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

NÁSTROJ PRO ŘÍZENÍ MISE LETKY DRONŮ

THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **IVAN HALOMI**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Halomi Ivan**
Program: Informační technologie
Název: **Nástroj pro řízení mise letky dronů**
Drone Squadron Mission Control Tool
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s tvorbou aplikací na platformě Unity, postupy návrhu klient-server architektury a GUI, a se systémem DroCo.
2. Navrhněte aplikaci pro operátora zásahu záchranářské jednotky využívající drony k monitorování situace. Zaměřte se na proces tvorby mise, včetně případných změn v průběhu mise. Navrhněte způsob editace misí na stanici operátora (PC/notebook) a GUI prvky pro klientské aplikace pilotů (tablet).
3. Při realizaci navrženého řešení se zaměřte na verzi pro stanici operátora. Využijte systém DroCo a další vhodné existující knihovny.
4. Řešení testujte na uživatelích a vyhodnoťte.
5. Prezentujte klíčové vlastnosti řešení formou plakátu a krátkého videa.

Literatura:

- Dieter Schmalstieg, Tobias Hollerer. *Augmented Reality: Principles and Practice.* Addison-Wesley, 2016. ISBN: 978-0321883575.
- SEDLMAJER Kamil, BAMBUŠEK Daniel a BERAN Vítězslav. *Effective Remote Drone Control Using Augmented Virtuality.* In: Proceedings of the 3rd International Conference on Computer-Human Interaction Research and Applications 2019. Vienna: SciTePress - Science and Technology Publications, 2019, s. 177-182. ISBN 978-989-758-376-6.
- Dále dle pokynu vedoucího.

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

Vedoucí práce: **Beran Vítězslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 9. listopadu 2021

Abstrakt

Cieľom tejto práce bolo preskúmať ako možno zjednodušiť orientáciu pilotov dronov v teréne pomocou orientačných bodov a vytvoriť užívateľské rozhranie, ktoré by to pilotovi efektívne umožňovalo ešte pred samotným letom. Navrhnuté riešenie bolo implementované do už existujúcej aplikácie. Moja práce rozšírila túto aplikáciu o možnosť efektívne vytvárať rôzne druhy orientačných bodov, vkladať ich do scény a následne pomocou nich vytvárať a ukladať spustiteľné misie.

Abstract

The goal of this work, was to find and create a solution to simplify drone pilots work in the terrain using orientation points in the terrain itself before the drone even begins its flight. A solution was put forth and implemented into an already existing application. My work further enhanceded this application with the ability to create various kinds of orientational points, ability to import these points into digital scenes, save the scenes and subsequently use them in playble missions.

Kľúčové slová

drony, kooperácia dronov, ovládanie dronov, rozšírená virtualita, virtuálna scéna, navigačné prvky, orientačné body, misia, tvorba misie, virtuálna mapa

Keywords

drones, drone cooperation, drone control, augmented virtuality, virtual scene, navigation elements, orientation points, mission, mission creation, virtual map

Citácia

HALOMI, Ivan. *Nástroj pro řízení mise letky dronů*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. VÍTĚZSLAV Beran, Ph.D.

Nástroj pro řízení mise letky dronů

Prehlásenie

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

.....
Ivan Halomi
9. mája 2022

Poďakovanie

Chcel by som poďakovať vedúcemu práce, Ing. Vítězslavu Beranovi, Ph.D, za cenné rady a podnetné nápady pri spoločných konzultáciách a takisto aj za podporu pri písaní práce na Erasme.

Obsah

Kapitola 1

Úvod

Moja práca sa zaoberá návrhom užívateľského rozhranie pre tvorbu misií a pridávnie objektov, v už existujúcej aplikácii, ktorá využíva rozšírenú virtualitu pre uľahčenie pilotovania dronu. Hlavným cieľom práce je návrh a implementácia užívateľského rozhrania, ktoré by umožnilo editovať virtuálne objekty, tvorbu a ukladanie spustiteľných misií.

V mojej práci som popísal rôzne využitia dronov, problematické situácie, kedy pilot potrebuje dodatočné orientačné prvky, ktoré táto aplikácia neumožňuje jednoducho pridávať. Tieto virtuálne body majú pilotovi uľahčiť orientáciu počas letu. Body sa dajú vkladať do ciest a vytvárať tak spustiteľné misie. Výsledné riešenie obsahuje niekoľko nových prvkov užívateľského rozhrania, ktoré umožňuje body pridávať, vkladať ich do misie a ukladať.

V prvej kapitole stručne popíšem stavbu dronov, rôzne typy ovládania, odvetvia, v ktorých sa drony využívajú a aké sú na ne kladené požiadavky. V krátkosti opíšem aj aktuálne používané aplikácie na ovládanie dronov a ich nedostatky. V závere tejto kapitoly zhrniem aké technológie využíva súčasná aplikácia Droco.

Nasledujúca kapitola sa venuje návrhu. V ňom zhrniem, čo všetko aplikácii chýba. Rozoberiem ako by sa tieto problémy dali vyriešiť, opíšem zoznam potrebných orientačných bodov, navrhnem spôsob a užívateľské rozhranie, ktorým by ich bolo možné pridávať na scénu a vkladať do misie.

V poslednej časti zhrniem implementáciu navrhnutých prvkov, popíšem najdôležitejšie pridané časti a problémy, na ktoré som narazil.Následne popíšem testovanie a overovanie funkčnosti aplikácie.

Kapitola 2

Drony

2.1 Drony

Dron je diaľkovo ovládané alebo autonómne vozidlo, ktoré môže byť schopné letu. Ak ide o lietajúci typ, môže sa označovať UAV (unmanned aerial vehicle), teda bezpilotné lietajúce vozidlo. V tejto práci budem používať slovo dron ako referenciu na konkrétny typ UAV, a to kvadrokoptéru. Dron sa zvyčajne skladá zo samotného lietajúceho objektu, ovládacej stanice a jeho pilota. Moderné drony využívajú skombinované dve možnosti ovládania. Autonómny let a manuálne ovládanie pilotom, kedy je dron primárne ovládaný pilotom, avšak jednoduché činnosti sú prenechané na program. Najlepší príklad automatizovanej činnosti je samostatný návrat dronu na miesto odkiaľ štartoval alebo držanie danej pozície. Zvládajú však aj komplikovanejšie úlohy napr. nasledovanie iného objektu, let na zadané GPS súradnice.

Stavba

Ak hovoríme o kvadrokoptérach, ich stavba pozostáva z niekoľkých základných súčiastok a ďalších rozšíriteľných modulov (obrázok 2.1). Pre základnú funkcionalitu sú potrebné rám, motory s vrtuľami, ESC (elektronický regulátor otáčok), batéria, flight controller a RC príjmač. Flight controller je mozgom celého dronu, keďže prijíma signály zo senzorov a rozkazy od pilota cez pripojený RC prijímač. Následne reguluje rýchlosť jednotlivých motorov a ostatných činností. Ako už bolo spomenuté RC prijímač slúži na komunikáciu s pilotom, a to konkrétne s jeho ovládacou stanicou. Pre pozemnú stanicu odosiela informácie o rýchlosti, výške, GPS súradniciach a iných dátach, ako aj prijíma príkazy ovládania. Často je dron obohatený o kameru, ak ide o FPV (first person view) kameru tak tá nie je na drone zvyčajne sama. Používa sa vo dvojici, kedy jedna kamera slúži na kvalitný záznam letu a druhá (zvyčajne s menším rozlíšením) ako samotná FPV. Pre prenos video signálu treba na dron ešte pripojiť video vysielač spolu s 5.8 gHz anténou. Kamera generuje signál, ktorý sa v flight controlleri obohatí o informácie z dronu a cez video vysielač je pohľad kamery s pridanými informáciami streamovaný do FPV okuliarov na pilotovej hlave alebo pozemnej stanice[14][6]. **RC ovládač** je zariadenie, ktoré komunikuje cez 2.4GHz komunikačné pásmo s RC prijímačom na kvadrokoptére a stará sa o riadenie drona. Existuje viacero spôsobov a módov ovládania, ktoré kvalitne spracoval vo svojej záverečnej práci pán Sedlmajster[15].

[1]Obrázok prevzatý z https://www.dronetechplanet.com/quick-drone-parts-overview-with-diy-tips/

Obr. 2.1: Popísaná základná stavba dronu[1]

O prenos a vizualizáciu dát z drona sa stará **pozemná stanica**. Najčastejšie je to tablet alebo mobilné zariadenie s pripojeným prijímačom videa a telemetrie a bežiacou aplikáciou, ktorú má takmer každý výrobca vlastnú. Stanica nie je priamo zodpovedná za jeho ovládanie, aj keď dnes je často pri drahších zariadeniach v balení RC ovládač obsahujúci integrovaný displej pozemnej stanice alebo je vytvorené miesto na vloženie telefónu priamo do ovládača. Pozemná stanica zobrazuje informácie o letových dátach prijatých z drona, ide hlavne o aktuálnu výšku, rýchlosť, smer letu, geografické súradnice, ale aj pozíciu na mape a prenos videa z kamery (ak je dron vybavený kamerou)[15].

Ovládanie

Drony je možné riadiť buď zo zeme, kedy dron neustále vidíme na oblohe alebo je možné využiť kamery umiestnené na drone. Pri druhom spomenutom spôsobe musíme využiť buď displej tabletu, telefónu, iného zariadenia s obrazovkou alebo špeciálne headset okuliare.

Riadenie zo zeme

Pri riadení zo zeme je pilot obmedzený tým, že zariadenie môže bezpečne ovládať len ak ho vidí. Okrem toho je pre pilota náročné určiť smer, ktorým bude dron letieť. Je to najmä z toho dôvodu, že zo zeme nevidí, ktorá časť dronu je tá „predná“. Tento spôsob je ideálny pre začiatočníkov, pretože ak je dron stále v našom dohľade, tak neustále vidíme jeho okolie a sme si vedomí blízkych objektov a prekážok. Hrozí však náraz na základe zlého odhadu vzdialenosti od objektu alebo smeru dronu. Takisto je veľmi obtiažne lietať v neprehľadnom prostredí, kde sa dron ľahko skryje za nejaký objekt, a teda sa stratí z dohľadu pilota. Pri tomto spôsobe ovládania je veľmi limitovaná oblasť v ktorej je možné lietať.

Riadenie pomocou kamery a displeja

V prípade daného spôsobu riadenia sa pilot orientuje hlavne podľa kamery umiestnenej na drone, ktorá je zväčša situovaná v prednej časti drona. Pilot sleduje displej s priamym prenosom obrazu z kamery a ovláda dron pomocou neho. Pilotovi však pomáha v orientácii aj to, že v prípade ak je dron v dohľade, môže pozerať na jeho okolie zo zeme voľným okom. Je teda logické, že pri strate signálu z kamery mimo zorného poľa pilota vzniká veľký problém.

Riadenie z pohľadu prvej osoby

Ide o čoraz populárnejší trend lietania, ktorému sa hovorí FPV (First Person View) lietanie. Ide o lietanie so špeciálnymi headset okuliarmi, ktoré prijímajú prenos z kamery. Tento názov pochádza z videohier, kde pohľad z prvej osoby znamená, že hráte z pohľadu očí vášho charakteru. To isté platí aj pri FPV lietaní, akurát namiesto charakteru máte dron a vidíte len to čo vidí jeho kamera. Pre neskúsených pilotov je toto ovládanie veľmi komplikované, pretože je veľmi zložité sa zorientovať v okolí drona len z pohľadu do jedného smeru, a tak veľmi ľahko môže prísť k nehode alebo nárazu a zničeniu drona. Z tohto dôvodu sa odporúča začať využívať okuliare až potom, ako sa správne naučíte ovládať dron a registrovať jeho okolie[15].

