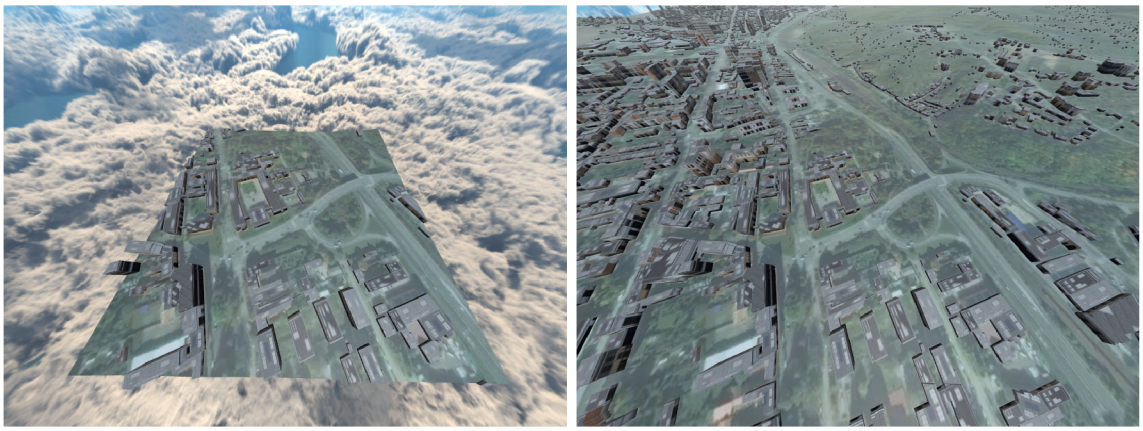
[1] Web projektu MapBox for Unity: https://www.mapbox.com/install/unity/




Obrázek 4.1: Základní nastavení mapy: v sekci GENERAL se nastavuje střed mapy, zoom, velikost dlaždic a jejich počet ve všech čtyřech směrech. Sekce IMAGE nabízí výběr z několika typů map. TERRAIN umožňuje nastavit, zda má být mapa modelována jako plochý terén, glóbus a nebo plastický terén – ten využije naše aplikace. Dále je zde možné určit, do jaké Unity vrstvy bude terén patřit, toho bude později využívat rayCast pro zachytávání kliknutí do mapy. V sekci MAP LAYERS je možné nastavit zdroj dalších dat. To je velice důležité pro vytvoření 3D budov, k dispozici je více možností, každá má ale jinak kvalitní data v různých částech světa. Budovy se přidávají v sekci FEATURES.

Mezi těmito souřadnými systémy musí aplikace umět informace o poloze převádět. Dále musí být také pro libovolnou souřadnici na povrchu země schopná určit nadmořskou výšku tohoto místa, aby bylo možné vypočítat výšku nad zemí z aktuální nadmořské výšky dronu. Obě tyto funkce jsou přítomny přímo v balíčku MapBox.

Objekt `Map` (objekt s celou mapou) obsahuje vždy script třídy `AbstractMap`. Tato třída slouží jako určité rozhraní mapy a nabízí sadu užitečných veřejných metod, které aplikace založené na poloze mohou potřebovat. Za zmínku stojí především metody pro převod mezi souřadnými systémy `GeoToWorldPosition` a `WorldToGeoPosition`, metoda pro zjištění nadmořské výšky terénu `QueryElevationInUnityUnitsAt`, která vrací výšku v Unity Units a `QueryElevationAtInternal` pro získání výšky v metrech. Tyto metody ale ne vždy vracely správné hodnoty. V knihovně jsou zřejmě chyby. Tento problém však bylo možné vyřešit změnou velikosti dlaždice tak, aby 1UU odpovídala 1m. Stačilo tedy nastavit velikost dlaždice na 400 namísto výchozí hodnoty 100. Toto měřítko je samozřejmě výhodné i z mnoha dalších důvodů a vzhledem k tomu, že operace v kartézských souřadnicích jsou mnohem jednodušší, je většina výpočtů vzdáleností a směrů k objektům prováděna právě v tomto souřadnicovém systému.



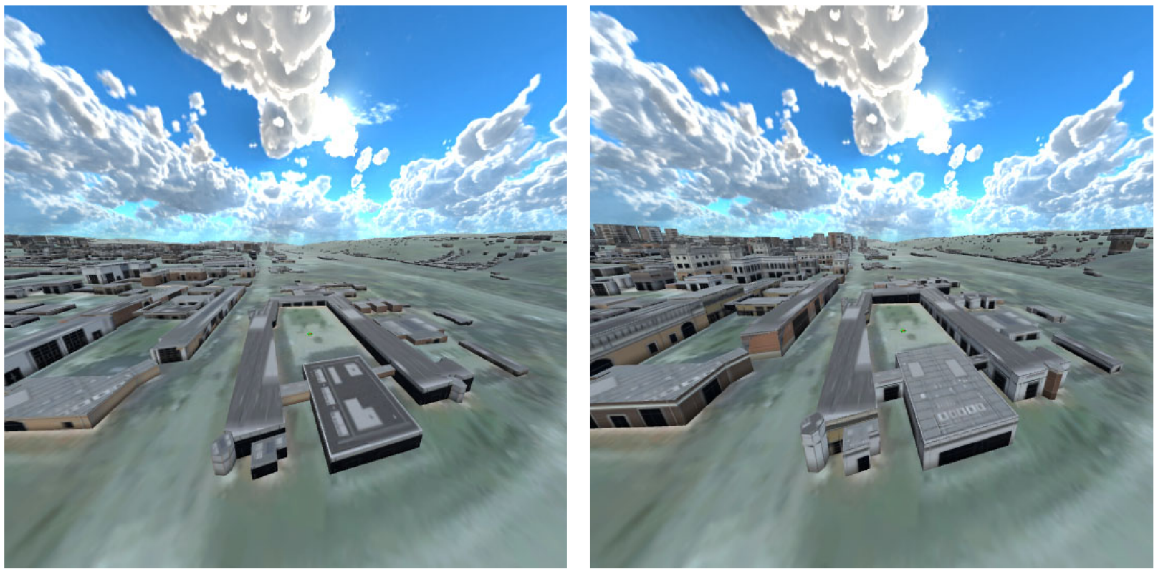
Obrázek 4.2: Mapa je složena z jednotlivých dlaždic. Každá dlaždice je velká 400x400m. VLEVO: jediná dlaždice s areálem FIT VUT, za hranicemi dlaždice je vidět skybox. VPRAVO: stejná oblast s dalšími přilehlými dlaždicemi.

### Budovy

Jednoduché 3D modely budov jsou přímo součástí MapBoxu. I zde však bylo nutné zkoušet spoustu možností nastavení, aby vše co nejlépe odpovídalo potřebám této aplikace. Při rozhodování, který mapový framework zvolit, to vypadalo, že MapBox obsahuje jen velmi málo 3D budov. Ve výchozím nastavení je totiž nastavená datová vrstva `MapBox Streets With Buildings Ids`, která pravděpodobně kvalitněji zobrazuje jiné části světa, v oblasti Brna však obsahuje pouze několik desítek budov, což rozhodně není dostačující. Naštěstí je možné tuto vrstvu změnit za `MapBox Streets`, které už i v prostředí českých měst obsahuje naprostou většinu budov. Nutno ale dodat, že kvalita dokumentace není příliš velká a u tohoto nastavení není také příliš jasné, že souvisí s budovami.

Budovy v MapBoxu (alespoň v českých podmínkách) nejsou plnohodnotnými 3D modely, ale pouze polygony odpovídajícího tvaru vytažených do určité výšky od země. Všechny budovy jsou tedy modelovány tak, jako by měly rovnou střechu. Informace o výšce budov ale nejsou příliš přesné. Obvykle jsou budovy tedy o něco nižší než ve skutečnosti (možná také vlivem chybějících sedlových střech). Vzhledem k tomu, že budovy v této aplikaci jsou především potenciálními překážkami, je lepší je modelovat raději o něco vyšší než ve skutečnosti jsou. Pokud by pak pilot chtěl létat v jejich okolí (bez ohledu na platnou legislativu), bude pro bezpečnost letu lepší, když v modelu budou zobrazeny raději o něco vyšší než jsou ve skutečnosti, aby se jim pilot bezpečně vyhnul. Z tohoto důvodu byl nastaven `Scale Factor` budov na hodnotu 1,9 (viz obrázek 4.3). Díky tomu výška budov lépe odpovídá realitě. U většiny budov by bylo lepší nastavit `Scale Factor` ještě o něco větší, budovy, které ale mají výšku uvedenou správně, by potom byly příliš vysoké.

Dalším krokem je volba textury budov. MapBox obsahuje několik typů textur nebo umožňuje použít texturu vlastní. První možností je využití textury `Satellite`, kde by měla být na střechu bodov umístěna satelitní fotka této střechy. Toto texturování však funguje zcela špatně a na střechu je tak umístěna úplně jiná část satelitní fotografie – obvykle tedy zeleň z okolí budovy. To by bylo pro pilota zcela matoucí a proto tedy tuto texturu nebylo možné použít. Další možností jsou pak jednobarevné světlé nebo tmavé materiály. Ty se ale absolutně nehodí k použité satelitní mapě. Třetí možností je pak textura pojmenovaná



Obrázek 4.3: Výška budov: **vlevo:** Budovy areálu FIT VUT se základní výškou z dat MapBoxu. **vpravo:** Budovy 1,9x zvětšené už skutečnosti odpovídají trochu víc. Reálně by bylo třeba je zvětšit ještě víc, budovy, u kterých je ale výška správná, by byly příliš vysoké.

`Realistic` (obrázek 4.4). Naneštěstí ani ta příliš realistická není a většina budov tak vypadá spíš jako průmyslové haly. Tato textura je však z dostupné nabídky stále nejlepší volbou.

Poslední věci, kterou bylo nutné stejně jako u terénu vyřešit, je to, do jaké Unity vrstvy budou budovy patřit, toho bude později využívat ReyCast pro zachytávání kliknutí do mapy a kamery pro nastavení viditelnosti jednotlivých vrstev (viz kapitola 4.5).

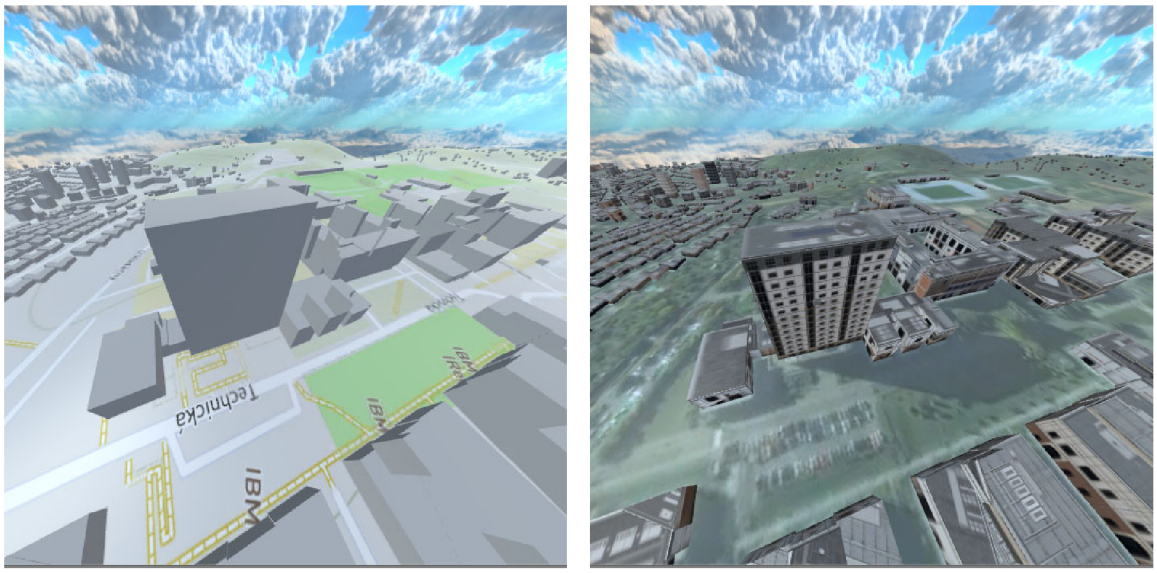
## 4.2 Dron a scéna

V Unity je možné objekty hierarchicky řadit do kontejnerů pojmenovaných `GameObject`, pokud se aplikuje transormace na kontejner, automaticky je aplikována na všechny objekty v tomto kontejneru. Takto je možné hierarchicky skládat transformace.