2.2 Legislatíva

K 31.12.2020 nadobudli platnosť delegované nariadenie Komisie (EÚ) 2019/945 a vykonávacie nariadenie Komisie (EÚ) 2019/47 (ďalej len ako "nariadenia"), ktoré obmedzujú a zároveň zjednocujú pravidlá pre všetky krajiny oblasti prevádzky dronov. Nahradzujú dovtedy platný doplnok X leteckého predpisu L2. Nariadenia ukladajú povinnosť registrácie na Úřadu pro civilní letectví každému pilotovi bezpilotného lietadla, ktoré :

- je ťažšie ako 250 gramov
- vyvinie pri dopade väčšiu silu ako 80 Joulov
- je vybavené kamerou alebo iným snímačom schopných zachytiť osobné údaje

Registrácia prebieha pomocou online testu skladajúceho sa zo štyridsiatich otázok (pre úspešné splnenie je potreba mať aspoň tridsať otázok zodpovedaných správne). Pilot po splnení získa oprávnenie platné nielen v Českej republike, ale aj v celej Európe. V zmysle vyššie spomenutých nariadení sa drony rozdeľujú do viacerých kategórií a podskupín a pre každú z nich špecifikujú ďalšie pravidlá a obmedzenia. Kategórie sú vymedzené nasledovne:

- Otvorená kategória - patria sem objekty, ktorých prevádzka nevyžaduje certifikáciu. Jedná sa hlavne o amatérsky využívané drony.
- Špecifická kategória - spadajú sem objekty, ktoré sa nevojdú do otvorenej kategórie a ďalej sa delia podľa spôsobu využívania.
- Certifikovaná kategória - kategória skôr do budúcnosti, ktorá špecifikuje proces certifikácie pre drony napr. na prepravu osôb.

Toto rozdelenie prebieha hlavne v závislosti od váhy lietajúceho objektu a jeho dopadovej energie. Obmedzenia sa týkajú najmä preletov ponad nezainteresované osoby, zhromaždenia ľudí, udržiavania vzdialenosti od rekreačných a obytných zón. Momentálne legislatíva pre otvorenú kategóriu, nepovoľuje lietanie s daným objektom mimo dohľadu pilota, a teda je lietanie výlučne pomocou FPV alebo kamery (dron je mimo dohľad) vo vonkajších priestoroch zakázané. Umožňuje však využitie "spottera", ktorý bude sledovať dron namiesto pilota. Pri špecifickej kategórii je toto umožnené, avšak len mimo obytných zón alebo nad zainteresovanými osobami. Aj z tohto pohľadu je teda veľmi dôležitý vývoj kvalitných aplikácii na uľahčenie lietania iba pomocou kamier a v záujme zaručenia absolútnej bezpečnosti všetkých okolitých ľudí[2].

2.3 Využitie dronov

Drony už dávno nie sú len vojenskou záležitosťou. Prenikli, respektíve postupne prenikajú takmer do každého odvetvia. Vďaka svojmu širokému spektru využitia a jednoduchému prístupu do ťažko dostupných oblastí sa stali výborným pomocníkom prakticky pre každého. Asi každý už videl úchvatné zábery hôr, prírodných či kultúrnych pamiatok použité v mnohých dokumentárnych filmoch, ktoré sú nasnímané práve pomocou dronov.Drony si našli významné využitie vo filmovom priemysle vďaka svojej cene, dostupnosti a jednoduchej manipulácii. Kvalita záberov ostáva rovnaká ako pri nakrúcaní pomocou helikoptér, pretože kamera sa jednoducho len umiestni pod dron.

Málokto však vie, že drony sa vďaka pokroku vo vývoji využívajú aj v menej očakávaných odvetviach. Ako príklad možno spomenúť poľnohospodársky priemysel, kde sa využívajú na sledovanie úrody v období dažďov, kedy je poľnohospodárom znemožnený vstup na polia z dôvodu rozmočenej pôdy. Vtedy pomocou ich záberov z multispektrálnych kamier je možné robiť analýzu pôd, plánovať výsadbu, optimalizovať zavlažovanie či zachytiť škodcov a choroby vo veľmi skorom zárodku a tak minimalizovať straty. Existujú aj špecializované agro-drony schopné postrekovať plodiny pesticídmi[13][4].

V stavebníctve sú drony využívané pri kontrolách stavieb, kde sa využitím dronu znižuje riziko zranenia zamestnancov a je umožnená kontrola aj z pohľadu, z ktorého by pomocou bežnej kamery kontrola nebola možná. Vhodným príkladom je stavba mrakodrapov alebo kontrola výškových komínov, kde človek bežne nedostane. Optimalizujú sa tak náklady, pretože chyba na vonkajšom plášti budovy sa ľahko odhalí. Okrem toho je veľmi jednoduché odhaliť úniky tepla vďaka dronom s termokamerami. Takéto zariadenie uľahčuje inšpekcie zariadení v energetike, kde kontroluje pomocou špeciálnych kamier úniky z potrubí, poruchy solárnych panelov či stĺpy vysokého napätia. Všetky spomínané činnosti boli doteraz vykonávané z helikoptér alebo lietadiel, čo sa samozrejme odrazilo na výške nákladov[11].

Ďalším menej očakávaným odvetvím je aj baníctvo, kde drony vytvárajú 3D mapy baní, získavajú dáta pomocou špeciálnych senzorov, prípadne vykonávajú iné dôležité úlohy, čo zvyšuje bezpečnosť a efektivitu v danom odvetví[17].

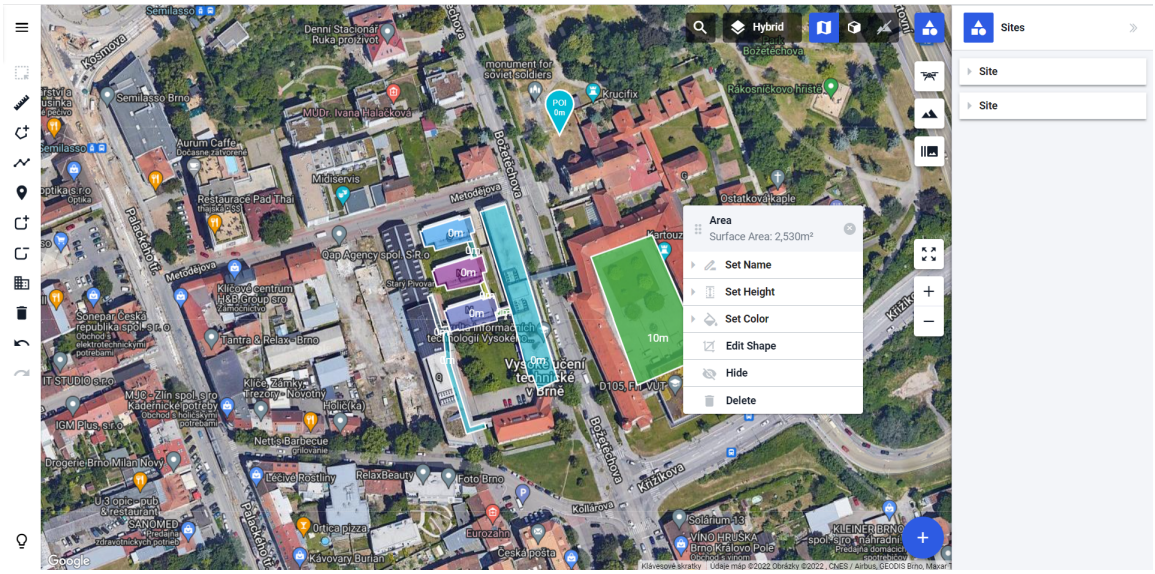
Pri pátracích či záchranných akciách umožňujú záchranným zložkám rýchlo preskúmavať danú oblasť. V prípade pátraní v lese po nezvestnej osobe dron s termokamerou pokryje oveľa väčší priestor ako dokáže človek zo zeme. Obrovskou výhodou, keďže ide o bezpilotné stroje je to, že dokážu lietať za takmer akýchkoľvek podmienok bez rizika zranenia inej osoby. Najmä v horských oblastiach je to nesmierne dôležité, keďže veľakrát sa osoba stratí alebo zraní práve v nepriaznivých podmienkach, ktoré nie sú bezpečné pre vzlietnutie helikoptéry, kde by pri lete by bola ohrozená aj samotná posádka. Pri dronoch môže dôjsť maximálne k materiálnym škodám, ktoré sú pri záchrane ľudského života zanedbateľné. Ta-

kisto je možné pripevniť a následne spustiť menšie až stredne veľké objekty zachraňovanej osobe, aby tak ľahšie zvládla prežiť do príchodu záchranárov. Môže sa jednať o vysielačku, lekárničku, termodeky, ale aj napríklad o samonafukovacie plavidlo, prípadne záchranné vesty pri obetiach vo vode. V prípade takýchto operácií je zvyčajne nasadených viacero dronov a o ich koordináciu sa stará operátor. Preto je veľmi dôležité aby všetci piloti videli aktuálne a presné informácie a rozkazy, ktoré im operátor zašle a nehrozili tak chyby ako prehľadávanie tej istej oblasti dvakrát[5].

2.4 Existujúce riešenia

S rozšírením využívania dronov sa vytvoril aj dopyt po kvalitných aplikáciach na zjednodušenie ich ovládania. Takýchto aplikácii je na trhu nespočetné množstvo. Drvivá väčšina je pre zariadenia s Androidom alebo iOS, avšak existuje aj niekoľko z nich, ktoré sú dostupné ako webová aplikácia. Zväčša majú obchodný model založený na voľnej základnej verzii s pár schopnosťami a platenej prémiovej verzii so všetkým čo ponúkajú. Niekoľko z tých najpopulárnejších som vyskúšal, aby som dokázal identifikovať ich nedostatky a na základe týchto poznatkov vylepšiť našu aplikáciu.

Typicky existujú dva režimy zobrazenia. Prvým je priamo pohľad z kamery drona s pridanými informáciami o rýchlosti, výške atď. Druhým je 2D mapa s označením vašej polohy, polohy drona a často sú tam vyznačené aj zóny podliehajúce určitej regulácii podľa platnej legislatívy (blízkosť letiska, vojenský objekt). Okrem predpokladaných funkcií slúžiacich na uloženie trajektórie letu a spárovanie s dronom ponúka každá aplikácia niečo iné.

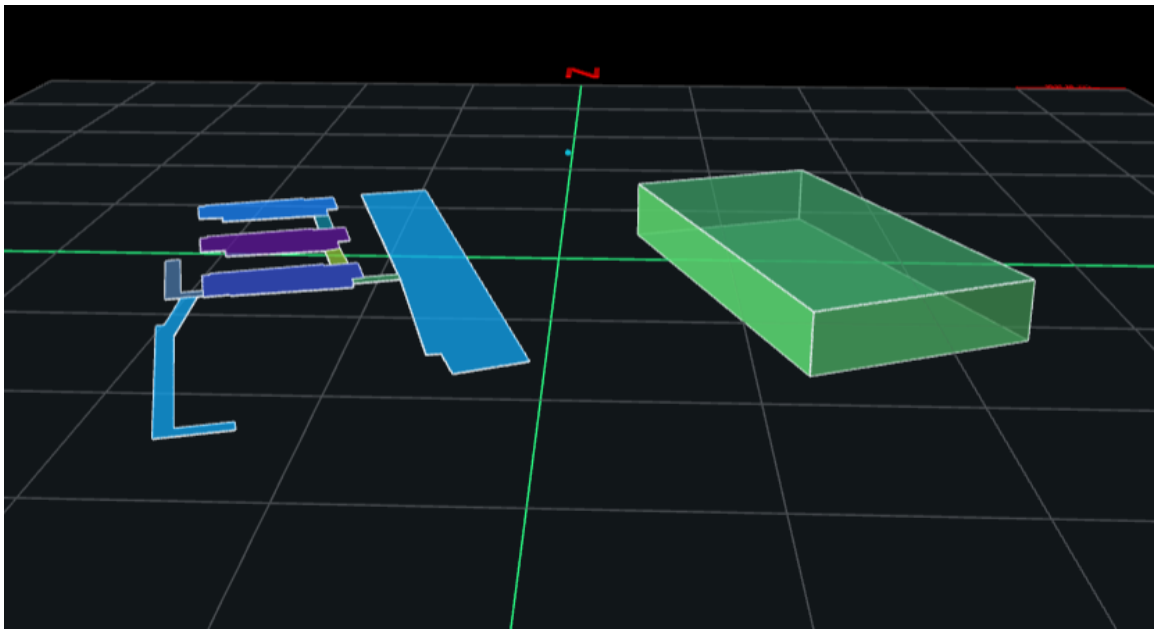


Obr. 2.2: Užívateľské rozhranie webovej verzie DroneHarmony s pár vytvorenými objektami. Oblasti s modrou a fialovou farbou sú vygenerované aplikáciou pomocou dát z mapového podkladu a zelená zóna je vytvorená manuálne s výškou 10 metrov

Ako príklad môžeme spomenúť DroneHarmony (obrázok 2.2), ktorá umožňuje vložiť vlastné body záujmu, pričom tieto môžu slúžiť ako waypointy a dron tak počas letu sám preletí po vytýčenej trase. Veľkým plusom je možnosť vygenerovať si zóny nad strechami budov, kedy sú tieto objekty vytvorené aplikáciou pomocou dát z OpenStreetMaps v zadanej oblasti. Takisto možno vytvoriť istú oblasť ručne a následne vygenerovať trasu drona

tak, aby zachytil na kameru celú oblasť čo najrýchlejšie. Okrem už uvedeného dovoľuje aj upravovať parametre týchto preletov, ako je sklon kamery, vzdialenosť medzi jednotlivými preletmi, výška atď. V prémiovej verzii je možné generovať aj trasy pre nahranie povrchu niektorých 3D objektov, ako je napr. budova.

Následne je možné dron poslať na túto trasu, ktorú sám preletí. Bohužiaľ zobrazovanie týchto orientačných bodov alebo oblastí záujmu nie je možné priamo v 3D priestore, pretože to aplikácia nepodporuje. Avšak umožňuje si vygenerovaný objekt pozrieť v 3D aspoň v rámci náhľadu mimo mapy (obrázok 2.3).



Obr. 2.3: Priestorový pohľad na vytvorené objekty z obrázka 2.2, kde vidíme rozdiel v ich výške

Pri vyhľadávaní informácií potrebných na vypracovanie tejto záverečnej práce som neprišiel do kontaktu so žiadnou aplikáciou, ktorá by dron dokázala zobrazovať vo virtuálnom prostredí pomocou informácií z neho. Ďalším nedostatkom týchto aplikácií bolo to, že zväčša podporovali len drony od firmy DJI.

Existujú aj komplexnejšie riešenia určené pre firmy zamerané na drony. Takouto aplikáciou je napríklad DroneDesk, ktorý slúži ako taký manažér pilotov. Je v ňom možné vytvárať pilotov, drony, zákazníkov, miesta výkonu práce a následne spájať do jednotlivých úloh. Manažér alebo zodpovedná osoba dopredu pripraví prehľad o mieste, na ktorom sa bude lietať, vloží nejaké poznámky a treba dať pozor na rizikové zóny. Nastaví trasu alebo body po ktorých treba letieť a pilot sa následne len prihlási do svojej aplikácie a stiahne si túto úlohu so všetkými informáciami.

2.5 DroCo

Moja práca nadväzuje na prácu výskumného tímu RoboFIT, ktorý už niekoľko rokov vyvíja mimo iného softvér na uľahčenie spolupráce človeka a robotov. DroCo vzniklo spoluprácou Karola Sedlmajstera a výskumného tímu ROBO@FIT s cieľom uľahčiť orientáciu pilota dronu v priestore a znížiť tak mentálnu zátaž pilota[15]. Na túto diplomovú prácu následne nadviazal Robert Hubinák, ktorý ďalej rozvinul potenciál aplikácie a pridal prvky, ktoré varujú pilota o nebezpečenstve blízkych objektov[9]. Aplikácia vytvára pomocou voľne dostupných mapových podkladov virtuálny priestor, do ktorého je umiestnený živý prenos z kamery dronu. Keďže vo svojej práci nadväzujem na prácu vyššie spomenutých autorov, zhrniem technológie doteraz zahrnuté v aplikácii, ktoré sú potrebné na pochopenie stavu, z ktorého začíname našu prácu.

Zmiešaná realita

Pojmy ako rozšírená realita alebo virtuálna realita sú známe aj pre laickú verejnosť, avšak zväčša sú chybne interpretované ako pomenovanie jednej a tej istej veci. Naša aplikácia využíva rozšírenú virtualitu, ktorá označuje rozšírenie virtuálneho sveta prvkami z reálneho sveta (v našom prípade dron, budovy). Rozšírená realita je umiestnením virtuálnych objektov do reálneho sveta, zatiaľ čo virtuálna realita je model reálneho sveta vo virtuálnom svete, kde môžu, ale aj nemusia byť dodržané fyzikálne zákony. Rozdiely medzi danými výrazmi popísal vo svojej práci Paul Milgram[12].

Unity

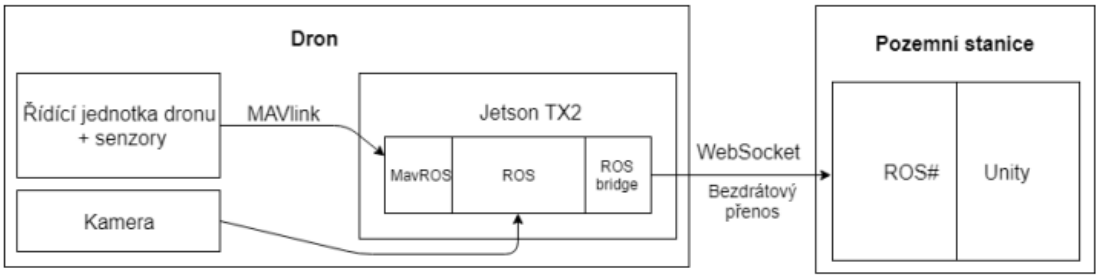
Cieľom aplikácie bolo vytvoriť 3D model scény/sveta vo virtuálnom prostredí a zároveň poskytnúť možnosť pilotovi interagovať s týmto prostredím pomocou určitého užívateľského rozhrania, takže bolo logickým krokom zvoliť niektorý herný engine ako základ aplikácie. Po dôkladnom zvážení tvorcovia zvolili Unity. Unity je multiplatformový herný engine na tvorbu 2D alebo 3D hier, mobilných aplikácií a webov, ako je dnes zvykom pri moderných herných enginoch. Ako príklad možno uviesť CryEngine alebo Unreal engine. Aj Unity je na nekomerčné účely zdarma. Jeho výhodou v porovnaní s ostatnými je veľkosť komunity, ktorá sa od jeho vytvorenia v roku 2005 stihla vybudovať, a teda je na internete obrovské množstvo návodov a knižníc, čo značne urýchľuje vývoj. Okrem toho je výhodou aj Unity Asset Store, ktorý je jeho súčasťou. Je na ňom dostupné veľké množstvo predpripravených komponentov, textúr, rozšírení od iných vývojárov. Bohužiaľ, z veľkej časti ide o platené assety. Hoci je Unity napísané v `C++`, ako skriptovací jazyk využíva `C#`.

Aplikácie v Unity fungujú pomocou scén a prepínania medzi nimi. Scény sú od seba navzájom nezávislé a každá scéna má vlastný hierarchický zoznam svojich objektov typu GameObject. GameObject je základný objekt, s ktorým pracuje Unity. Jediným povinným komponentom je transform alebo RectTransform pre 2D objekty. K tomuto základu je možné pridávať komponenty rozširujúce jeho vlastnosti. Typickým komponentom upravujúcim vizuálnu stránku objektu je **Mesh Filter** a **Mesh Renderer**, ktoré dodávajú objektu tvar a farbu, prípadne textúru. Ďalej je možné ako komponenty vytvárať skripty definujúce správanie objektu. Je možné definovať a pripnúť aj niekoľko rôznych skriptov pre jeden objekt. Tieto objekty sa dajú ukladať a načítavať ako *prefaby*. Pri inicializácii prefabu vzniká kópia daného objektu, čo pomáha pri vytváraní tých istých alebo veľmi podobných objektov. Pri 3D aplikácii sa takmer vždy aplikácia skladá z dvoch častí. 3D scény s daným modelom sveta pracujúceho so súradnicovým poľom x, y, z. Ten zobrazuje priamo

„hrateľnú“ časť a 2D plátna, v ktorom sú umiestnené prvky užívateľského rozhrania, ako rôzne tlačidlá, texty, panely[1].

Komunikácia s dronom

Samotnou podstatou aplikácie je virtualizovať informácie z dronu v reálnom čase, pričom na to musí byť aplikácia schopná komunikovať s dronom. Unity ako herný engine nemá priamu podporu komunikácie s robotickým softvérom. Bolo potrebné nájsť určitú platformu, knižnicu, ktorá by spĺňala požiadavky komunikácie. Bol zvolený protokol ROSBridge z open source projektu ROS - Robotic Operation System. Nejde o operačný systém ako by názov indikoval, ale o knižnicu ovládačov, modulov, nástrojov na ovládanie súčastí robotických zariadení, ako sú senzory alebo kamery. Cieľom ROS-u je urýchlenie a uľahčenie vývoja poskytnutím jednotnej modernej platformy rozširovanej komunitou. ROS architektúra je založená na vzájomnej komunikácii uzlov. Hlavný uzol ROS master spracúva a uchováva zoznam ostatných uzlov a stará sa o ich komunikáciu. Uzol, ktorý produkuje nejaké dáta napr. senzor, sa prihlási ako publisher. Naopak uzol, ktorý chce nejaké dáta získať je subscriber. ROSbridge poskytuje rozhranie pre komunikáciu so zariadením bežiacim na ROS pre aplikácie, ktoré nie sú vytvorené na princípe ROS. Pre komunikáciu s ROSbridge serverom aplikácia využíva RosSharp plugin z Unity Asset Store[2][9][15].



Obr. 2.4: ROSbridge spracúva a odosiela dáta z ROS-u pomocou WebSocket protokolu do RosSharp, ktorý ich prekladá Unity[4].

[2] http://wiki.ros.org/
[4] Obrázok prevzatý z [15]

Kapitola 3

Návrh

Drony sú čoraz častejšie využívané pri mnohých činnostiach spomenutých v sekcii 2.3. Je teda potrebné aby piloti dokázali mať veľký prehľad o okolí dronu. Pri nenáročnom lietaní nad lúkou je pilot pokojný a všetko vníma, no pri lietaní v neznámom teréne alebo bez možnosti sledovať dron zo zeme sa môže pilot cítiť pod stresom alebo tlakom. Navyše, ak má sledovať ostatné drony a dávať pozor aby nedošlo k zrážke, tak je veľmi dôležité, aby sa vedel rýchlo zorientovať v priestore. Teda aplikácia by mala zobrazovať tieto informácie čo najjednoduchšie. Vkladanie **virtuálnych objektov** do scény zľahčuje orientáciu a prehľad o okolí. Tieto objekty môžu mať rôzne využitie a je teda vhodné aby existovalo viacero typov. Aplikácia by mala tiež poskytnúť možnosť rýchlo upravovať pridané objekty. Tieto objekty by pri misiách alebo operáciach, kde spolupracuje viacero pilotov, mali byť zobrazované na všetkých zariadeniach rovnako. Z tohto dôvodu, je nutnosťou, ponúknuť možnosť pripraviť si scenár dopredu a vytvoriť z neho misiu, ktorú aplikácia musí vedieť uložiť. Následne by sa mal tento súbor nahrať do zariadení pilotov lietajúcich v misii, ktorý potom budú schopní danú misiu načítať a spustiť.

Nedostatky súčasnej aplikácie

V súčasnej aplikácii existujú dve odlišné riešenia na zobrazovanie objektov, ktoré majú uľahčiť orientáciu pilota v teréne. Tieto riešenia sú nezávislé od seba, a teda medzi sebou vôbec nespolupracujú. Dá sa povedať, že obe implementujú to isté, ale iným spôsobom.

Jedno z nich načítava misiu ako celok z dopredu vytvoreného súboru. Pričom toto riešenie takisto poskytuje základné možnosti postupnej navigácie pilota po scéne, a teda nasledujúci kontrolný bod sa zobrazí až po príchode k predchádzajúcemu. Avšak neumožňuje žiadne úpravy a ani pridávanie objektov po spustení misie. Neexistuje ani jednoduchý spôsob ako misiu vytvoriť a je potrebné manuálne napísať JSON súbor s objektmi. Toto riešenie je schopné zobraziť ciele misie ako checkpointy a zóny. Checkpointy sa rozdeľujú na dva typy. Pri prvom type sa nevyžaduje akcia a stačí k nim priletieť pre jeho splnenie. Druhá možnosť slúži v momente, kedy je pri danom checkpointe potrebné vykonať dopredu danú akciu (zobrazenú v popise). Toto je implementované pomocou odkliknutia tlačidla o splnení.

Druhá možnosť poskytuje vytváranie smerových bodov, buď pomocou kliknutia na miesto, kde sa majú vytvoriť, a teda bod sa vždy nachádza na zemskom povrchu alebo vytvorenie na aktuálnej pozícii a výške drona. Bohužiaľ tieto body nemajú veľké využitie, pretože sa nedajú pomenovať ani farebne označiť. Aplikácia takisto umožňuje vytvoriť zónu, ktorá vždy siaha až do „neba“. Zóna sa vytvára pomocou bodov, ktoré je následne možné

aj presúvať. Momentálne je však možné vytvoriť týmto spôsobom len jednu zónu, ktorá nemá ani popis, ani názov. Vytvorené objekty sa po vypnutí aplikácie nikam neukladajú. Pilot musí vždy po spustení simulácie letieť na dané miesta a vytvoriť si body pre neskoršie využitie, čo zaberá množstvo času. Moja práca by mala tieto nedostatky odstrániť a prepojiť dve nespolupracujúce riešenie do jedného.