Ve scéně jsou jsou objekty pevné (terén, budovy, WayPointy) a objekty pohyblivé (dron, virtuální obrazovka, kamera pro sledování scény, navigační šipky...). Z toho důvodu jsou všechny pohyblivé objekty zařazeny v kontejneru `DroneObject` a právě ten je posouván po scéně na základě aktuálních dat o poloze. V tomto kontejneru jsou dále řazeny další objekty – navigační šipky, kontejner `CameraBox` s kamerami pro pohled do scény a nejdůležitější kontejner `ModelPlusVideo`. Ten obsahuje, jak jeho název napovídá, samotný model dronu, virtuální obrazovku a projektor videa.

### Data o pohybu dronu

Samotný pohyb kontejneru `DroneObject` (a tedy modelu dronu a všech dalších objektů, které se pohybují s ním) po scéně zajišťuje script `DroneController`, který zároveň slouží jako ústřední propojovací článek celé aplikace. Data, kde se dron aktuálně nachází, je možné získat z tříd odvozených od abstraktní třídy `AbstractDroneData`. Ta obsahuje metodu `GetPosition()` vracející aktuální polohu dronu v Unity units, dále `GetPitchRoll()`, která vrací hodnotu náklonu dronu ve stupních a metodu `GetRotation()`, která vrací hodnotu




Obrázek 4.4: Textury budov – ukázáno na budově FSI VUT v Brně: **vlevo:** Budovy s jednobarevnou texturou. Jako textura terénu je použita klasická mapa. **vpravo:** Budovy s texturou „Realistic“. Textura sice příliš realistická není, i tak ale poskytuje pilotovi mnohem lepší možnosti prostorové orientace a odhadu výšky. Jako textura terénu je použita satelitní mapa.

rotace dronu (také ve stupních) získanou z elektronického kompasu. Poslední důležitou metodou je pak `GetCameraRotation()`, která vrací úhly natočení kamery vůči dronu v případě, že je dron vybaven gimbalem.

### Náhodný pohyb dronu

V první fázi prací na aplikaci bylo nutné nějak simulovat pohyb dronu prostorem. Proto byla vytvořena třída `DroneData` odvozená od již zmíněné třídy abstraktní. Tato třída v každém kroku připočítá náhodné číslo k hodnotám zrychlení ve všech 3 osách a hodnotě rotace kolem svislé osy. Tyto hodnoty jsou pak přičteny k aktální pozici dronu a podle toho zrychlení je také příslušně upraven náklon letounu.

### Manuální řízeni pohybu — jednoduchý simulátor

Jak již bylo řečeno v kapitole 3.6, součástí aplikace je i jednoduchý simulátor. Jeho implementace je obsažena v jedné jediné třídě `DroneDataManual`, která je také odvozena od třídy `AbstractDroneData`. Rychlost pohybu letounu v jednotlivých osách je přímo čtena z hodnot výchylek ovladače pomocí standardní Unity metody pro čtení ze vstupu `Input.GetAxis("Throttle")`. Tato hodnota je dále ještě upravena tak, aby ve větší výšce nad zemí simulovaný dron letěl výrazně rychleji než u země a pohyb po scéně tak uživatele nezdržoval. V malé výšce je naopak vyžadována větší přesnost ovládní. Aplikace dále hlídá, aby nadmořská výška dronu nikdy nemohla být menší než 0 a simulovaný dron se tak nedostal pod texturu terénu.

Pro ovládání simulátoru byl použit jednoduchý herní ovladač. V produkční verzi by však bylo nutné aplikaci propojit přímo s RC vysílačkou, aby uživatel nemusel používat jiný ovladač pro simulátor a pro skutečný dron a pouze nějakým vhodným způsobem přepínat

mezi režimy. Vzhledem k tomu, že bezdrátové připojení vysílačky tak, aby bylo možné číst hodnoty výchylek řízení, je poměrně obtížné, v této fázi projektu bylo rozhodnuto o použití herního ovladače, což bylo výrazně jednodušší.


Obrázek 4.5: Gamepad Genius Grandias 12 má stejně jako modelářské vysílačky 2 joysticky, proto je pro ovládání simulároru vhodný.

#### Data z reálného dronu

Třída `DroneRosData` poskytuje data z reálného dronu, k tomu potřebuje předat pomocí metody `setRosConnector` instanci ROS konektoru, který spravuje připojení k ROSbridge. Třída `DroneRosData` samotná toto připojení nevytváří ani nespravuje, pouze čte data z jeho komponent, převede data do jednotného tvaru a předá je dál kontroleru modelu dronu. Dron posílá data o poloze prostřednictvím ROS topicu `/mavros/global_position/global`, ten je ve formátu JSON se zprávou typu `NavSatFix` (viz kapitola 4.3). Obsahuje tedy nějaké informace o stavu GPS a především pro nás nejdůležitější atributy `latitude`, `longitude` a `altitude`. Komponenta konektoru tato data při příchodu zprávy uloží do své veřejné proměnné. Instance třídy `DroneRosData` pak data při vykreslování každého snímku načte z těchto veřejných atributů, převede je do souřadného systému Unity (viz kapitola 4.1) a vrátí DroneControlleru. Podobným způsobem tato třída pracuje také s daty z IMU jednotky a kompasu. IMU data dron posílá ve formě kvaternionu. Ten je převeden na úhly ve stupních a je nutné je o 90° otočit okolo svislé osy, aby odpovídalo mapování na osy roll a pitch (viz obrázek 2.1).

## 4.3 Připojení k ROSbridge

Připojení k ROSbridge zajišťuje knihovna ROS#, která musí být přidána jako komponenta k nějakému `Unity Game Objectu` – v tomto případě tedy k prefabu (předpřipravenému objektu) pojmenovanému `DroneRosConnector`. Jako prefab je tento objekt vytvářen proto, aby v případě ztráty spojení bylo možné tento objekt smazat a snadno vytvořit novou instanci a opětovně toto spojení navázat. Tento objekt obsahuje několik komponent (C# scriptů). První a nejdůležitější komponentou je script se stejným názvem jako celý prefab – tedy `DroneRosConnector`, což je pro tuto aplikaci speciálně upravená verze scriptu

`RosConnector` z knihovny ROS#, který zajišťuje samotné připojení k ROSbridge serveru. Díky úpravám script při startu automaticky načte uživatelem definovaná nastavení (např. URL ROSbridge serveru) a umožňuje také detekci nefunkčního spojení. To knihovní script neumí. V aplikaci, kde může být spojení snadno přerušeno např. velkou vzdáleností dronu a výpadkem signálu, by to však byl problém. Proto bylo nutné detekci ztráty spojení do scriptu doplnit. Vzhledem k tomu, že dron posílá zprávy neustále, bylo možné ztrátu spojení detekovat pouze na základě uplynutí časového limitu od přijetí poslední zprávy.

Dalšími komponentami tohoto objektu jsou jednotliví subscribeři. Každý subscriber čte jeden ROS topic, jehož jméno je nutné zadat do veřejné proměnné `Topic`, při vytvoření komponenty se subscriber automaticky přihlásí k odběru tohoto topicu. Zde tedy bylo nutné vytvořit a přidat subscribery pro polohu, IMU data, data z kompasu a obraz z kamery. Do budoucna by zde mohli přibýt subrcibeři dalších letových dat nebo např. publisher letových misí, aby bylo množné definované WayPointy přenést do řídící jednotky dronu.

Při přijetí zprávy je tato zpráva předána metodě subscribera `ReceiveMessage()` a ten ji dále zpracuje – obvykle pouze tak, že data uloží do veřejné proměnné, odkud si je později čtou další objekty, které tato data potřebují. Tato metoda musí mít správně nastavený typ přijímané zprávy, aby odpovídal zadanému topicu. Knihovna ROS# samozřejmě neobsahuje typy MavROS zpráv, proto bylo nutné je přidat.

### Přidání vlastního typu zprávy

Knihovna ROS# obsahuje jen několik základních typů zpráv Veškeré další typy je nutné doplnit. Jako příklad zde uvedu již zmíněnou zprávu typu `sensor_msgs/NavSatFix`, díky které jsou přenášena data o poloze. Tato zpráva byla právě jednou z těch, které bylo nutné doplnit. Definice jednotlivých datových polí je dostupná v oficiální dokumentaci [2]. Zprávy také často obsahují další zanořené datové typy, které je také nutné doplnit, dokud nebudou pro všechna datová pole použity definované datové typy.

```
public class NavSatFix : Message
{
    [JsonIgnore]
    public const string RosMessageName = "sensor_msgs/NavSatFix";

    public NavSatStatus status;
    public double latitude;
    public double longitude;
    public double altitude;
    public uint position_covariance_type;

    public NavSatFix()
    {
        status = new NavSatStatus();
        latitude = new double();
        longitude = new double();
        altitude = new double();
        position_covariance_type = 0;
    }
}
```

[2] Zpráva ROS typu NavSatFix: http://docs.ros.org/api/sensor_msgs/html/msg/NavSatFix.html

## 4.4 Obraz z kamery

Nejsložitější subscriber je bezpochyby `DroneImageSubscriber`, ten má za úkol zpracovávat přijaté snímky videa. Kromě něj obdobnou funkčnost poskytuje také druhý script `DroneImageRawSubscriber`, který naopak zpracovává video nekomprimované, se kterým bylo nutné pracovat během testování. Následně se ale ukázalo, že bez komprimace není možné dosáhnout přijatelné latence ani počtu snímků za vteřinu, proto bylo nutné přejít právě na komprimované video. Aby bylo možné přijímat oba formáty videa a přepínat mezi nimi, objekt `DroneRosConnector` obsahuje obě tyto komponenty a po startu v závislosti na uživatelském nastavení jednu z nich aktivuje a druhou odstraní.

Oba tyto scripty dostávají jako parametry, předávané pomocí veřejných proměnných rozlišení videa, příslušný ROS topic s videem a dvojici materiálů, na které mají toto video vykreslovat. První materiál je určen pro virtuální obrazovku, druhý pro projektor, který bude video promítat do scény. Každý způsob zobrazení videa má svoje specifika, proto nebylo možné video vykreslovat jednotným způsobem na jeden materiál. Dalším parametrem, který bylo nutné nastavit je tzv. `TimeStep`, který nastavuje dobu mezi snímky přijímaného videa a tím šetří potřebnou šířku pásma přenosového média. To bylo nutné omezovat zejména pro video nekomprimované, kde by jinak latence dosahovala i několika vteřin, což by bylo v praxi totálně nepoužitelné. Pokud je tato hodnota nastavena na nulu, video je přenášeno s maximální možnou frekvencí a pro komprimované video bylo možné tuto hodnotu bez problémů ponechat. Hodnotu `TimeStep` je možné nastavit v konfigurační scéně (viz kapitola 4.6), kde je ale přepočítána na maximální FPS.

Po startu komponenty je vytvořena 2D textura v zadaném rozlišení a přiřazena oběma materiálům. V případě optimalizace pro projektor musí mít textura nastavený `wrapMode` na `Clamp`, v opačném případě na `Repeat`. Textura pro projektor musí být dále ohraničena bílým rámečkem (viz obrázek 4.7). V případě nekomprimovaného videa musí být textura navíc ve formátu `BGRA32`, aby odpovídala formátu přenášeného obrazu.