3.1 Orientačné body

Pilot sa počas letu môže orientovať pomocou viacerých možností. Najjednoduchšou je pohľad priamo na dron. Čo však robiť, ak pilot dron nevidí? Orientuje sa pomocou kamery a jej pohľadu z dronu. V tomto obraze sa musí zorientovať pomocou reálnych objektov, ktoré vidí. Jedná sa najmä o známe alebo netypické budovy, komíny alebo iné ľahko rozlíšiteľné miesta.

Ako už bolo spomenuté skorej, aplikácia ponúka navigačné možnosti, ktoré vkladajú určité orientačné body. Neumožňujú ich však nijako upravovať alebo dokonca pridávať inak ako pomocou JSON súboru. Naším cieľom bude daný nedostatok vylepšiť a umožniť tak pilotovi vkladanie upraviteľných bodov, či už počas lietania alebo pri pripravovaní si scenáru dopredu.

Checkpoint

Checkpoint by mal byť spojením dvoch rôznych, ale zároveň parametricky tých istých bodov. Prvým z nich je kontrolný bod počas misie. K tomuto bodu sa pilot chce dostať, a teda bod je jeho cieľom ku ktorému, by mal byť pilot navigovaný. Avšak bod nemusí byť natoľko dôležitý, aby pilot musel k nemu zaletieť. Zároveň však môže výrazne uľahčiť orientáciu v priestore. V tomto prípade by šlo o bod orientačný, ktorý má presne tie isté parametre ako kontrolný bod, akurát nie je k nemu pilot navigovaný.Aplikácia by mala takisto zobrazovať rozdiel medzi bodmi na zemskom povrchu a vo vzduchu, tak aby bolo na prvý pohľad zjavné či sa jedná o pozemný alebo vzdušný bod.

Pilot by mal byť schopný, takýto bod vytvoriť rýchlo a jednoducho, počas lietania jedným kliknutím. Išlo by samozrejme o bod v aktuálnej pozícii s vygenerovaným menom. Avšak pre prípad, ak by si pilot chcel scenár letu pripraviť dopredu a všetko si označiť a popísať, tak by mal mať možnosť vkladať body, či už pomocou kliknutí na miesto, kam chce bod vložiť alebo pomocou zemepisnej výšky a šírky. Zrejme si tiež bude chcieť body od seba farebne odlíšiť alebo vytvoriť skupiny bodov s rovnakou farbou, takže by mal pri vytváraní bodu, dostať možnosť vybrať mu farbu. Z dôvodu širokého spektra využití je vhodné takýto checkpoint rozdeliť na tri podkategórie.

Simple Checkpoint

Jednoduchý bod, ktorý by bol využívaný pri scenároch s pár bodmi, kde meno alebo farba nehrajú rolu. Mal by to byť základný bod, ktorý by nasledujúce kategórie len rozšírili. Bod by mal názov vygenerovaný automaticky. Ďalej potrebujeme vedieť zemepisné súradnice, ktoré môžeme získať z aktuálnej pozície dronu alebo z kliknutia od užívateľa a farbu, ktorá bude náhodne pridelená. Keďže kliknúť „do vzduchu“ sa nedá, je potrebné, aby pri tejto forme pridávania bola ponúknutá zmena výšky bodu, napr. pomocou posuvného slideru. Tento objekt nepotrebuje popis, možnosť zvoliť farbu či veľkosť, pretože užívateľ ho chce

vložiť čo najrýchlejšie a nechce byť zdržovaný zadávaním alebo potvrdzovaním zbytočných informácií.

Detailed Checkpoint

Detailnejší bod s možnosťou upraviť takmer všetky jeho vlastnosti. Nepredpokladá sa, že tento bod by bol vkladaný počas samotného letu, a teda je v poriadku, ponúknuť používateľovi možnosť, zmeniť všetky jeho parametre. Tieto vlastnosti objektu by mali byť nastavené na určitých základných hodnotách a následne si pamätať posledné zadané hodnoty, aby sa v prípade opakovaného vkladania sebe podobných bodov, proces čo najviac urýchlil. Využitie možno očakávať predovšetkým pri tvorení komplikovanejších scenárov, kde je viacero bodov a pilot si ich môže navzájom mýliť aj napriek rôznej farbe a názvu. V tomto prípade by mu bol nápomocný bližší opis bodu. Rovnako by to uľahčilo vrátenie sa k určitému scenáru po dlhšej dobe a celkovú prácu s ním. Vhodným rozšírením oproti jednoduchému checkpointu, by bola aj možnosť nastaviť detekčnú zónu, teda zónu v ktorej sa dron považuje dostatočne priblížený, a teda kontrolný bod je splnený. Používateľ by mal byť schopný vložiť popis bodu, zmeniť farbu alebo veľkosť bodu, nastaviť polohu pomocou zemepisných súradníc a výšky, rovnako ako pri predchádzajúcom objekte.

Circle Checkpoint

Tento typ by našiel využitie skôr v misiách ako nejaký orientačný bod. Malo by ísť o kruhový kontrolný bod, ktorého cieľom by bolo usmerniť pilota, aby preletel vnútri jeho okrajového obrysu. Body s podobným konceptom poznáme z mnohých hier (napr. GTA V, obrázok 3.1). Objekt by mohol slúžiť napr. na vytváranie tratí a pretekov, v ktorých by mohli jednotliví piloti medzi sebou súťažiť. Takisto by výrazne pomohol pri navigácii po presne danej krivke vo vzduchu. Keďže ide o kruhový objekt v 3D priestore, je veľmi dôležitá jeho rotácia. Pri ostatných objektoch kvôli ich modelom, takmer nezáleží na tom ako sú otočené, avšak pri tomto to možno vnímať ako zásadný rozdiel. Zladiť rotáciu na troch osiach je náročná úloha aj v prípade, ak objekt vidíte. Preto by mal užívateľ dostať pri tvorbe tohto bodu určitú pomôcku alebo hodnoty, ktoré by ho usmernili a uľahčili mu tak tvorbu. Nepredpokladá sa, že by objekt zadával pilot počas letu, to znamená, že bude môcť využiť pohyb kamery po priestore aby bod pridal. Významnou pomôckou môže byť aktuálna rotácia kamery. Tá by sa nastavila ako základné hodnoty bodu, po kliknutí na jeho pridanie. Okrem rotácie bude možné vložiť popis, upraviť farbu, vnútorný priemer kruhu a zemepisné súradnice s výškou.

Zóna

Niekedy ako orientačná pomôcka nestačí len jeden bod, ale treba ich niekoľko. Zvyčajne slúžia na vyznačenie určitého priestoru, v ktorom sa niečo deje. Môže sa jednať o oblasť, v ktorej treba niečo nájsť, zakázanú oblasť, do ktorej nemožno vletieť za žiadnych okolností a pod. Zóna by mala existovať presne pre takéto prípady. Mala by byť schopná jednoducho, rýchlo a intuitívne zobraziť určitú oblasť. Zóna by sa vykresľovala pomocou množiny bodov so zemepisnou šírkou, dĺžkou, nadmorskou výškou a stenami medzi nimi. Z dôvodu, že nie vždy je zóna nekonečne vysoká, okrem bodov na zemi, ktoré určujú polohu stien je vhodné nechať užívateľovi možnosť nastaviť aj výšku stien. V základnom nastavení by mohla siahať zóna niekoľko stoviek metrov vysoko. Zóna by mala mať názov, výšku, farbu a popis.

[1]Obrázok prevzatý z https://www.gameinformer.com/b/news/archive/2013/10/07/gta-v-39-s-air-race-checkpoints-are-dangerous.aspx

Obr. 3.1: Circle checkpoint v hre GTA V[1]

NPC

NPC, teda Non playable character, by našiel využitie v mnohých scenároch. Doteraz spomenuté body zdieľali jednu spoločnú vlastnosť, a to, že boli statické. Čo však v prípade, ak bude úlohou sledovať určitý pohyblivý cieľ? Presne z tohto dôvodu, je potrebné vytvoriť nehrateľný charakter, ktorý by sa pohyboval nezávisle od užívateľa. Môže sa jednať o nejaké vozidlo, ktoré treba sledovať, hľadanú osobu, prípadne strážnika, ktorému sa treba vyhnúť. Takisto sa môže jednať o nejaký objekt, ktorý síce nie je pohyblivý, ale reprezentovať ho pomocou bodu alebo zóny, nie je vhodné. Z uvedeného dôvodu je dôležité, aby NPC objekt nebol definovaný jedným modelom. Logické je zobraziť prenasledované auto modelom auta, ako aj pri hľadaní niečoho iného modelom danej veci. Z tohto dôvodu by mal užívateľ okrem pár pripravených modelov (napr. auto, človek, dron), byť schopný nahrať a použiť aj vlastné modely objektov. Keďže sa môže jednať aj pohyblivý typ, tak pohyb objektu treba odniekiaľ získať. Môže sa jednať o reálne dáta priamo od modulu v objekte (ako je to aj v prípade nášho dronu) alebo len o náhodné generované dáta nejakým serverom. Aplikácia by mala byť schopná dopytovať sa na aktuálnu polohu a následne ju aj nastavovať. Ak sa objekt pohybuje je zbytočné mu nastavovať počiatočnú polohu, keďže si ju získa sám, ale existuje možnosť, pri ktorej je objekt stojatý a polohu je nevyhnutné zadať. Popis správania alebo objektu by bol v mnohých situáciách tiež potrebný.

Dron

Ak je v misii viacero dronov môžeme povedať, že okrem nášho sa jedná o NPC objekty, avšak myslím, že je vhodným riešením oddeliť dron ovládaný iným človekom do samostnatnej kategórie. Tento objekt by sa nemal dať pridať rovnakým spôsobom ako ostatné. Možnosť pridať by existovala len počas vytvárania misie a dron by mimo nej neexistoval. Ak by sme chceli dron, ktorý je na scéne stále, vytvorili by sme NPC objekt s modelom dronu. Pri tomto type by sme však dopredu vedeli, že daný dron bude niekto počas misie pilotovať a

že bude pracovať na rovnakých cieľoch ako my. Je to najmä z dôvodu rozlíšenia objektov, ktoré majú určitý cieľ a tých, ktoré sa len voľne pohybujú. Treba vedieť názov, počiatočnú zemepisnú polohu alebo adresu, kde získavať aktuálnu polohu.

Kamera

Posledným objektom, ktorý by veľmi uľahčil prácu s aplikáciou DroCo je kamera. Aj keď kamera nebude mať veľmi veľké využitie pri samotnom lietaní, pretože pilot chce väčšinu času sledovať svoj dron. Pri práci na vytvorení scenára bude významnou pomocou. Veľakrát chceme mať rôzne pohľady na to, čo vytvárame a častokrát sa nám to nepodarí vytvoriť podľa predstáv na prvýkrát. Z tohto dôvodu je vhodné disponovať možnosťou uložiť si aktuálnu pozíciu kamery a následne mať možnosť sa na ňu rýchlo prepnúť. Nepredpokladám žiadne využitie v rámci misií, pretože kamera nemôže byť cieľom, ale je vhodné reprezentovať ju určitým spôsobom v priestore, aby pilot vedel o možnosti prepnutia kamery na dané miesto a skontrolovania situácie. Pri ukladaní kamery potrebujeme vedieť jej názov, polohu a hlavne rotáciu.