Samotný příjem snímku videa opět probíhá přes metodu `ReceiveMessage()`, data jsou uložena do proměnné spolu s příznakem, že byla přijata zpráva ke zpracování. Následně je při vykreslování nového snímku v Unity zavolána metoda `Update()`, kde je v případě, že byla od zpracovávání posledního snímku přijata nová data, zavolána metoda `ProcessMessage()`, která překreslí texturu. To je prováděno pomocí metody `LoadRawTextureData()` samotné textury pro nekomprimované video a pomocí `LoadImage()` pro video komprimované, kde jsou jednotlivé snímky přenášeny ve formátu JPG.

### Virtuální obrazovka

Virtuální obrazovka s videem se pohybuje v určité vzdálenosti před dronem a otáčí se společně s fyzickou kamerou na gimbalu. Je tedy umístěna vždy tak, aby zobrazila to, co kamera vidí přesně tam, kde to ve skutečnosti je. Virtuální obrazovka je v současnosti řešena pouze jako plochý 2D obdélník, bylo by však možné použít i nějakým způsobem zakřivenou plochu, která by lépe odrážela vlastnosti fyzické kamery. Velikost obrazovky je vypočítána z FOV (Field of view) kamery a nastavené vzdálenosti od dronu. Tuto vzdálenost může uživatel nastavit. Vhodná vzdálenost je přibližně 20-50m, ale vzhledem k tomu, že je velikost obrazovky přepočítávána s ohledem na vzdálenost, její velikost z pohledu uživatele je stále stejná. Liší se pouze to, které další objekty (WayPointy, hranice oblastí...) uživatel uvidí. Viditelné totiž zůstanou pouze ty, které jsou před touto obrazovkou.


Obrázek 4.6: Virtuální obrazovka pro zobrazení videa se pohybuje v určité (uživatelem nastavené) vzdálenosti před dronem a otáčí se společně s fyzickou kamerou na gimbalu. Na obrázku je vidět netypický pohled zboku.

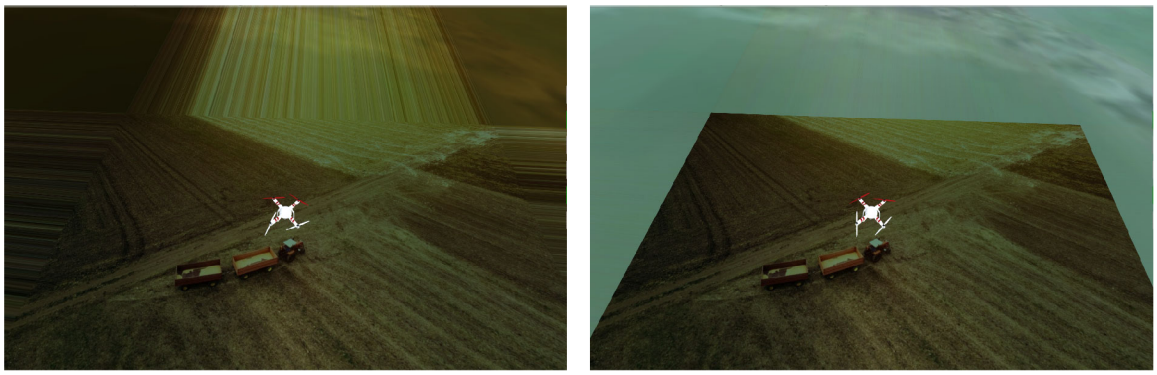
### 4.4.1 Projekce do scény

Projektor je součástí `Standard Assets` od Unity a umožňuje projektovat materiál na všechny objekty, které protínají jeho zorné pole. Aby efekt fungoval správně, musí materiál musí používat speciální typ shaderu [6]. Dostupné shadery jsou `Projector/Additive` a `Projector/ Multiply`, který byl pro tuto aplikaci nakonec zvolen. Oba tyto shadery se snaží kombinovat původní texturu objektu s texturou promítanou projektorem, liší se ale způsobem, jakým textury kombinují. Projektoru je nutné předat dvě textury, jedna obsahuje obraz, který je promítán do scény, druhá masku, která upravuje způsob projekce, zejména pak slábnutí světla projektoru se vzdáleností. Což je ale věc, kterou v tomto případě nechceme, pokud by ale tato maska nebyla projektoru předána vůbec, promítal by obraz nejenom před sebe, ale také za sebe. Vhodná maska je tedy pouze 1D textura – bílý pruh s jedním černým pixelem na levé straně, který zabrání projekci za projektor.

Pro inicializaci vlastností projektoru byl vytvořen script `ProjectorController`, který při startu nastaví FOV projektoru a poměr stran videa. Poslední věcí, kterou bylo nutné nastavit, je vlastnost `Ignore Layers`, tedy vrstvy obsahující objekty, na které se obraz z projektoru promítat nemá – zejména tedy WayPointy a další navigační prvky.

## 4.5 Pohled do scény

Pohled do scény zprostředkovává Unity kamera. Hlavní kamerou je možné pohybovat pomocí šipek na klávesnici a zoomovat pomocí kolečka myši nebo tlačítek PageUp a Page-



Obrázek 4.7: Projekce pomocí Unity projektoru přímo na texturu terénu: **vlevo:** Pokud by textura neobsahovala bílý rámeček o šířce 1px, krajní pixely se roztáhnou donekonečna. **vpravo:** Textura s rámečkem už funguje podle očekávání a je zkombinována s texturou terénu.

Down. Do budoucna je plánované také ovládání pomocí head-trackeru v brýlích pro virtuální realitu.

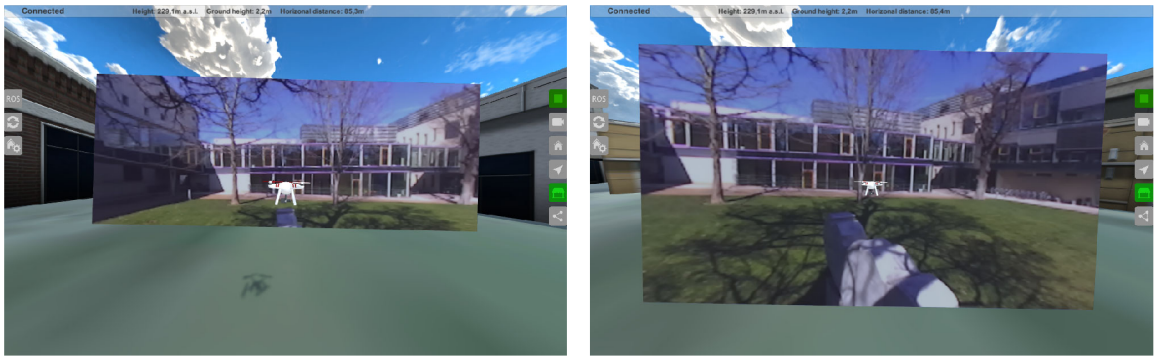
### Viditelnost objektů

Scéna obsahuje velké množství objektů. Kromě dronu samotného, virtuálního promítacího plátna s videem a navigačních prvků jsou to především terén a budovy. Terén a budovy jsou však pro pilota mnohem méně důležité než ostatní prvky, jsou ale poměrně velké a mohly by tyto důležitější objekty zakrýt. V Unity přirozeně funguje to, že objekty bližší překrývají ty vzdálenější, zde je však nutné nějaké objekty upřednostnit, aby byly vždy viditelné, i když se zrovna nacházejí za budovou nebo částí terénu. Příkladem může být právě již zmíněné virtuální plátno s videem, které se nachází poměrně daleko od dronu a bližší budovy by ho tak překryly. Tím by ale pilot přišel o svůj nejdůležitější zdroj informací, proto je nutné, aby video bylo vždy viditelné (obrázek 4.8).

Toho je možné v Unity dosáhnout několika způsoby. Jedním z nich je tvorba vlastních shaderů. Vzhledem ke značné složitosti tohoto řešení bylo použito řešení využívající kombinaci obrazů dvojice kamer s téměř totožnými parametry umístěnými na stejném místě, každá z nich však zabírá jiné objekty. Toho lze docílit úpravou dvou vlastností těchto kamer. První vlastností je `Culling Mask`, která určuje viditelnost jednotlivých vrstev [15]. Objekty tedy musí být rozděleny do těchto vrstev a první kamera tak vidí pouze méně důležité objekty – terén, budovy a skybox (nebe a mraky). Druhá kamera naopak vidí důležitější objekty jako je dron, video a navigační prvky. Výsledný obraz je následně získán jako kombinace obrazů z těchto dvou kamer. K tomu slouží druhá vlastnost Clear Flags, která je u první kamery nastavena na hodnotu `Skybox` a u druhé na `Depth only`.

### Pohyb kamery a zoom

Objekt `CameraBox` je v objektové hierarchii součástí `DroneObject` – pohybuje se tedy současně s dronem, aby však bylo možné pohled otočit a podívat se na dron např. shora, obsahuje `CameraBox` komponentu `CameraController`, ta pomocí lokálních transformací kameru otáčí na základě vstupu z klávesnice (šipky). Pohyb kamery je ale omezen na 90° shora, 30° zdola a 45° ze stran. Díky tomu pilot vždy vidí dron přibližně zezadu nebo shora a



Obrázek 4.8: Viditelnost objektů – ukázáno na „virtuálním plátnu“ s videem: **vlevo: Spodní třetina videa je schována pod terénem, pilot tak část obrazu nevidí. Vpravo:** Video upřednostněno před ostatními objekty pomocí použití dvou kamer a kombinace jejich obrazů.

vždy tedy snadno určí, kam směřuje příď letounu, navíc má vždy v zorném poli alespoň část videa z kamery.

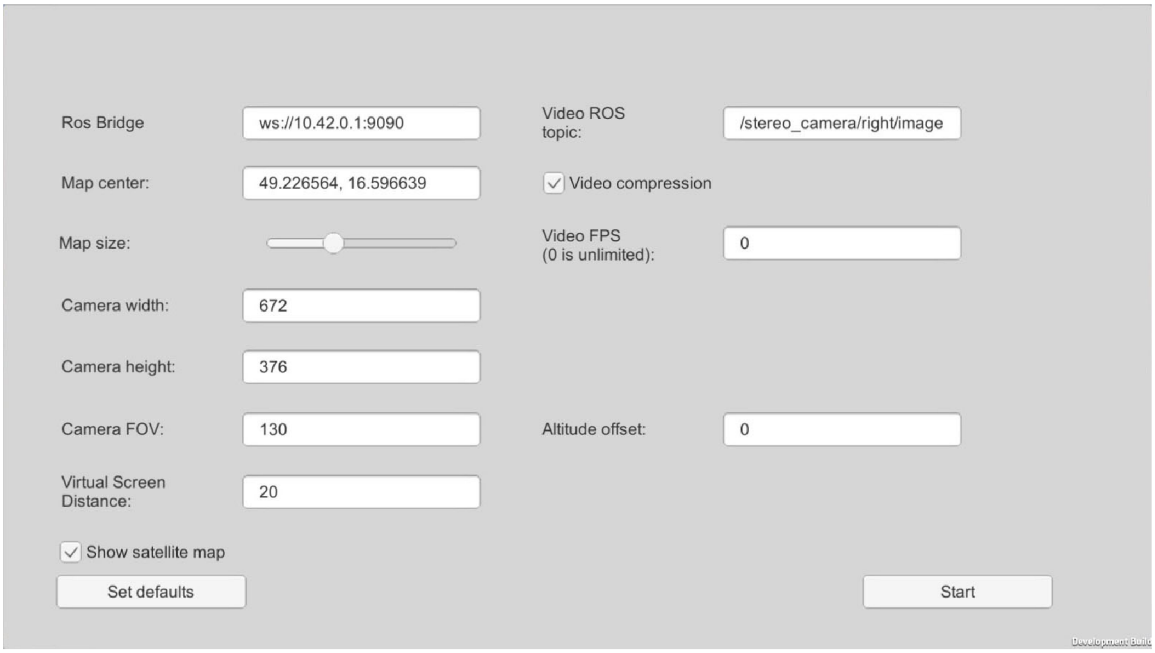
Objekt `CameraBox` obsahuje také kameru `ArrowCamera` s ortografickou projekcí, o které bude řeč níže a objekt `MainCameraPair`, který sdružuje obě kamery, které zajišťují správnou viditelnost objektů, tak jak bylo popsáno v předchozí podkapitole. Objekt `MainCameraPair` má také přiřazenou komponentu `CameraZoom`, která má právě zoom na starosti. Zoomování je v Unity možné docílit dvěma způsoby. Prvním je změna FOV (šířky zorného pole) kamery, druhou je pak fyzický posun kamery blíž nebo naopak dál od sledovaného objektu a `CameraZoom` provádí právě toto a hýbe s objektem `MainCameraPair`, který obsahuje obě kamery. Díky hierarchickému skládání transformací stačí s kamerou pohybovat pouze v ose Z a nezáleží na tom, jak je kamera zrovna otočená a jestli uživatel zrovna dron sleduje zezadu a nebo shora.

## 4.6 Uživatelské rozhraní

Aplikace se skládá ze svou scén. První scéna, kterou uživatel uvidí po spuštění aplikace je scéna konfigurační, kde nastaví důležité parametry jako je URL ROSbridge serveru, jméno ROS topicu s videem, vlastnosti kamery nebo parametry mapy. Po uložení nastavení přejde do hlavní scény.

### Konfigurační scéna

Konfigurační scéna neobsahuje žádné 3D objekty a skládá se tak pouze z prvků GUI. Ty jsou organizovány v prvku `Canvas`, který v Unity slouží pro vykreslování 2D prvků a prvků uživatelského rozhraní. Jako podklad je použit prvek `Panel` s šedou barvou pozadí, který překrývají formulářové prvky. Pro textový vstup (např. ROS bridge URL) slouží prvek typu `InputField`. K nastavení velikosti mapy slouží slider, ten umožňuje nastavit mapu velikosti 1 - 10 dlaždic do všech světových stran, což odpovídá mapě až 8400x8400m (viz kapitola ). Dalším použitým UI prvkem je zaškrtávací políčko `toggle`, které je zde použito např. k povolení komprese videa. Pro přepnutí do hlavní scény pak slouží tlačítko „Start“ (typ `Button`).



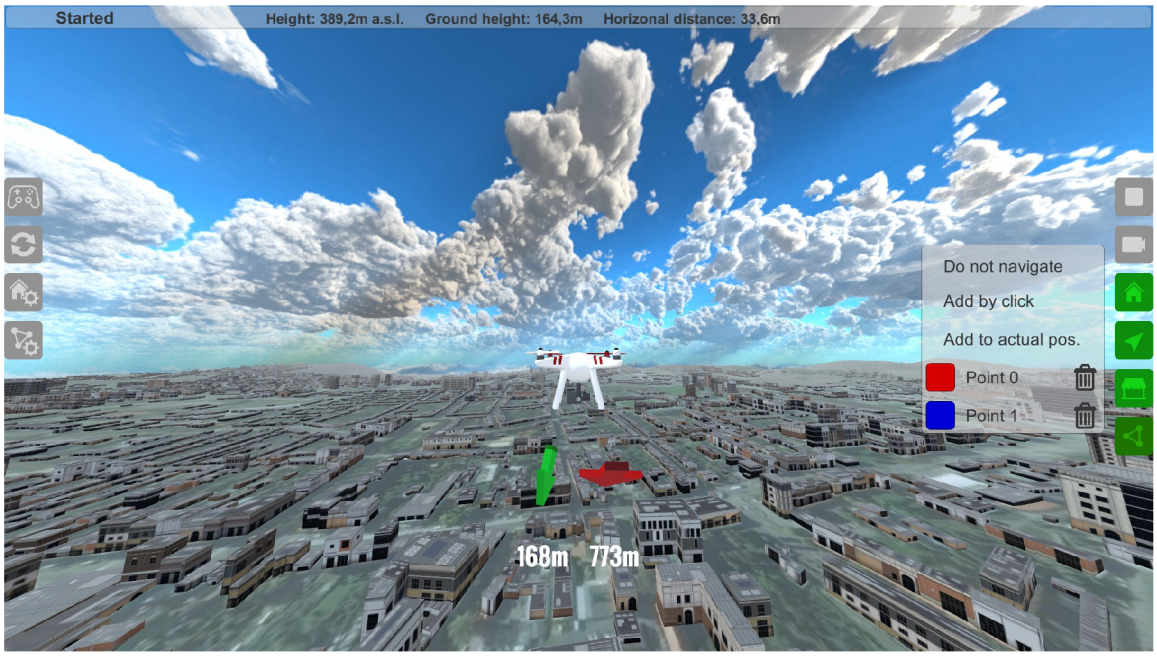

Obrázek 4.9: Konfigurační scéna obsahuje mnoho formulářových prvků. Ty jsou pře změně automaticky uloženy a zapamatovány pro další spuštění aplikace.

Zpracování událostí UI zajišťuje script `StartUpManager`. Ten po startu inicializuje všechna formulářová pole na výchozí hodnoty nebo na hodnoty, se kterými byla aplikace naposledy spuštěna. K tomuto účelu slouží v Unity třída `PlayerPrefs`, která funguje jako asociativní paměť a slouží právě k uchování uživatelských nastavení mezi jednotlivými spuštěními aplikace. Každému formulářovému prvku je přiřazena jedna metoda scriptu `StartUpManager`, která je volána při změně hodnoty a na základě této nové hodnoty je upraven příslušný záznam v `PlayerPrefs`.

### Hlavní scéna

Hlavní scéna má za úkol primárně zobrazovat dron a video, proto bylo nutné rozmístit ovládací prvky tak, aby zabíraly co nejméně místa. Proto bylo zvoleno zobrazení pomocí ikon po stranách. Byl použit balíček ikon z Unity Asset Store, ten ale všechny potřebné ikony neobsahoval, proto musely být některé upraveny a dokresleny. Aby ikony byly vždy na správném místě a správně velké při použití libovolné velikosti okna aplikace, bylo nutné přidat script `CanvasScaler`, který umožní přepočítávat velikost UI prvků z referenčního rozlišení na rozlišení aktuálně používané. Kromě ikon po stranách GUI obsahuje také úzký bílý panel nahoře, který slouží pro zobrazení základních informací o připojení k dronu a letových údajích.

Funkcionalitu všech klikatelných prvků zajišťuje script `GuiController`, ten obsahuje spoustu jednoduchých metod, které obsluhují události kliknutí na jednotlivé ikony. Ikony v pravém sloupci slouží pro zapnutí a vypnutí jednotlivých funkcí. To, zda je funkce zapnuta je symbolizováno zeleným podbarvením ikony. Script samotný aktivaci a deaktivaci funkcí neprovádí, ale volá vždy příslušné metody kontrolérů těchto funkcí. Nejsložitějším prvkem GUI je kontextové menu navigace. To je realizovaného pomocí částečně průhledného panelu a umožňuje WayPointy přidávat, odebírat, zobrazit jejich seznam a po kliknutí na daný




Obrázek 4.10: Hlavní scéna obsahuje horní panel se základními informacemi o připojení k dronu a letových údajích a klikatelné ikony po stranách. Ikony na pravé straně slouží k zapínání jednotlivých funkcí. Ikony vlevo k nastavení nejdůležitějších funkcí aplikace.

bod k němu navigovat. Přidání nového bodu se provádí buď na aktuální pozici dronu a nebo je aktivován `MapCliCkController` v režimu přidání nového WayPointu, který čeká na kliknutí uživatele do terénu (viz kapitola 4.7). Pokud k takovému kliknutí dojde, je opět volána metoda GUI `OnMapClickInWaypointMode()`, která volá `NavigationController` a informuje uživatele o úspěšném přidání nového bodu.

## 4.7 WayPointy a navigační šipky

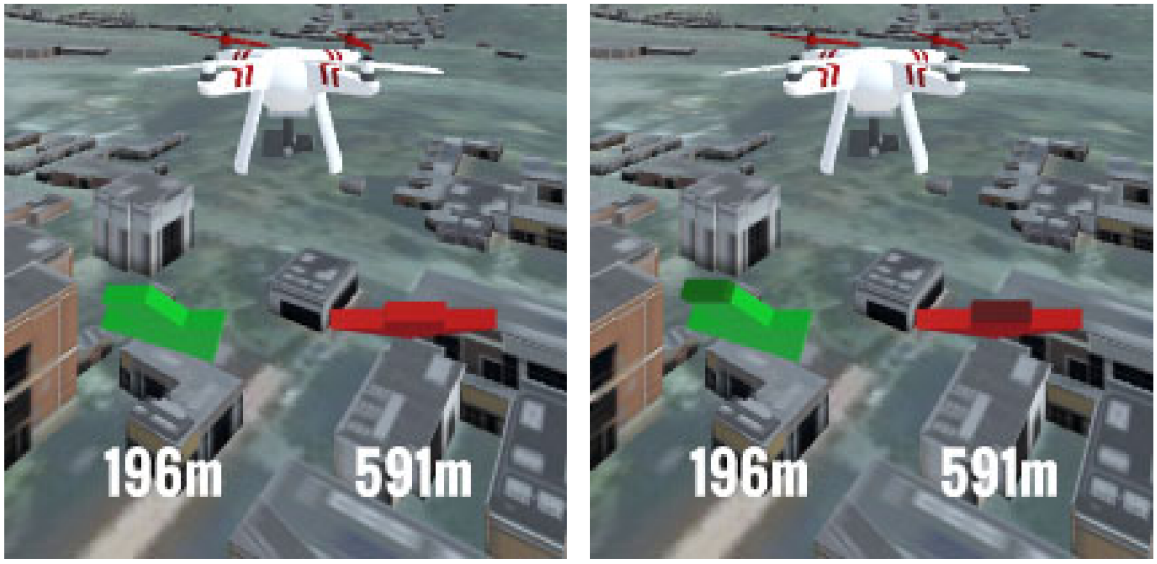
WayPointy – tedy důležité body, ke kterým má být uživatel navigován během letu, mohou být dvojího typu – bod na zemi a bod ve vzduchu. Bod na zemi je možné přidat na pozici, kde se dron aktuálně nachází. Zobrazován je jako barevná, částečně průhledná koule umístěná v prostoru. Bod na zem se přidává kliknutím a je zobrazen jako barevný kruh na zemi. HomePoint (místo startu) je navíc doplněn červenou kružnicí a písmenem „H“. O správu WayPointů vytváření nových se stará třída `NavigationController`, ta udržuje seznam těchto bodů, umožňuje navigaci k nim a při vytvoření nového bodu objekt přidá do scény.

Nalezení pozice, kam uživatel klikl má na starost komponenta `MapClickController`. Tato třída má za úkol zachytit kliknutí do mapy, nalézt jeho pozici a předat tyto 3D souřadnice dalším třídám, podle toho, v jakém režimu se zrovna nachází. Tato třída slouží tedy nejenom pro vytváření WayPointů, ale také pro definování hranic oblastí. Nalezení souřadnic kliknutí je prováděno pomocí tzv. RayCastu, kdy je z 2D bodu na obrazovce, kam uživatel klikl, vystřelen paprsek do scény a získána 3D souřadnice místa, kde tento paprsek protnul nějaký objekt. Tyto objekty je dále ještě možné filtrovat pomocí LayerMask [6]. Tato maska určuje, kterými objekty může paprsek projít a kterými nikoliv. Přiřazení

terénu do příslušné Unity vrstvy ukazuje obrázek 4.1. Díky použití této masky, je možné získat souřadnice, které leží vždy na povrchu země (průsečík s terénem), i když se v cestě tohoto paprsku nacházejí ještě další objekty (budovy, ukazatele WayPointů, prvky GUI...).

Navigační šipky byly vytvořeny v modelovacím programu Fusion 360 a do Unity následně importovány. Otáčení šipek tak, aby vždy směřovaly k WayPointu je v Unity možné snadno realizovat pomocí transformace `LookAt()`, která objekt automaticky otočí směrem k cílovému bodu. Pro šipky se také ukázalo jako výhodnější vypnout perspektivní zobrazení. Šipky tak ukazují směr k blízkým objektům přesněji a lépe se dá také nastavit jejich pozice a velikost na obrazovce uživatele. Aby bylo možné perspektivní zobrazení vypnout, je nutné přidat další kameru, která vidí pouze tyto šipky a používá právě ortografickou projekci. Obraz ze zkombinuje s dalšími kamerami stejným způsobem, jako bylo popsáno v kapitole 4.5.