Vkladanie objektov

Ako bolo spomenuté pri nedostatkoch existujúcej aplikácie (viz. 3), vkladanie objektov síce existuje, ale je nedostatočné pre vytvorenie komplikovanejšieho scenára. Pravdepodobne najväčším problémom súčasného riešenia, ktorý do veľkej miery obmedzuje pridávanie nových bodov, je nemožnosť pohybu po scéne inak ako lietaním s dronom. Voľný pohyb kamery by mal byť elementárnym prvkom jednoduchej práce s bodmi. Pomocné by boli aj funkcie na obnovenie pozície kamery na dron alebo 2D mód, pri ktorom by sa prepla aplikácie do práce s 2D priestorom. Veľkým problémom pri vkladaní objektov v 3D priestore je nemožnosť získať všetky tri súradnice pomocou jedného kliknutia. V našom prípadne ide o nemožnosť získať výšku objektu pri kliknutí, keďže kliknúť „do vzduchu“, tak aby to Unity správne rozpoznalo, nie je možné. Tento nedostatok bohužiaľ inak ako manuálnym zadaním výšky vkladaného objektu nevieme odstrániť. Z toho vyplýva, že je nutné zobraziť **dialógové okno** aj pri vkladaní najjednoduchšieho bodu. Pri ostatných bodoch je však potrebné zadať aj iné parametre pre správne vytvorenie objektu, preto nepovažujem toto riešenie za nepraktické. Vytváranie bodov by malo byť čo najjednoduchšie a nemalo by otravovať s vypĺňaním nepodstatných informácií. Všetky povinné atribúty by mali byť predvyplnené, tak aby umožnili pridať bod len potvrdením dialógového okna. Samozrejme užívateľ musí mať možnosť, tieto atribúty upraviť podľa svojich predstáv.

3.2 Práca s objektmi po vložení

Užívateľ si vytvorí scenár, vloží všetky potrebné objekty a následne zistí, že niekde spravil chybu. Je nelogické aby musel objekt zmazať a vytvoriť nanovo. Preto treba zaistiť, aby boli objekty upraviteľné aj po vložení. Celkovo by mal mať užívateľ prehľad o všetkých objektoch na scéne v prehľadných zoznamoch. Pre prácu s nimi bude vhodnou pomôckou možnosť centrovania objektov, a teda presunutie kamery pár metrov pred objekt. Pri úprave polohy objektu, môžeme využiť opätovné kliknutie niekam na mape, ale aj presun na špecifickú zemepisnú polohu zadaním súradníc do textového poľa. Mali by sme predpokladať, že objekt už je súčasťou vytváranej misie, a teda musíme prepočítať všetky „pomocné čiary“ pri zobrazení ciest.

Ukladanie objektov

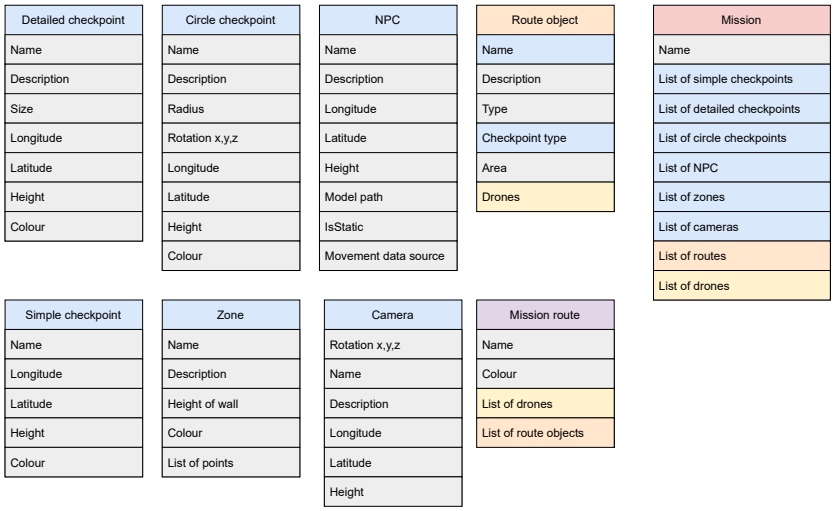
Vkladanie objektov, ich editácia a celková príprava scenára postráda zmysel, ak by nebolo možné s danými objektmi pracovať aj na inom zariadení alebo po zatvorení aplikácie. To znamená, že užívateľ si bude chcieť scenár vytvoriť, uložiť a následne pred lietaním načítať. Keďže na scéne sú aj objekty, ktoré nie sú potrebné priamo v misii, ale môžu byť taktiež podstatné, je dôležité ukladať, všetko čo užívateľ vytvoril. Z tohto hľadiska vieme rozdeliť ukladanie na dva typy. Objekty sú na scéne ako orientačné body a neexistuje správna postupnosť ako má medzi nimi pilot zaletieť alebo táto postupnosť existuje a jedná sa o misiu. Preto by mal užívateľ dostať na výber ako chce vytvorený scenár uložiť.

Ukladanú **misiu** môžeme chápať ako logickú postupnosť bodov, ktoré daný dron musí navštíviť práve v určenom poradí. Avšak, nie vždy existuje správna postupnosť bodov alebo jednoducho body len označujú dôležité miesta na scéne. V tomto prípade je vhodné uložiť scénu len ako jednoduché objekty. Aplikácia teda bude poskytovať dve možnosti ukladania objektov. Prvým je jednoduchý scenár s orientačnými bodmi. Druhým, viac komplexnejším typom, bude misia. Rozdielom bude, že pri misii bude potrebné zoradiť dané objekty podľa postupnosti, ktorá určuje poradie v akom sa k nim má dron dostať. Táto postupnosť bodov sa pre každý dron môže líšiť. Môžeme uvažovať, že pri vytváraní misie má človek dostatok času, a teda netreba aby bol tento postup čo najjednoduchší. Dôležitejšie je, aby bol systém univerzálny a umožňoval čo najširšie využitie, či už pri jednoduchých alebo veľmi rozsiahlych misiách. Pri načítaní opäť platí rovnaké rozdelenie. Nie vždy chce pilot zapnúť misiu a byť navigovaný od bodu k bodu. To znamená, že treba docieliť kompatibilitu medzi oboma typmi ukladania. A teda načítanie uloženej misie ako scenára bude viesť k načítaniu všetkých jej objektov, ale nezačne sa proces navigácie k prvému cieľu. Takisto načítanie jednoduchého scenára bez žiadnych cieľov, ako misie bude viesť k načítaniu všetkých objektov, ale navigácia sa nespustí, keďže cieľ neexistuje. Samozrejme načítanie misie ako misie bude viesť k načítaniu všetkých bodov, zobrazeniu ostatných dronov spolupracujúcich na misii a začatiu navigácie k prvému cieľu. Zoznam všetkých údajov, ktoré je potrebné uložiť je zobrazený na obrázku 3.2.

Vytváranie misie

Ako bolo spomenuté skorej, tak nejde o najjednoduchšie riešenie, ale o riešenie, ktoré pokrýva najširšiu škálu situácií. Či už ide len o jednoduché lietanie od miesta k miestu alebo o rozsiahly scenár s niekoľkými dronmi a desiatkami bodov. Uvažoval som nad viacerými možnosťami.
Prvou, najjednoduchšou, bolo zaletieť cestu s dronom, odkliknúť miesta, ktoré budú cieľom a následne ju len uložiť a zobraziť. Takýto postup však trvá veľmi dlho a nie je veľmi využiteľný. Ďalším riešením, nad ktorým som uvažoval, bolo vytváranie misie spôsobom, že by užívateľ začal pridávať jednotlivé body a v danom poradí by sa aj ukladali do misie. Tu bol problém pri opätovnom využití bodov, keďže by ich bolo nutné vždy nanovo vytvoriť pre každú jednu cestu. Práve z tohto dôvodu som sa rozhodol navrhnúť proces, ktorý by recykloval už vytvorené body a podporoval by vytváranie rôznych ciest pre rôzne drony . Proces tvorby misie je rozdelený na pridávanie samotných bodov a ich vkladanie do ciest. Užívateľ teda musí najprv vytvoriť všetky objekty, ktoré využije v rámci misie. Tvorba misie potom bude prebiehať len pomocou grafického rozhrania a niekoľkých okien. Toto rozhranie by malo efektívne umožniť vytvárať cesty a prideľovať k nim drony.

Cesta je postupnosť bodov určených pre jeden alebo viacero dronov, avšak celá postupnosť je rovnaká pre všetky drony pridané do danej cesty. Túto cestu je po vytvorení

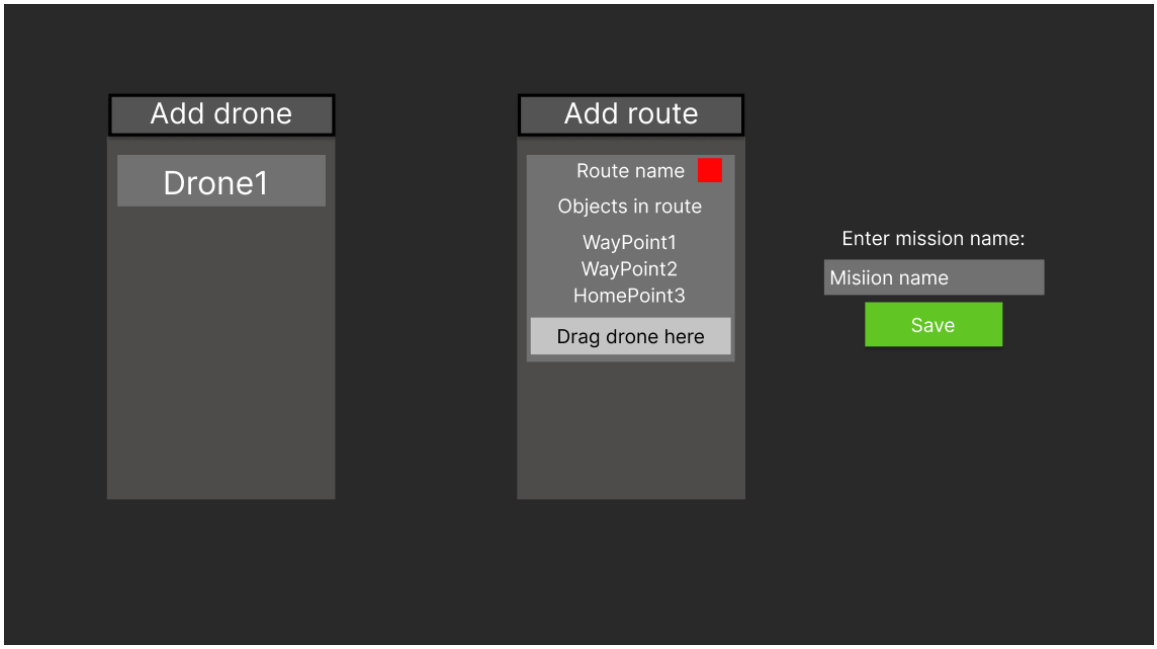


Obr. 3.2: Diagram zobrazujúci atribúty, ktoré potrebujeme pre znovu-vytvorenie orientačných bodov a misie po načítaní

vhodné nejako reprezentovať v rámci prípravy scenára. Vhodným zobrazením by bolo napr. prepojenie pomocou čiar medzi bodmi podľa ich poradia. Každá cesta by mala svoj názov a farbu, ktorou by bola reprezentovaná.

Rozhranie by teda malo byť rozdelené do dvoch častí. Keďže v naša aplikácia primárne pracuje s jedným dronom, treba aby tam ostatné drony boli pri tvorbe misie pridané. V prvej časti by sa pridávali nové drony a priraďovali by sa do už vytvorených ciest. Okno (obrázok 3.3) by teda obsahovalo dva zoznamy. Zoznam dronov v misii a zoznam ciest v misii. Dron by sa dal zaradiť do vytvorenej cesty a tým by sa mu určila ako cieľ. Ciest môže byť viacero, preto by mali byť, okrem ich názvu, zobrazené aj body, ktoré cesta obsahuje.

Na **vytvorenie cesty** by existovalo samostatné okno (obrázok 3.4). V danom okne by malo byť možné usporiadať vytvorené body do logickej postupnosti, teda do zoznamu. Keďže môžeme pracovať s veľkým množstvom objektov, tak najlepšou a najintuitívnejšou možnosťou na ich zoradenie mi prišiel **„drag&drop“** systém. Pridanie objektu do cesty sa vykonáva pomocou „pretiahnutia“ panelu, reprezentujúceho objekt, z jedného listu do druhého. Aby užívateľ nemusel prechádzať všetky vytvorené objekty, mala by existovať možnosť filtrovať ich podľa typu. Objekt môže byť cieľom opakovane, a preto by po pridaní do cesty, nemal zmiznúť zo zoznamu všetkých objektov. Keďže z pôvodného popisu objektu, zadaného pri jeho tvorbe, nemusí byť jasné, prečo je cieľom, je potrebné umožniť zadať popis daného objektu len v rámci vytváranej cesty. Tento popis by bol následne zobrazený, ak by bod bol aktuálnym cieľom. Takisto sa môže jednať o rôzne typy cieľa. Požadovaný typ



Obr. 3.3: Mockup navrhnutého rozhrania slúžiaceho na organizáciu dronov do jednotlivých ciest v rámci jednej misie. Vľavo je zoznam dronov a vpravo sa nachádza zoznam existujúcich ciest.