Během testování se navíc ukázalo, že šipky při určitém otočení je velmi špatně vidět, kam šipka ukazuje, protože není přímo vidět její špička. Tento problém se podařilo vyřešit úpravou šipky tak, aby její zadní konec byl vždy o něco tmavší. Díky tomu uživatel mnohem rychleji pozná, jestli šipka ukazuje směrem k němu nebo od něj (viz obrázek 4.11).




Obrázek 4.11: Navigační šipky mohou výrazně pomoci pilotovi se správně zorientovat, zelená šipka směřuje k místu startu, šipka jiné barvy (zde červená) ukazuje k WayPointu, kam dron právě letí. Na obrázku vlevo je velice špatně poznat, kam vlastně červená šipka ukazuje (od uživatele nebo k němu?). Pokud ale má šipka jeden konec tmavší, uživatel dokáže mnohem rychleji určit, kam šipka míří.

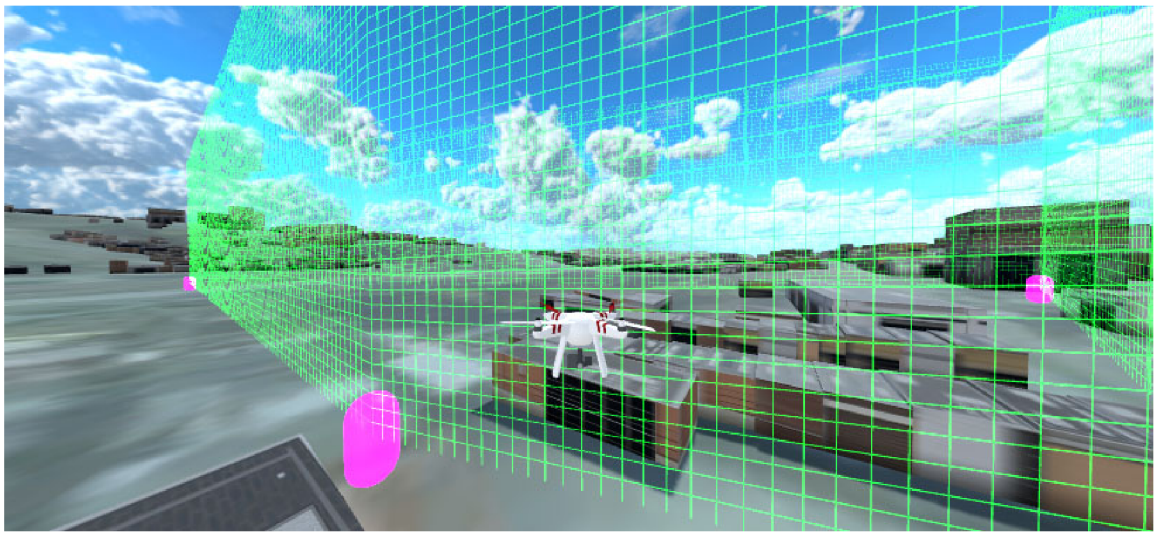
## 4.8 Hranice oblastí

O správu bezpečných zón a vytváření jejich hranic se stará třída `ZoneController`. Zachycení kliknutí do mapy a získání souřadnic zajišťuje stejně jako v případě vytváření WayPointů třída `MapClickController`. Pokud uživatel klikne na ikonku editace oblastí, `MapClickController` je přepnut do režimu editace zón a při kliknutí do mapy volá metodu OnClickInZoneMode() třídy `ZoneController` a předá jí polohu místa, kam uživatel kliknul. Nový bod je uložen do seznamu bodů a na místě kliknutí je vytvořen ukazatel hraničního

bodu. Jako hraniční bod byl zvolen základní Unity objekt `Capsule` – tedy válec se zaoblenými hranami. Aby byl hraniční bod dobře vidět ale zároveň se nepletl s WayPointy, byl pro něj zvolen materiál výrazné růžové barvy. Pokud je aktivní režim editace oblastí, tyto body jsou viditelné a je možné je posouvat po scéně. Pokud je režim editace vypnutý, body jsou skryty a zůstávají viditelné pouze hranice oblastí.

Při přidání nového bodu polygonu je zároveň nutné aktualizovat také hranice oblastí. K tomu slouží metoda `DrawZone()`, která hranice oblasti při zavolání překreslí. Ta postupně volá metodu `DrawZoneWall()` pro všechny stěny polygonu.

Samotné stěny oblastí jsou vytvořeny pomocí vodorovných a svislých přímek širokých 5cm a vzdálených 1m od sebe. Z těchto přímek tedy vznikne mříž, díky níž pilot vidí hranici, přitom je ale přes ni dostatečně dobře vidět, aby stále viděl ještě objekty za ní. Aby byly ale takto tenké přímky dobře vidět, musely mít výraznou barvu a navíc sami světlo vyzařovat a nespoléhat tak pouze na nasvícení scény. Toho je možné docílit použitím `Standard` shaderu a nastavením jeho vlastnosti `Emmission`. Pomocí ní je možné řídit barvu a intenzitu světla vyzařovaného z povrchu objektu. Díky tomu je tento objekt viditelný i v naprosto tmavé scéně.



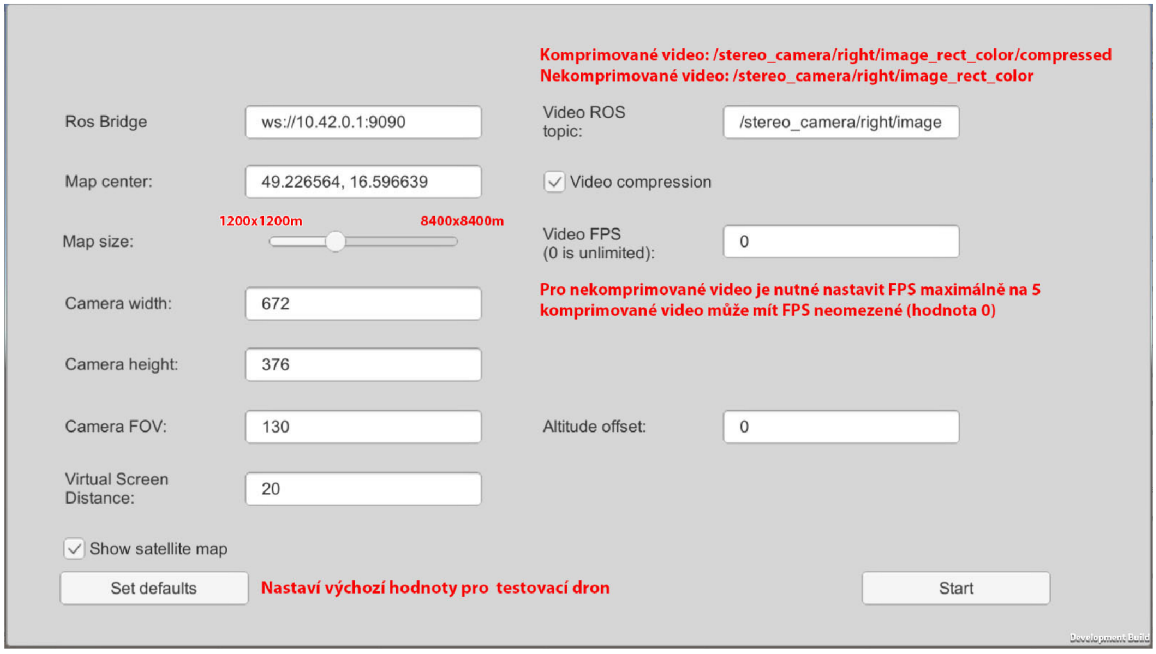
Obrázek 4.12: Hranice oblastí jsou vytvořeny pomocí mříží z materiálu, který vyzařuje světlo a je tak díky tomu dobře viditelný i v ne úplně dobře osvětlené scéně. Hraniční body mají velmi výraznou růžovou barvu, jsou ale viditelné pouze v editačním režimu, kdy je možné posouvat je myší po scéně.

## 4.9 Manuál k použití

Tato krátká sekce má za cíl v ucelené podobě shrnout znalosti z předchozích kapitol a doplnit je o další informace nutné k úspěšnému připojení aplikace k dronu a jejímu používání.

Po spuštění aplikace se otevře malé okno, kde je možné nastavit základní parametry aplikace. Především pak rozlišení a množství detailů. V sekci `Input` se nastavuje ovládání simulátoru a pohybu kamery. Nejdůležitější je správné přiřazení os `Roll`, `Pitch`, `Yaw` a `Throttle` osám použitého ovladače (vhodný gamepad je zobrazen na obrázku 4.5). Dále je možné nastavit také ovládání os `Horizontal` a `Vertical`, které slouží pro manipulaci s kamerou. Ve výchozím nastavení je kamera ovládána šipkami na klávesnici. Ovládání osy

`CameraZoom` je ve výchozím stavu nastaveno na kolečko myši, je však možné přidat např. i nějakou další osu joysticku. Alternativně je pro ovládání zoomu možné použít klávesy „PageUp“ a „PageDown“. Po kliknutí na tlačítko „Play“ je spuštěna konfigurační obrazovka samotné aplikace.




Obrázek 4.13: Konfigurační obrazovka aplikace s poznámkami k některým nastavením.

Konfigurační obrazovka je zobrazena a popsána na obrázku 4.13). Aplikace je ve výchozím stavu nastavena tak, aby bylo možné připojit se beze změny konfigurace k testovacímu dronu. Při změně nastavení je možné s vrátit k těmto výchozím hodnotám pomocí tlačítka „Set defaults“ Nejdůležitějším nastavením je URL ROSbridge serveru, ve které je obvykle nutné měnit pouze IP adresu. ROSbridge server je možné spustit přímo na testovacím dronu. Nastavovaná IP adresa tedy náleží WiFi hotspotu na dronu.

Druhým velmi důležitým nastavením je místo startu. Vzhledem k tomu, že jde pouze o testovací aplikaci, aplikace si místo neurčí sama, je nutné souřadnice zadat. Aplikace potřebuje v okamžiku startu na novém místě být připojena k internetu, aby si mohla stáhnout mapová data. Teprve poté je možné se připojit k WiFi na dronu. Při dalším spuštění na stejném místě už ale data načítá z vlastního úložiště. Parametr „Map Size“ určuje velikost mapy, která má být načtena. Čím výkonnější zařízení, tím větší území je možné zobrazit. Pro průměrně výkonný notebook je vhodné nechat posuvník přibližně ve stejné poloze jako je na obrázku.

Další parametry nastavují vlastnosti kamery – rozlišení a FOV (field-of-view). Je důležité, aby nastavené rozlišení bylo stejné jako rozlišení v publikovaném topicu. Samotný video topic je možné nastavit v dalším poli. Na obrázek 4.13 byla doplněna poznámka se jmény topiců s komprimovaným i nekomprimovaným videem.

Testovací dron někdy špatně určuje nadmořskou výšku a v testovacích záznamech se výška od skutečné liší dokonce o 73m. Proto bylo do aplikace doplněno pole „Altitude offset“. Tento záporný offset je následně k výšce přičten a dron se tak zobrazuje na správném místě. Offset je možné nastavit i za provozu, když dron leží na zemi, stisknutím klávesy „O“.

### Hlavní obrazovka aplikace


Obrázek 4.14: Hlavní obrazovka aplikace s popsanými ovládacími prvky. Dále jsou vyznačeny body, ke kterým ukazují navigační šipky.

Hlavní ovládací prvky aplikace jsou popsány na obrázku 4.13). Ikony vlevo slouží k nastavení funkcí aplikace. První přepíná zdroj dat. Kliknutím se postupně mění 3 režimy, současně se také mění zobrazená ikona. Zdroje dat jsou tyto: náhodný pohyb (ikonka RAND), data z dronu (ikonka ROS) a simulátor (ikonka s gamepadem). Při přepnutí na režim „ROS“ se aplikace automaticky začně připojovat k ROSbridge serveru. Stav připojení je zobrazen v horní liště vlevo. Pokud je připojení neúspěšné nebo vypadne, je možné ho obnovit kliknutím na druhou ikonku. Třetí ikonka změní pozici HomePointu a nastaví ho na na zem pod aktuální pozici dronu (HomePoint je vždy na zemi). Poslední ikonkou vlevo se zapíná a vypíná editace oblastí. Pokud je režim editace zapnutý (ikona svítí zeleně), je možné kliknutím na terén přidávat hraniční body oblasti. Hraniční body je také možné posouvat (drag & drop). Režim editace se vypne opětovným kliknutím na ikonu.

Ikony na pravá straně slouží k vypínání a zapínání jednotlivých funkcí aplikace. Zelená ikona znamená, že je daná funkce zapnutá. Je tak možné zapínat virtuální obrazovku, video projektor, navigaci k místu startu (zelená šipka), zobrazení budov a zobrazení hranic oblastí (pokud jsou nastaveny).

Jedinou výjimkou je zde ikona pro navigaci k WaypPointům, ta neslouží pouze k zapínání a vypínání druhé navigační šipky, ale zobrazí kontextové menu, pomocí kterého je možné nové body také přidávat a to buď na aktuální pozici dronu ve vzduchu a nebo na zem kliknutím. Každý bod má navíc jinou barvu a jeho barvě odpovídá také navigační šipka, aby se tyto body nepletly.

Některé funkce je možné vypínat a zapínat také pomocí určitých kláves. Jendou z funkcí, pro které není v GUI ikona je přepnutí mapy za provozu. K tomu slouží klávesa „V“,

která přepíná mezi obyčejnou a satelitní mapou. Pomocí klávesnice lze přepínat také mezi simulátorem (klávesa „M“) a vstupem z dronu (klávesa „R“). Dále je možné klávesou „J“ přepnout na pohled shora a klávesou „K“ zpět na pohled zezadu. Pohled je dále možné upravovat klasicky šipkami. Klávesa „B“ slouží k rychlému zapnutí a vypnutí virtuální obrazovky.

# Kapitola 5

# Testování a možnost dalších rozšíření

V této kapitole jsou definovány základní případy užití, které popisují úkoly, na kterých byla aplikace zkoušena a průběhy testů jak v simulátoru, tak se skutečným dronem. Cílem bylo vyzkoušet nejenom tento koncept řízení jako takový, ale také navržené navigační prvky. Dále popisuje průběh těchto testů, jejich vyhodnocení a navrhuje úpravy aplikace, které by mohly pomoci nalezené problémy eliminovat a ukazuje možné směry dalšího vývoje navrženého konceptu.

## 5.1 Případy užití

### Monitorování oblasti, nad kterou dron nesmí vletět

Při využití dronu např. ve službách policie je často nutné monitorovat nějakou oblast (silnice, demonstrace...). Policie musí ale také dodržovat předpisy a ne vždy má sjednanou výjimku. Z toho důvodu ani jejich drony nesmí letět např. v blízkosti dálnice (viz kapitola 2.8) nebo nad náměstím plným lidí. Pokud pilot chce pomocí dronu danou oblast sledovat, ale zároveň se udržet mimo její ochranné pásmo, musí neustále kontrolovat, nad čím se dron právě nachází. Zde by mohly nalézt uplatnění navržené virtuální zdi na hranicích oblastí, které by pilotovi měly přesně ukázat, kam letět ještě může a kam už by neměl. Toto by bylo dobré vyzkoušet jak v malé letové výšce (do 40m), tak také ve výšce větší, (alespoň 80m). V každé výšce totiž pilot přirozeně vnímá jinak to, nad čím se letoun nachází.

### Průzkum vzdáleného objektu

Tento případ popisuje další celkem běžný úkol, kdy je nutné prozkoumat vzdálenější objekt (např. dům nebo zaparkované auto), které je ale příliš daleko na to, aby pilot mohl řídit přímým pohledem na dron. Zde je cílem vyzkoušet, jestli aplikace opravdu může pomoci zlepšit pilotovu prostorovou orientaci během letu na toto místo, obletu zkoumaného objektu a návratu zpět. Při tomto testu se předpokládá, že pilot dopředu zná pozici tohoto objektu a může tak na daném místě vytvořit WayPoint a nechat se k němu navigovat.

### Let v malé výšce mezi překážkami

Běžná situace, kdy není možné použít autonomní let, ale pilot musí ručně prokličkovat mezi spoustou překážek (např. stromy) a přitom ani v prostředí bez výrazných orientačních bodů neztratit prostorovou orientaci a přehled o směru, kam letí. V tomto režimu není možné využít mapová data, ale je nutné letět pouze pomocí videa a pečlivě sledovat překážky. Zde by ale mohly pomoci navigační šipky, které pilota neustále informují o směru k WayPointu, ke kterému letí i k místu startu. Díky tomu by pilot mohl zvládnout tento úkol rychleji.

## 5.2 Testování během vývoje

V první fázi vývoje byl využíván pouze náhodný pohyb dronu po scéně. To samozřejmě velmi rychle přestalo stačit a proto byl do aplikace přidán simulátor, který umožnil ruční řízení a pohyb po scéně. Díky tomu bylo možné začít testovat navigační prvky aplikace. Simulátor však neobsahuje video z reálného světa, které by bylo možné zobrazit na virtuální obrazovku. Reálný dron ale nebyl k dispozici a stále na něm probíhaly práce. Vzhledem k tomu, že dron ještě nebyl schopný bezpečného letu, byl alespoň ručně (nošením dronu) vytvořen krátký záznam ROSbag, který obsahoval jak video, tak data o poloze a náklonu letounu. Tento záznam tedy obsahoval několik základních ROS topiců nutných k provozu aplikace. Záznam byl následně doplněn ROSbridge serverem a virtualizován pomocí aplikace Docker, aby jej bylo možné snadno přehrát na běžném počítači.

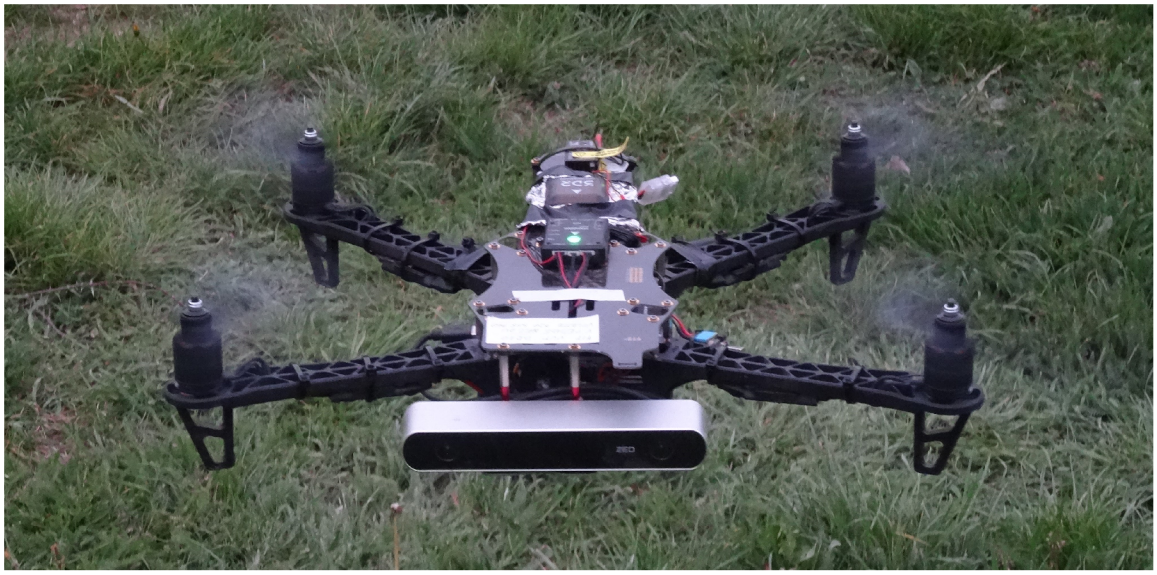
Tento záznam byl i přes všechna svá omezení velmi užitečný při zkoušení propojení dronu s aplikací. Obsahoval totiž plnohodnotný ROSbridge server, ke kterému se bylo možné připojit. Ukázalo se však, že virtualizace pomocí Dockeru je poměrně náročná na HW počítače a bylo tedy vhodnější ji nechat běžet na jiném počítači než samotnou aplikaci. Vzhledem k tomu, že veškerá komunikace zde probíhá po síti přes protokol WebSocket, nebylo umístění na jiný fyzický počítač žádným problémem a v aplikaci při startu stačilo pouze zadat URL s jinou IP adresou. Aplikace se připojila, požádala o odběr příslušných topiců a data bylo možné začít zobrazovat v aplikaci. V této fázi bylo možné vyladit problémy jako přehozené osy IMU, kdy se dron nakláněl dopředu a dozadu namísto do stran a podobně. Záznam však neobsahoval celý seznam ROS topiců, ale jen některé vybrané. To se ukázalo jako problém zejména u videa, kde byl k dispozici pouze topic s nekomprimovanými snímky a nebylo zcela jasné, jestli je reálný dron schopný poskytovat také video komprimované. V této fázi bylo tedy rozhodnuto o dočasném použití videa nekomprimovaného a vyřešení tohoto problému až ve chvíli, kdy bude reálný dron dostupný a schopný provozu. Aby ale vše fungovalo i bez komprese, bylo nutné omezit FPS videa. V opačném případě docházelo k výpadkům ROSbridge serveru a bylo nutné restartovat celou virtualizaci v Dockeru.

Dalším problémem byla poloha dronu a zejména pak nadmořská výška uložená v dostupných záznamech, ta totiž byla průměrně o 75m vyšší než nadmořská výška terénu v daném místě, což vzhledem ke skutečnosti, že při nahrávání těchto záznamů byl dron pouze ručně nošen ve výšce 1-2m nad zemí, nemohlo být správné. Proto bylo do konfigurační scény aplikace doplněno formulářové pole pro nastavení offsetu výšky. Tento záporný offset byl následně k výšce přičten a dron se tak zobrazoval na správném místě – tedy přibližně 1-2m nad zemí. Tento problém se později projevoval také během testů s dronem reálným.

## 5.3 Testování s reálným dronem

Když byl dostupný testovací dron, bylo možné začít zkoušet připojení přímo k němu namísto Docker virtualizace. Dron je vybaven superpočítačem Nvidia Jetson TX2 s běžícím plnohodnotným OS Linux Ubuntu a dvojicí WiFi antén (schéma dronu zobrazuje obrázek 3.5). Pozemní stanici s aplikací tedy bylo nutné připojit na WiFi hotspot dronu a spustit na něm ROSbridge server a publikaci MAVros topiců a topiců stereoskopické kamery ZED.

Po připojení dronu se potvrdil předpoklad, že přenos nekomprimovaného videa bude problém, protože latence mezi pohybem dronu a zobrazením tohoto pohybu v aplikaci dosahovala někdy i 20s, co by bylo pro létání samozřejmě naprosto nepoužitelné. Ukázalo se však, že ZED kamera publikuje také topic s komprimovaným videem, který nebyl v ROSbag záznamu dostupný. Aplikaci tak bylo možné upravit, aby umožňovala volbu také tohoto topicu. Komprimované video vyžaduje přibližně 32x menší šířku pásma a latence díky němu klesla na přijatelnou úroveň.



Obrázek 5.1: Testovací dron s řídící jednotkou Pixhawk, stereoskopickou kamerou ZED a superpočítačem Nvidia Jetson vyfocený za letu těsně nad zemí.