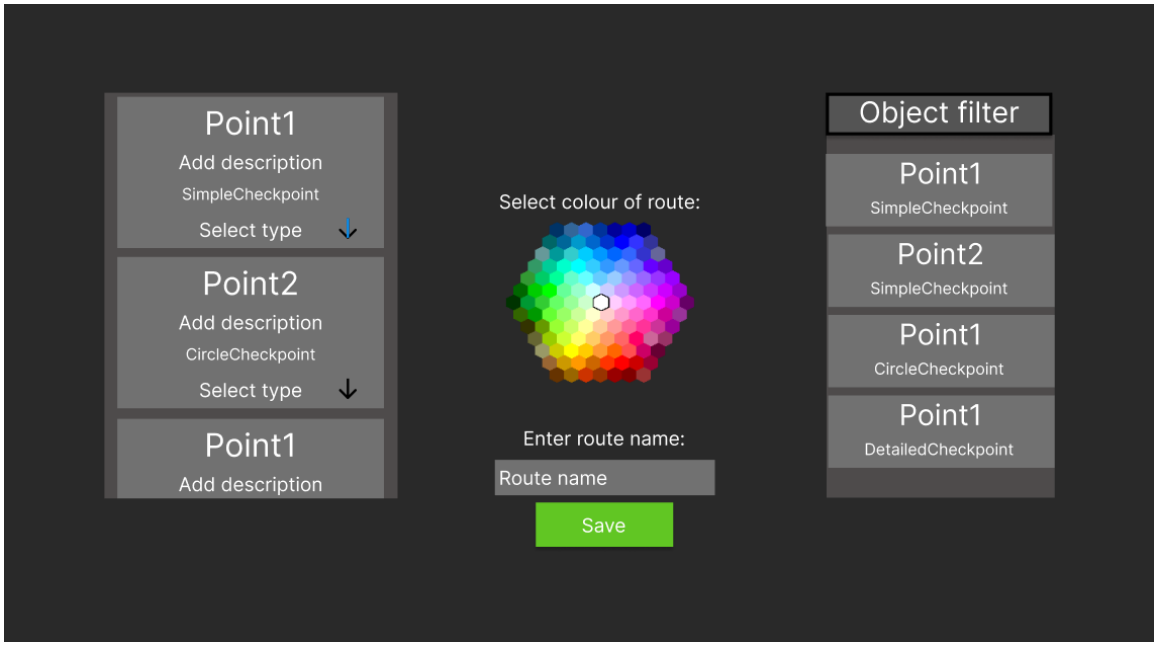
by sa mal dať nastaviť výberom z dostupných možností. Celkovo by teda okno pozostávalo z dvoch zoznamov, textovým poľom na zadanie názvu a vzorkovníkom farieb.

3.3 Uživateľské rozhranie

Ako môžeme vidieť na obrázku 3.5, aplikácia obsahuje užívateľské rozhranie, ktoré zaberá značnú časť obrazovky. V ľavej časti sa nachádza panel, kde sa dajú upraviť parametre spustenej simulácie ako napr. zemepisná šírka, ROS adresa, nastavenia videa a kamery. Pod ňou je dotykové ovládanie kamery. Vpravo sa zobrazujú informácie o aktuálne spustenej misii, **aktuálnom cieli** a ostatných dronoch na scéne. Tento panel s misiami zaberá pilotovi **1/5 obrazovky**, aj keď žiadna misia **nebeží**. Pod ním sa nachádza minimapa a vedľa nej súbor štyroch tlačidiel. Prvé umožňuje pridať/odstrániť orientačné body a nastaviť bod ku ktorému sa spustí navigácia. Druhé slúži na reštartovanie pripojenia k ROS adrese. Tretie je na vytváranie a editovanie zóny. Posledné mení možnosť ovládania (náhodné lietanie, ROS alebo ovládač).

Zo všetkých týchto prvkov ide schovať len panel s nastaveniami. Z tohto som usúdil, že na pridanie nových prvkov, tak aby príliš neobmedzili pilota, nie je priestor bez schovania už existujúcich. Po vyskúšaní viacerých návrhov som sa rozhodol, že najlepšie bude rozdeliť rozhranie na elementy využiteľné pri lietaní a na elementy na úpravu orientačných bodov.

Pri **lietaní** pilot potrebuje mať dostupné dotykové ovládanie kamery pre prípad, že je na zariadení s dotykovou obrazovkou. Základné nastavenia, reštart pripojenia a takisto sa tu spúšťa misia. V prípade misie potrebuje mať zobrazený aktuálny cieľ, jeho popis a ostatné drony. Keďže je tento panel nevyužitý v prípade, že nebeží misia, vhodným využitím je zobraziť v ňom existujúce orientačné body s ich popisom, aby pilot mohol jednoducho začať navigáciu k nim a nemusel sa preklikávať cez niekoľko tlačidiel, ako tomu bolo doteraz. Panel



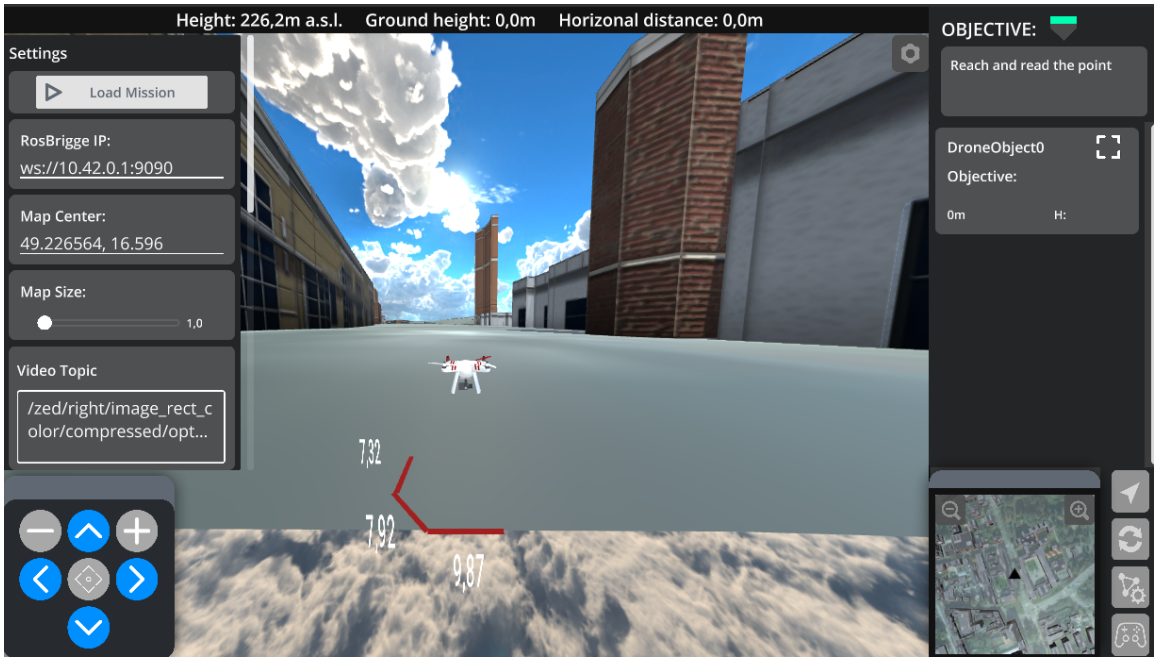
Obr. 3.4: Mockup navrhnutého rozhrania slúžiaceho na vytvorenie novej cesty v rámci misie. V pravom zozname sú zobrazené všetky vytvorené objekty. V ľavom sú objekty, ktoré budú vo vytváranej ceste.

s nastaveniami ide úplne skryť v prípade, že ho nepotrebujeme, pričom rovnaká možnosť by mala existovať aj pre panel s dronmi/bodmi. Pilot môže potrebovať **označiť aktuálnu pozíciu**. Bolo by zbytočné prechádzať, kvôli tomu do druhého rozhrania, takže by malo existovať tlačidlo na pridanie jednoduchého bodu na aktuálnu polohu. V hornej časti je čierna lišta zobrazujúca letové informácie a značná časť na stranách je nevyužitá, preto by som práve sem pridal prepínanie medzi rozhraniami.

Po prepnutí do **editačného rozhrania** pravdepodobne budeme chcieť odpútať kameru od dronu a voľne sa s ňou pohybovať. Pri pohybe kamery môže ísť o presuny na veľké vzdialenosti, ale aj iba o presun napr. k vedľajšej budove. Jedna rýchlosť pohybu by teda mohla zbytočne frustrovať užívateľa, ktorý by nevedel trafiť na žiadanú pozíciu kamery alebo by mu presun trval veľmi dlho. Pridanie posuvného slidera na ovládanie rýchlosti kamery vedľa tlačidiel na prepínanie módov kamery, by malo tento problém odstrániť. Správanie kamery by bolo ovplyvnené pomocou tlačidiel v hornej časti obrazovky. Mali by tak vzniknúť dve zoskupenia tlačidiel. Jedny na prácu s kamerou a druhé na pridávanie bodov.

Keďže existuje viacero druhov **orientačných bodov**, tak pre ich pridanie musíme vedieť, ktorý z nich chceme pridať. Prvým nápadom bolo vytvoriť rozbaľovacie menu s možnosťami, avšak ukázalo sa skôr ako nepraktické pri opakovanom vytváraní viacerých bodov. Vďaka odstráneniu všetkých nepotrebných panelov, máme na pridanie nových elementov takmer celú obrazovku. Prečo to teda nevyužiť a nevytvoriť pre každý typ samostatné tlačidlo. Nemyslím, že by bolo možné vytvoriť 6 rôznych tlačidiel, ktoré by jasne indikovali pridávaný typ orientačného bodu len pomocou ikony. Zvolil som teda tlačidlá s textovým popisom, čo za bod pridávajú.

Po vytvorení chceme objekty vidieť, prípadne ich upraviť. Užívateľ už pozná panel s dronmi a existujúcimi bodmi. Myslím, že body v tomto rozhraní by mohli byť zobra-



Obr. 3.5: Aktuálne UI rozhranie DroCa

zené v rovnakom paneli, avšak pomohol by nejaký filter, ktorý by filtroval body na základe typov. Po kliknutí na panel konkrétneho bodu, by boli zobrazené dostupné možnosti na prácu s ním. Ďalej je potrebné niekam umiestniť tlačidlá na načítavanie a ukladanie scenárov. Nepredpokladám, že budú využívané príliš často, a teda mohli by byť niekde skryté. Avšak, pre lepší vizuálny vzhľad som sa rozhodol umiestniť ich do ľavého horného rohu medzi skorej spomínané skupiny tlačidiel. Pri načítaní treba zobraziť všetky dostupné misie a dať tak užívateľovi na výber.

3.4 Počítačová a mobilná verzia

Predpokladom je, že pilot počas letu nebude mať možnosť sledovať obrazovku počítača a bude využívať maximálne jednu ruku na prácu na svojom telefóne, prípadne tablete. Pre jednoduché pridávanie bodov je potrebný pohyb kamerou po mape, avšak nepredpokladám, že by tento pohyb užívateľ zvládol jednou rukou na dotykovej obrazovke. Ani samotné pridávanie objektov, ktoré zobrazuje dialógové okná, nie je optimalizované na prácu bez klávesnice a myši. Takisto sa nepočíta s tým, že by pilot počas letu s dronom vytváral misiu. Pri týchto akciách sa očakáva práca na počítači ešte pred samotným letom. Z tohoto dôvodu je vhodné oddeliť užívateľské rozhrania tak aby pilot dostal všetky potrebné informácie, ale nebol zaťažený zbytočnými tlačidlami a oknami na editovanie orientačných bodov. Keďže užívateľské rozhranie je rozdelené na lietajúci a editačný mód, najjednoduchším riešením je jednoducho schovať tlačidlo pomocou, ktorého sa zapína editačný mód na mobilných zariadeniach. Týmto by sa zabránilo vstupu užívateľa do tohto režimu, ak by bol na zariadení s dotykovou obrazovkou.

Kapitola 4

Implementácia

Bolo potrebné implementovať mechanizmus na pridávanie orientačných bodov zľahčujúcich pilotovu orientáciu. Umožniť ich usporiadať do logických postupností, uložiť ich a následne aj načítať. Toto všetko by malo byť možné pomocou prehľadného užívateľského rozhrania. Proces pridávania bodov je mierne odlišný pre každý bod. Podstata je však rovnaká a teda problémy sa dajú vyriešiť pre všetky body súčasne.