Dron byl sice funkční, ale ještě téměř nevyzkoušený, proto nebylo možné, aby aplikaci propojenou s ním zkoušelo více lidí. Z toho důvodu jsem musel veškeré testy udělat pouze já sám. Dron byl poměrně stabilní, ale v režimu `Position` se stále poměrně dost samovolně pohyboval přibližně 3-4m do stran, což bylo způsobeno tím, že při určování polohy spoléhá pouze na GPS a nebyl vybaven dalšími zpřesňujícími senzory (např. Optical Flow). Také se na rozdíl od profesionálních dronů značně kýval a při poryvu větru mu chvilku trvalo než se opět srovnal. Toto kývání se projevilo také na obrazu z kamery, protože koptéra nebyla vybavena gimbalem a veškerý pohyb dronu byl tedy ve videu vidět. Nicméně i tak se s dronem létalo poměrně příjemně a většinu testů bylo možné provést. Úspěšně se podařilo vyzkoušet také režim fail-safe, kdy se dron při výpadku signálu sám vrátil na místo startu a přistál. Při jednom z testů, ale došlo k závadě na řídící jednotce, což vedlo k nemožnosti pokračovat v testování a další testy už se po opravě dronu nestihly.

Vzhledem k připojení pozemní stanice pomocí WiFi nebylo možné s dronem létat příliš daleko, aby nedošlo ke ztrátě signálu. Koptéra se tedy po celou dobu pohybovala ve

vzdálenosti do 80m od místa startu. Dron měl také během testů z bezpečnostních důvodů omezenou maximální rychlost na 4m/s a bylo s ním létáno poměrně nízko (do 25m) a daleko od zastavěných oblastí. To jednak z legislativních důvodů, ale také proto, že dron ještě nebyl pořádně vyzkoušen a jeho chováním si tak nemohl být nikdo příliš jistý. Vzhledem k problému, který se nakonec ukázal, to bylo správné rozhodnutí a při pádu dronu do pole tak nedošlo k žádnému vážnějšímu poškození koptéry ani majetku nikoho dalšího.

Velkou nevýhodou této koptéry je její video kamera. Ta je sice stereoskopická a dron tak umí vytvářet point cloud, kamera je však zaostřena pouze na blízké objekty. Vzdálenější objekty jsou naopak značně rozostřené. Kamera také nedisponuje technologií HDR. Z toho důvodu je výsledný obraz neostrý, nekontrastní a nelétalo se podle něj příliš dobře (viz obrázek 5.2). Bylo by tedy vhodné dron doplnit ještě druhou kamerou, která těmito nedostatky trpět nebude. Problematikou kamer se zabývala kapitola 2.6.


Obrázek 5.2: Obraz z kamery na dronu nebyl za těchto světelných podmínek příliš dobrý a létání podle něj tak bylo obtížné. Při létání v této oblasti také nebylo možné použít satelitní mapu, protože obsahovala pouze jednobarevnou louku a nové cyklostezky na ní nebyly vůbec vidět, proto byla použita pouze obyčejná mapa.

## 5.4 Výsledky testů a navrhované úpravy

I přes všechny výše popsané problémy, bylo možné aplikaci s reálným dronem vyzkoušet a testování potvrdilo, že tento koncept funguje a je pomocí něj možné dron ovládat. S testovacím dronem však nebylo možné letět příliš daleko ani vysoko, proto hlavní přínos aplikace na prostorovou orientaci pilota bylo možné vyzkoušet pouze v omezené míře.

### Test 1: Monitorování oblasti, kam dron nesmí vletět

Testování hned prvního případu užití bylo nejobtížnější. Monitorování oblasti, kam dron nesmí vletět totiž předpokládá sledování oblasti z výšky, k tomu je ale nutné mít možnost otočit kameru směrem dolů. Testovací dron ale není vybaven gimbalem a vidí tak pouze před sebe. Oblast tak bylo možné sledovat jen z velmi malé výšky. Samotný testovací let měl tedy ověřit, zda přidání virtuálních zdí pomůže pilotovi udržet se na hranici povolené oblasti a přitom pozorovat dění za nami. Test by samozřejmě neměl smysl, pokud by se dron u hranice zastavil a dále se nepohyboval. Z toho důvodu musel být dron neustále v pohybu i když to v danou chvíli nemělo příliš význam.

Virtuální průhledné zdi byly velmi užitečné při létání uvnitř oblasti, pilot totiž velmi dobře pozná, že se k hranici blíží nebo že ji právě proletěl a ihned se tak vrátí zpět. Let na jejich hranici se ale ukázal jako překvapivě obtížný. Při letu v blízkosti těchto zdí (cca 5-8m) se totiž velice špatně odhaduje jejich vzdálenost. Vhodné by tedy bylo jejich skrytí a zobrazení pouze té nejbližší části hranice až ve chvíli, kdy se k ní dron opravdu přiblíží. Druhou možností by mohlo být postupné zprůhledňování zdí, pokud se od nich dron vzdaluje. Nejvýrazněji by tedy svítila ta část hranice, která je letounu zrovna nejblíž, vzdálené části by naopak byly úplně skryty. Pokud by tedy pilot uviděl před sebou výrazně zářící mříž, věděl by s naprostou jistotou, že je hranici už velmi blízko. Vzdálenost, ve které se hranice objeví, by bylo dobré ještě upravovat podle rychlosti, kterou se k ní dron zrovna blíží, aby měl pilot vždy dostatek času zareagovat, ale zároveň mu tato hranice zbytečně nepřekážela ve výhledu. Dalším problémem pak bylo prolínání virtuální zdi s virtuální obrazovkou, které působilo rušivě a nepříjemně. I tento problém by ale navrhované řešení mohlo alespoň částečně vyřešit, zobrazena by totiž vždy byla jen ta část hranice, která je v danou chvíli opravdu potřeba.

### Testy 2 a 3: Průzkum vzdáleného objektu a let mezi překážkami

Při testování průzkumu vzdáleného objektu byl na zem umístěn předmět ve vzdálenosti přibližně 60m od místa startu a na tomto místě byl vytvořen WayPoint. Mezi místem startu a tímto bodem se nacházela asfaltová cyklostezka a několik řad čerstvě zasázených mladých stromků, které vytvořily přirozené překážky. Tyto stromky bylo samozřejmě možné nadletět, pro účel testování bylo ale lepší kličkovat mezi nimi. Díky tomu bylo možné současně otestovat také třetí případ užití – let v malé výšce mezi překážkami.

Vzhledem k tomu, že se v okolí nacházely jen louky a nízké stromky, nebyly zde téměř žádné přirozené orientační body. Orientace pouze podle tohoto nepříliš kvalitního videa tak byla sama o sobě velice náročná. Během tohoto testu se ale právě velmi osvědčily navigační šipky, které plnily perfektně svoji roli, velmi usnadnily prostorovou orientaci a výrazně pomohly snížit kognitivní zátěž během řízení. Přirozené orientační body totiž vůbec nebylo nutné hledat, úplně stačilo sledovat blízké překážky, navigační šipku a postupně se snižující vzdálenost k cíli. Po chvíli se objevil na zemi ukazatel cílového bodu.

V této fázi ale nastal problém, červená koule označující cílový WayPoint totiž neustále cestovala po zemi o několik metrů všemi směry od cílového objektu, kterým byla relativně malá sportovní taška umístěná na zemi (viz obrázek 5.4). To bylo zřejmě způsobeno nepřesností GPS navigace. Při snaze obletět zkoumaný objekt a prozkoumat ho ze všech stran bylo neustálé pohybování tohoto bodu a zmatené otáčení šipky velice nepříjemné a matoucí. Průzkum tohoto předmětu to tak spíš komplikovalo.

Samotný oblet a „průzkum“ bylo tedy mnohem jednodušší udělat s vypnutou navigační šipkou pouze podle videa. Pohybující se barevný WayPoint, který v aplikaci skrýt nelze, ale


Obrázek 5.3: Virtuální zdi se osvědčily při létání uvnitř oblasti, pohyb po jejich okraji byl ale značně náročný, protože se špatně odhadovala jejich vzdálenost. Nepříjemné bylo také prolínání virtuální zdi s virtuální obrazovkou, které působilo poněkud rušivě. Na obrázku je také vidět, že mapový podklad na obraz z videa poměrně dobře navazuje (zakroužkováno).

při průzkumu také trochu překážel. V této chvíli by bylo tedy pravděpodobně lepší, kdyby se po příletu do těsné blízkosti cíle navigační prvky automatiky skryly a objevily se opět až v případě, že by se dron od cíle opět výrazněji vzdálil.

V tuto chvíli naopak bylo možné pro orientaci použít druhou šipku, která ukazuje k místu startu. Ta směr ukazovala spolehlivě, protože místo startu bylo dostatečně daleko. Tuto šipku tak bylo možné použít jako jakýsi kompas, díky kterému bylo jasně poznat, jak je dron zrovna otočen. Pro pilota je navíc ukazatel k místu startu o něco přirozenější než obyčejný kompas, který ukazuje k severu.

### Shrnutí

Celkově bylo řízení přes tuto aplikaci relativně příjemné a pozitivně lze hodnotit také to, že aplikace byla schopna částečně kompenzovat absenci gimbalu. Díky posouvání videa podle aktuálního náklonu dronu byly totiž objekty na videu zobrazeny stále na přibližně stejném místě ve scéně. Komfort používání aplikace ale samozřejmě dost snižovalo ovládání otáčení kamery pomocí šipek na klávesnici notebooku. Připojení VR brýlí s head-trackerem, které by pilotovi umožnily přirozeně se rozhlížet po scéně už se ale nestihlo.

Aplikace se při testování naopak příliš neosvědčila ve velmi malých výškách (do 3m), kde se velmi špatně odhadovala výška letu i vzdálenost od menších objektů. Tento problém však nastává i u pohledu první osoby (FPV). Přidání modelu dronu do scény však tento problém nesnížilo, ale naopak ještě mírně zvýšilo.


Obrázek 5.4: WayPoint (červená koule) byl umístěn u zkoumaného objektu (zakroužkováno). Vlivem nepřesnosti GPS se ale ukazatel WayPointu neustále pohyboval o několik metrů všemi směry od cílového objektu. To bylo velice nepříjemné, matoucí a znesnadňovalo to oblet a průzkum tohoto předmětu. Navigační šipky i ukazatel cílového bodu by tedy po dosažení určité minimální vzdálenosti od objektu bylo lepší automaticky skrýt a nechat už pilota řídit čistě podle videa.

Poměrně překvapivé bylo to, že obraz na virtuální obrazovce většinu času relativně dobře navazoval na virtuální scénu, což dokonce mírně překonalo očekávání a to především s ohledem na to, že testovací dron rozhodně nedisponoval nejpřesnějšími dostupnými senzory. Data z profesionálních dronu tak pravděpodobně budou ještě výrazně přesnější. Přesnost senzorů ale naprosto nedostačovala při zobrazení videa pomocí projektoru. Zejména pak při promítání videa na budovy, jejichž modely jsou navíc samy značně nepřesné. Potíže může způsobovat také samotný video projektor v Unity, ten je totiž primárně určený pro výrazně jednodušší projekce a s tímto využitím pravděpodobně nikdy nebylo počítáno. Projektor by ale s velkou pravděpodobností správně fungoval při projekci na terén z výšky, to však nebylo možné vyzkoušet, protože kameru na dronu nebylo možné sklopit směrem dolů.