4.1 Checkpointy

Existuje 6 rôznych orientačných bodov, ktoré slúžia na rozličné účely. Napriek tomu, je potrebné vytvoriť rovnaký postup pridávania pre všetky z nich. Súčasná aplikácia ponúkala isté možnosti pridávania jednoduchých bodov, avšak rozširovanie o nové typy nebolo možné. Preto bolo potrebné prísť s riešením, ktoré by do budúcna umožňovalo ľahko rozširovať orientačné body v DroCu o nové typy. Vytvoril som teda riešenie, ktoré pracuje pomocou niekoľkých prefabov, kde každý reprezentuje jeden typ objektu. Následne sú len inicializované a upravené pomocou zadaných parametrov. O inicializáciu a úpravu objektov sa stará **EditorController**.

SimpleCheckpoint a DetailedCheckpoint

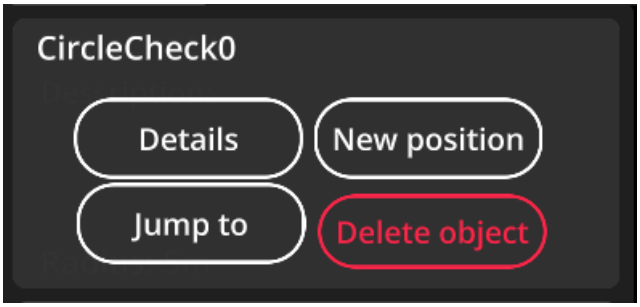
Tieto dva typy sú jediné zdieľajúce ten istý **prefab**. Prefab má ako komponenty skripty s triedami- **SimpleCheckpoint** a **DetailedCheckpoint**. V týchto triedach sú definované metódy na základnú manipuláciu s objektmi. Podotknem, že sú to takisto jediné dva objekty, ktoré majú rôzny prefab pre polohu na zemi a vo vzduchu.

Pri inicializácii objektu je nutné vytvoriť všetky prvky v užívateľskom rozhraní, ktoré potrebujeme pre zobrazenie objektu alebo prácu s ním. Na toto slúži *GeneratePointStruct*. V nej je ako prvé potrebné vytvoriť zväčšenú kópiu, ktorá bude zobrazená v **2D móde**. Následne bolo potrebné vyriešiť inicializáciu panelov na prácu s objektom a ich umiestnenie do zoznamov. Každý objekt (okrem kamery, ktorá sa nedá vložiť ako ciel v misii) potrebuje 6 panelov. Panely sa inicializujú z prefabov pripravených pre každú triedu zvlášť.

1. ObjectPanel - zobrazuje základné informácie o objekte a po kliknutí sa mení na OptionsPanel

2. optionsPanel(obrázok 4.1) - obsahuje tlačidlá na prácu s objektom a po kliknutí mimo tlačidlo sa mení na objectPanel

3. DetailsPanel - dajú sa v ňom upravovať informácie o objekte
4. missionPanel - využívaný pri tvorbe misie
5. navigationPanel - položka v zozname pre navigáciu
6. textPanel - zobrazený v **2D móde** objekt ponad objekt s jeho názvom



Obr. 4.1: OptionsPanel

K panelom, ako aj ich tlačidlám treba priradiť udalosti a tiež to čo sa pri nich má vykonať. Toto sa deje pomocou komponentu **Button** a jej metódy *onClick.AddListener(() => požadovaná funkcia)*.

Ďalej trieda obsahuje metódy na základnú manipuláciu so „sebou“, ako prepínanie panelov, zmena parametrov po zadaní užívateľom alebo aj zmazanie samého seba. Avšak, sú využívané aj metódy iných tried napr. na manipuláciu kamery.

CircleCheckpoint

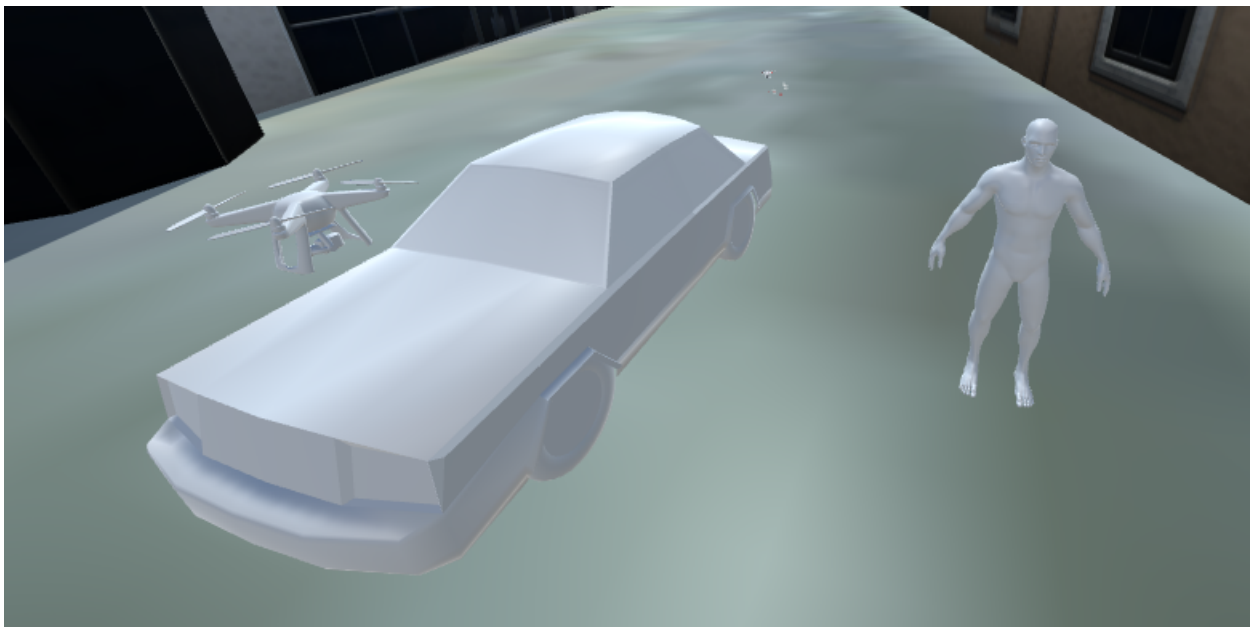
Tento checkpoint sa líši od ostatných v tom, že je potrebné vykresliť kruh s prázdnym priestorom a nie celistvý tvar. A teda využitie Mesh Rendereru, tak ako pri iných objektoch by zbytočne skomplikovalo vytváranie kruhov s určitým priemerom. Ako riešenie som zvolil komponent priamo z Unity Enginu a to **Line Renderer**. Po pridaní do **GameObject**-u je mu možné poslať pole dvoch alebo viacerých súradníc a on medzi nimi vykreslí čiaru. V mojom riešení pole pozostáva z ôsmich súradníc. Skúšal som väčší alebo menší počet, ale pri menšom vykreslený objekt nepripomínal kruh a pri väčších počtoch to síce vyzeralo vizuálne lepšie, avšak rozdiel nebol taký výrazný, aby sa to oplatilo v rámci optimalizácie výkonu. Toto pole so súradnicami sa vypočíta na začiatku metódy *GeneratePointStruct* a následne je nastavené pomocou *LineRenderer.SetPositions*, potom už je treba len upraviť parametre ako farba, hrúbka, ale aj *numCornerVertices* a *numCapVertices*, ktoré zaobľujú hrany a vytvárajú krajší, kruhu podobný tvar.

NPC

Non Playable Character je vo všeobecnosti postava/objekt, ktorý nie je ovládaný hráčom. Ide najmä o objekty, ktoré budú existovať v reálnom svete a budeme chcieť sledovať ich pohyb. Preto je, narozdiel od ostatných typov, nutné pravidelne aktualizovať aktuálnu polohu objektu, ktorá sa bude „samovoľne“ meniť. V **NPCObject** bola vytvorená metóda *Update*, ktorá sa volá pri výpočte každého snímku obrazu. V našom prípade sa reálna činnosť vykonáva len dvakrát za sekundu, kedy sa aplikácia opýta na nové dáta o objekte.

Dáta, kvôli nutnosti komunikácie so zariadením, nie je možné získať počas výpočtu jedného snímku obrazu. Preto *Update* spúšťa korutinu *GetData*. Korutina sa vykonáva v priebehu výpočtu viacerých snímkov, počas ktorých sa pomocou **UnityWebRequest** spýta serveru na aktuálnu polohu objektu a nastaví ju.

Tvar tohto objektu nie je dopredu známy. Preto bolo potrebné prísť s riešením, ktoré ho bude vedieť načítať a nastaviť až pri tvorbe objektu. Aplikácia poskytuje tri základné modely pre objekt - človek, auto, dron(obrázok 4.2). Ak chce užívateľ využiť svoj vlastný model, je mu to umožnené, vďaka voľne dostupnému rozšíreniu z Unity Asset Store - OBJ-Loader[1].Toto rozšírenie dokáže načítať obj a mtl formáty modelových súborov spolu s textúrami formátu BMP, TGA, JPG, PNG, DDS, a CRN.



Obr. 4.2: Tri základné modely pripravené na použitie pre užívateľa

4.2 Parametre objektu v misii

Vytvorené objekty a misie je potrebné po vytvorení aj uložiť. Zvolený bol JSON formát, ktorý je podporovaný aj zo strany Unity. Trieda **JSONUtility** z UnityEngine slúži na prácu s JSON-om v rámci Unity a obsahuje dve funkcie, ktoré nám slúžia na uloženie a načítanie či už objektov alebo misií. *FromJson* vytvára objekty z JSON-u (pri chýbajúcom atribúte dosadí základné hodnoty). *ToJson* konvertuje podporované verejné atribúty tried objektov na JSON. Tieto dve funkcie sú využité na uloženie a načítanie súborov.

Pre ľahkú prácu s objektmi je nutné mať všetky informácie o nich dostupné na jednom mieste. Z tohto dôvodu bola na zbieranie a ukladanie dát vytvorená trieda **EditorDataCollector**, ktorá spolupracuje s **EditorController**-om a uchováva všetky informácie o existujúcich objektoch. Uchovávané sú v triede **EditorObjectsCollection**.

Ako už vieme, Unity podporuje serializáciu do JSON-u.Avšak, na to aby sa štruktúra alebo trieda serializovala, je potrebné pridať označenie [Serializable], aby Unity vedelo, že

[1] https://assetstore.unity.com/packages/tools/modeling/runtime-obj-importer-49547

musí danú triedu/štruktúru zoserializovať. Tu vznikol problém, pretože typ GameObject nie je možné serializovať. Preto musia byť všetky dáta uchovávané v samostatných triedach bez odkazov na GameObjecty. Existuje teda GameObject reprezentujúci SimpleCheckpoint. Tento objekt má ako komponent triedu *SimpleCheckpoint*, ktorá obsahuje ukazovatele na všetky pomocné objekty typu GameObject, na prácu s checkpointom (bočný panel, panel pri tvorbe misie,...), metódy na prácu s objektom a inštanciu triedy SimpleCheckpointData. Táto trieda obsahuje všetky potrebné dáta pre opätovné vytvorenie objektu po uložení. Serializovateľná je len samotná trieda **SimpleCheckpointData**. Štruktúru možno vidieť v nasledujúcej ukážke.

```
public class SimpleCheckpointData
{
    public Color color;
    public string name;
    public Vector3 position;

    public SimpleCheckpointData(Color _color, string _name, Vector3 _position)
    {...}

}
public class SimpleCheckpoint : MonoBehaviour
{
    public SimpleCheckpointData data;

    public GameObject pointObject;
    public GameObject pointObjectBig;
    public GameObject OptionsPanel;
    public GameObject ObjectPanel;
    public GameObject DetailsPanel;
    public GameObject missionPanel;
    public GameObject navigationPanel;
    public GameObject textPanel;

    public Methods
    .
    .
    .
}
```

Pre uloženie je potrebné tieto dáta nejako zozbierať a vytvoriť z nich jednu celistvú štruktúru, ktorá sa uloží do súboru. Aj toto je úlohou **EditorDataCollector**-a, ktorý pri ukladaní prejde všetky zoznamy objektov na scéne, vloží z nich dáta do svojho wrappera, obsahujúceho listy dát, pridá zoznam dronov a zoznam jednotlivých ciest v misii pre dané drony(ak existujú). Následne všetko spolu serializuje pomocou *JsonUtility.ToJson()* a uloží do užívateľom pomenovaného súboru. Ak súbor existuje, tak je prepísaný.