## 5.5 Další vývoj aplikace

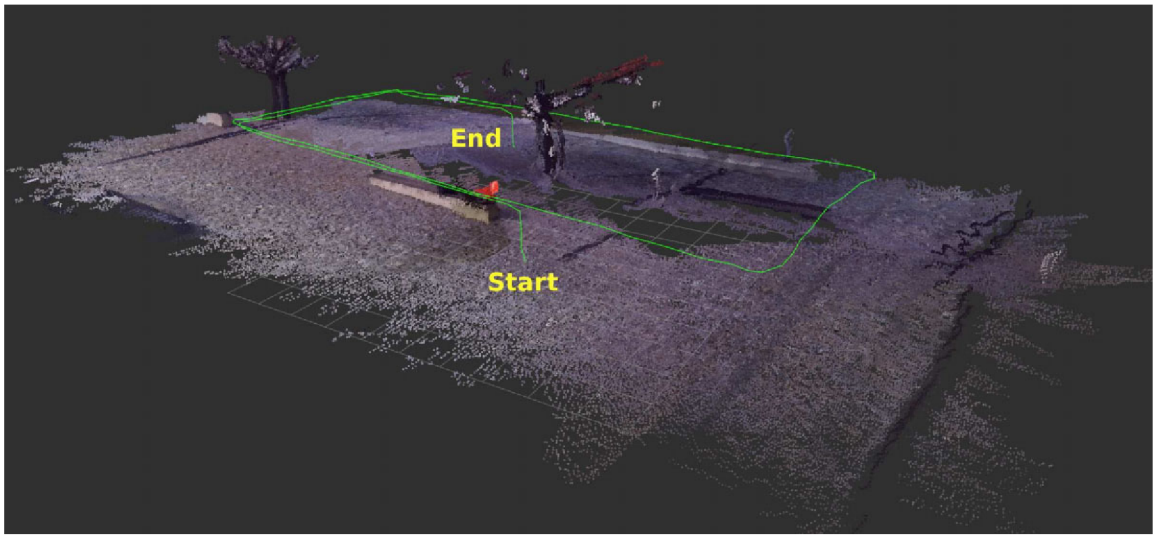
Další vývoj by se měl primárně zaměřit na úpravy navržené v předchozí podkapitole – tedy vyřešení problémů, které byly objeveny během testování a dále na věci, které byly sice navrženy již na začátku, ale jejich realizace se nestihla. Jde zejména dokončení hranic oblastí, které nebyly implementovány v celém navrhovaném rozsahu a při testování se navíc zjistilo, že je nutné je ještě vylepšit. Pak jde také o připojení VR brýlí s head-trackerem, který umožní přirozené rozhlížení po scéně. Další takovou věcí je implementace volné kamery a vyzkoušení jejích možností. O volné kameře pojednávala kapitola 3.4, která zároveň již naznačila některé problémy se kterými je nutné při jejím použití počítat. Další problémy i zcela nové možnosti využití se pravděpodobně objeví při testování.


Obrázek 5.5: Přesnost senzorů naprosto nedostačovala při zobrazení videa pomocí projektoru na budovy, jejichž modely jsou navíc samy značně nepřesné. Potíže může způsobovat také samotný video projektor v Unity, ten je totiž pravděpodobně určený pro výrazně jednodušší projekce. Projekci na terén, která by pravděpodobně fungovala mnohem lépe však nebylo možné vyzkoušet, protože kameru na dronu nelze sklopit směrem dolů.

Směr, kterým by se mohl vývoj také ubírat, je možnost připojení více dronů současně. To by se mohlo hodit v situacích, kdy misi plní současně více pilotů a musí nějak spolupracovat. Jako příklad může posloužit akce nějaké složky záchranného systému, kde drony mohou být použity např. k monitorování rozsáhlého požáru. Zde by tento vizualizační nástroj sloužil nejenom samotným pilotům, ale také jejich veliteli, který by mohl využívat právě volnou kameru a sledovat tak celou oblast a pohybující se drony v ní. Zároveň by měl mít možnost přepínat mezi svojí volnou kamerou a pohledy na jednotlivé drony. Velmi dobré by také bylo přímo do aplikace integrovat možnost záznamu dat z dronu a umožnit tak uživateli kdykoliv si tato data zpětně přehrát a podívat se tak na let z jiného úhlu nebo místa.

Testovací dron umí vytvářet point cloud – tedy mračno bodů, které se nacházejí na vnějších plochách objektů v okolí (viz obrázek 5.6). Tato data je možná opět získat čtením příslušného ROS topicu, aplikace to ale zatím nedělá a point cloud neumí vizualizovat. Tato data by ale mohla být velice užitečná. Mohou totiž výrazně zpřesnit a doplnit model světa vytvořený z veřejně dostupných dat a díky nim by pilot mohl mít mnohem lepší přehled o překážkách v okolí dronu, jako jsou stromy, auta a podobně.

Aplikaci by bylo také dobré doplnit o funkce, které ji více přiblíží běžným pozemním stanicím. V první řadě jde o zobrazení stavových informací (letový režim, stav baterie, síla signálu...), dále pak o možnost data nejenom vizualizovat, ale také do dronu posílat a např. umět nahrát vytvořenou misi do dronu, přepnout letový režim, určit kam má dron letět pouhým kliknutím a podobně. V neposlední řadě je také nutné umožnit ovládání základních funkcí aplikace pomocí vysílačky, aby pilot nemusel příliš často pouštět řízení a sahat na dotykový displej nebo dokonce na myš.




Obrázek 5.6: Point cloud vytvořený testovacím dronem a trajektorie letu při jeho vytváření [9].

# Kapitola 6

# Závěr

Cílem práce bylo navrhnout vizualizační interaktivní aplikaci, která bude integrovat on-line data z dronu a off-line data v jedné 3D scéně tak, aby usnadnila pilotovi řízení za pomoci video přenosu.

Podrobně jsem si tedy nastudoval problematiku dronů a jejich ovládání. Vyzkoušel jsem si let jak s reálnou koptérou, tak v simulátoru a během létání se mi potvrdila domněnka, že ztráta orientace a vnímání pozice během letu je pro pilota problém a pilotáž je tak značně náročná. Navrhl jsem tedy řešení tohoto problému pomocí pohledu třetí osoby a rozšířené virtuality, která navíc umožňuje vizualizaci dalších informací jako jsou bezpečné a nebo naopak zakázané oblasti, data ze senzorů a podobně. Navrhl jsem základní prostředí aplikace a dále se věnoval jeho realizaci. Vytvořil jsem testovací aplikaci pomocí veřejně dostupných map, propojil ji s dronem a tato data zobrazil ve scéně.

Testování se ukázalo jako značně náročný úkol. Sériově vyráběné drony bez úprav je totiž složité připojit k nestandardní aplikaci. Testy proto probíhaly na experimentální platformě. Tento dron byl sice lépe propojitelný, vzhledem k tomu, že ale šlo o prototyp a ne o vyladěný sériový výrobek, vyznačoval se značně menší spolehlivostí, přesností letu, výdrží baterie i dosahem signálu. S testovacím dronem tak nebylo možné letět příliš daleko ani vysoko, proto hlavní přínos aplikace na prostorovou orientaci pilota bylo možné vyzkoušet jen v omezené míře.

Během testování jsem zjistil, že koncept funguje a je pomocí něj možné dron ovládat. Z navigačních prvků se velice osvědčily 3D navigační šipky, které pilota spolehlivě dovedou na místo určení a pomáhají také v prostorové orientaci během letu. Jako částečně problematické se naopak ukázaly poloprůhledné hranice oblastí, které sice pomohou pilotovi tuto hranici nepřekročit, pokud se ale chce pohybovat v jejich v těsné blízkosti, velice špatně se odhaduje jejich vzdálenost. Navrhl jsem ale řešení, která by tento problém mohla odstranit. Aplikace se také příliš neosvědčila při letu ve velmi malých výškách, kde se poměrně špatně odhadovala výška letu i vzdálenost od menších objektů.

Testování ale ukázalo, že koncept funguje a závěrečná kapitola ukazuje směr dalšího možného vývoje a navrhuje konkrétní úpravy a vylepšení aplikace. Pro produkční verzi by však bylo dobré získat také kvalitnější mapová data a 3D modely budov. Další vývoj by se měl tedy zaměřit především na spolupráci s poskytovateli mapových dat a knihoven.

Ve vývoji tohoto konceptu určitě stojí za to pokračovat a v budoucnosti může přinést pilotům a operátorům dronů nové a velmi zajímavé možnosti.

# Literatura


[1] Dokumentace ROS wiki. [Online].
URL http://wiki.ros.org/

[2] INAV wiki. [Online].
URL https://github.com/iNavFlight/inav/wiki

[3] Letecký předpis L2. [Online].
URL https://lis.rlp.cz/predpisy/predpisy/dokumenty/L/L-2/index.htm

[4] Oficiální dokumentace Ardupilot. [Online].
URL http://ardupilot.org/copter/docs/

[5] Rosbridge v2.0 Protocol Specification. [Online].
URL https:
//github.com/biobotus/rosbridge_suite/blob/master/ROSBRIDGE_PROTOCOL.md

[6] Unity Documentation. [Online].
URL https://docs.unity3d.com/Manual/

[7] Web Drony-bezpečně.cz. [Online].
URL https://www.drony-bezpecne.cz

[8] Web MAVlink.io - dokumentace k protokolu MAVlink. [Online].
URL https://mavlink.io/en/

[9] Beran, V.; Plascencia, A. C.; Herout, A.; aj.: Technical report 2018. *Tools and Methods for Video and Image Processing to Improve Effectivity of Rescue and Security Services Operations (VRASSEO), Project no. VI20172020068, VUT Brno, Faculty of Information Technology*, December 2018.

[10] Bischoff, M.; Siemens: Dokumentace ROS# wiki. [Online].
URL https://github.com/siemens/ros-sharp/wiki

[11] Calhoun, G.; H. Draper, M.; F Abernathy, M.; aj.: Synthetic vision system for improving unmanned aerial vehicle operator situation awareness. *Proceedings of SPIE - The International Society for Optical Engineering*, ročník 5802, 05 2005, doi:10.1117/12.603421.

[12] Cho, K.; Cho, M.; Jeon, J.: Fly a Drone Safely: Evaluation of an Embodied Egocentric Drone Controller Interface. *Interacting with Computers*, ročník 29, č. 3, 09 2016: s. 345–354, ISSN 0953-5438, doi:10.1093/iwc/iww027, http://oup.prod.sis.lan/iwc/article-pdf/29/3/345/11149035/iww027.pdf.
URL https://doi.org/10.1093/iwc/iww027

[13] Erat, O.; Isop, W. A.; Kalkofen, D.; aj.: Drone-Augmented Human Vision: Exocentric Control for Drones Exploring Hidden Areas. *IEEE Transactions on Visualization and Computer Graphics*, ročník 24, č. 4, April 2018: s. 1437–1446, ISSN 1077-2626, doi:10.1109/TVCG.2018.2794058.

[14] Jerald, J.: *The VR Book: Human-Centered Design for Virtual Reality*. Morgan Claypool Publishers, 2015, ISBN 9781970001150.

[15] Korzuszek, P.: Understanding the Importance of Using Multiple Cameras in Unity. [Online].
URL https://blog.theknightsofunity.com/using-multiple-unity-cameras-why-this-may-be-important/

[16] Murata, R.; Songtong, S.; Mizumoto, H.; aj.: Teleoperation system using past image records for mobile manipulator. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, ISSN 2153-0858, s. 4340–4345, doi:10.1109/IROS.2014.6943176.

[17] Orság, F.: *Robotika ROB – Studijní opora*. VUT Brno, Fakulta informačních technologií, 2006.

[18] Sedlmajer, K.: *ROZVOJ WEBU TURISTICKÝ ATLAS, Bakalářská práce. Vedoucí práce prof. Ing. Adam Herout, Ph.D.* VUT Brno, Fakulta informačních technologií, 2017.

[19] Shah, S.; Dey, D.; Lovett, C.; aj.: AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics*, 2017, arXiv:1705.05065.
URL https://arxiv.org/abs/1705.05065

[20] Teixeira, J. M.; Ferreira, R.; Santos, M.; aj.: Teleoperation Using Google Glass and AR, Drone for Structural Inspection. In *2014 XVI Symposium on Virtual and Augmented Reality*, May 2014, s. 28–36, doi:10.1109/SVR.2014.42.

# Příloha A

# Plakát

# Příloha B

# Obsah přiloženého DVD

**thesis/technicka-zprava.pdf** technická zpráva ve formátu PDF

**thesis/video.mp4** prezentační video

**thesis/plakat.pdf** plakát v PDF

**thesis/latex** adresář zdrojových souborů technické zprávy ve formátu jazyka Latex

**src/** zdrojové soubory Unity projektu

**build/** aplikace zkompilovaná pro platformu Windows