Jeden bod môže byť počas misie využitý na **viacero účelov**, čo znamená, že by sa mohol vo výslednej štruktúre, vyskytovať opakovane. Bolo teda potrebné nájsť spôsob, ktorý by umožnil opakovaný výskyt objektu s tým, že pri načítaní by bol inicializovaný iba jedenkrát. Preto som sa rozhodol, rozdeliť ukladanie na dve časti. Ako prvé sa uložia objekty so svojimi

parametrami do svojich zoznamov a následne pri ukladaní misie sa využíva už len názov a typ checkpointu, podľa ktorého vieme presne určiť o ktorý objekt ide. Toto rozdelenie do dvoch fáz, takisto umožňuje ukladať misiu aj obyčajný scenár do súboru s rovnakým formátom. Kedy sa pri ukladaní scenára jednoducho vynechá druhá fáza.

Celková štruktúra ukladaného súboru sa skladá zo zoznamov pre každý typ bodu, zoznamu rôznych ciest v misii a zoznamu dronov. RouteList nie je povinná položka a existuje len, ak sa misia ukladá ako misia a nie ako scéna s objektmi. Obsahuje zoznam ciest pre jednotlivé drony, avšak body v nej, ako už bolo spomenuté, sú definované len dvojicou meno-typ. Objekt v ceste má nasledovné parametre:

- **name** - názov objektu
- **description** - popis objektu pre danú cestu, odlišuje sa od popisu uloženého mimo cesty
- **checkpointType** - určuje typ objektu (jednoduchý, NPC, zóna..)
- **type** - určuje ako je daný objekt využitý (regular, confirm, waitAll,...)
- **area** - oblasť,v ktorej ak sa dron nachádza, tak sa cieľ považuje za splnený
- **drones** - je zoznam dronov, ktoré majú bod ako svoj cieľ

4.3 Vytváranie misie

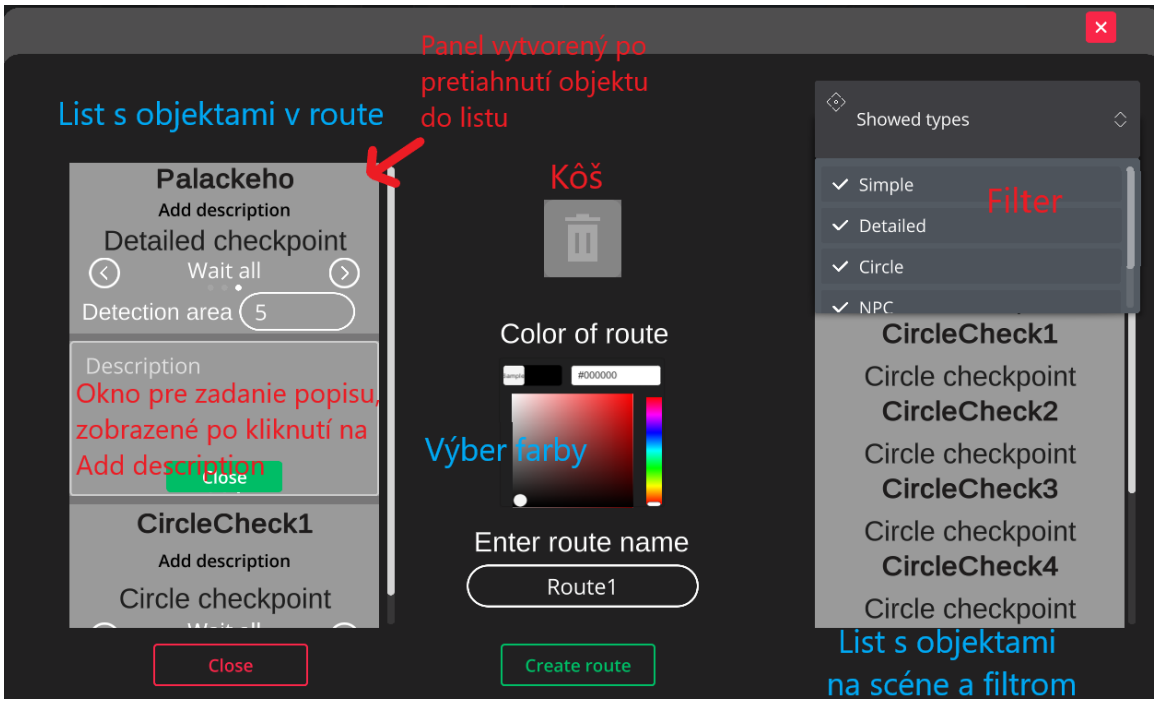
Aby sme mali čo uložiť, musíme najprv misiu vytvoriť. Vytváranie misie sa dá taktiež rozdeliť na dve časti. V prvej časti ide o samotné vytváranie objektov v priestore. Tu som nepracoval úplne od základov, keďže samotné pridávanie objektu pomocou kliknutia už existovalo. Avšak pridať objekt s parametrami, ktoré chceme zadať, je podstatne zložitejšie a vyžadovalo si značne rozšíriť existujúcu funkcionalitu. Najväčším problémom bolo vytvoriť jednotný systém pre všetky typy bodov a ich následnú úpravu. Proces pridávania objektu majú preto na starosti až tri triedy. **MapClickController**, **GuiController** a **EditorController**. Pre vytvorenie správneho typu objektu, na správnom mieste, bolo potrebné získať polohu kliknutého bodu z **MapClickController**-u, získať od užívateľa informácie o objekte (názov,popis,...) a zaslať ich **EditorController**-u, aby objekt vytvoril. Postup pridania teda vyzerá nasledovne:

1. Používateľ stačí tlačidlo pre pridanie objektu, **GuiController** prepne **MapClickController** do jedného z režimov na pridávanie objektu a zároveň si zapamätá aký typ objektu ide pridáva.
2. **MapClickController** pošle súradnice kliknutého miesta (technické detaily tohto procesu popísal vo svojej práci pán Sedlmajster[15])
3. **GuiController** zobrazí grafické okno pre používateľa, aby zadal ostatné atribúty objektu a po potvrdení sa údaje odošlú **EditorController**-u
4. **EditorController** vytvorí daný objekt

Zóna nie je len jeden bod ako všetky ostatné typy, preto nie je možné použiť rovnaký postup a vyžaduje si samostatný proces vytvárania. Trieda s touto funkcionalitou už existovala. **ZoneController** mal na starosti pridávanie a vykresľovanie zón. Problémom, však

bolo to , že dokázal pracovať len s jednou zónou na scéne. Preto ho bolo potrebné upraviť a vylepšiť na prácu s viacerými zónami súčasne. Jadro problému je v tom, že **MapClickController** posiela dáta o kliknutí priamo **ZoneController**-u a teda nemôže existovať pre každú zónu zvlášť. Možnosťou by bolo posielať dáta napr. **EditorController**-u, ktorý by si vytváral nové **ZoneController**-y. Toto mi však neprišlo ako rozumné riešenie, a to z dôvodu zbytočného presúvania zodpovedností cez viacero modulov a následná nutnosť hľadania správnej zóny pri jej úprave.
Problém bol vyriešený tak, že **ZoneController** pracuje s viacerými zónami pomocou slovníkov, kde kľúčom k uloženej zóne je jej názov. Vždy však pracuje len s jednou zónou súčasne a to na základe aktuálne aktívneho názvu zóny uloženého v *actKey*. Po tom ako užívateľ zapne mód pridávania zóny, nastaví sa *actKey* na „unfinishedZoneName“, ktoré sa využíva pre nedokončenú zónu. Po ukončení pridávania hrán sa štruktúry a objekty vytvárajúce zónu duplikujú a vložia do slovníka s novým názvom (získaným od užívateľa). Staré objekty patriace nedokončenej zóne sa zmažú. Editácia zón funguje na rovnakom princípe akurát pri zapnutí editovania zóny sa nastaví *actKey* na názov editovanej zóny.
Zóna je definovaná ako postupnosť bodov, medzi ktorými sú vykreslené hrany tvorené z čiar pomocou Unity komponentu **LineRenderer**. Zóna sa vždy vykresľovala s prednastavenou výškou bez možnosti zmeny. Preto som pridal možnosť nastaviť výšku stien pri vytvorení zóny. Funkcia *ChangeHeightOfWall()*, ktorá je volaná pri tvorbe pomenovanej zóny a prekreslí steny zóny aby odpovedali zadanej výške.

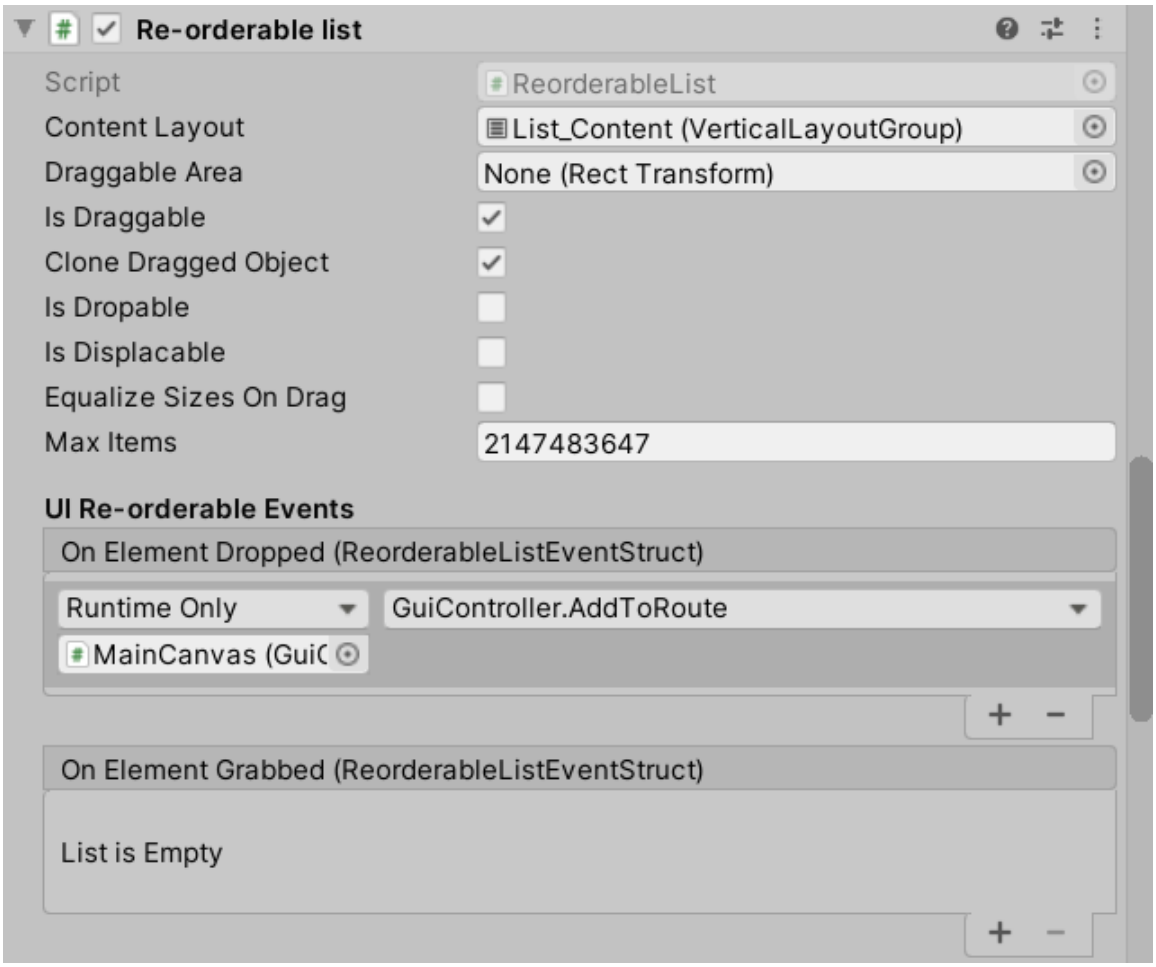
Vytváranie ciest pre drony



Obr. 4.3: Okno, kde sa vytvára cesta pre dron

Na vytvorenie misie pred jej uložením, je potrebné zoradiť objekty, teda ciele, do logickej postupnosti. Bol zvolený „drag&drop“ systém, ktorý bolo treba implementovať. Základné listy Unity neumožňujú presun ich elementov. Preto bolo treba nájsť iné riešenie. Rozšírenie

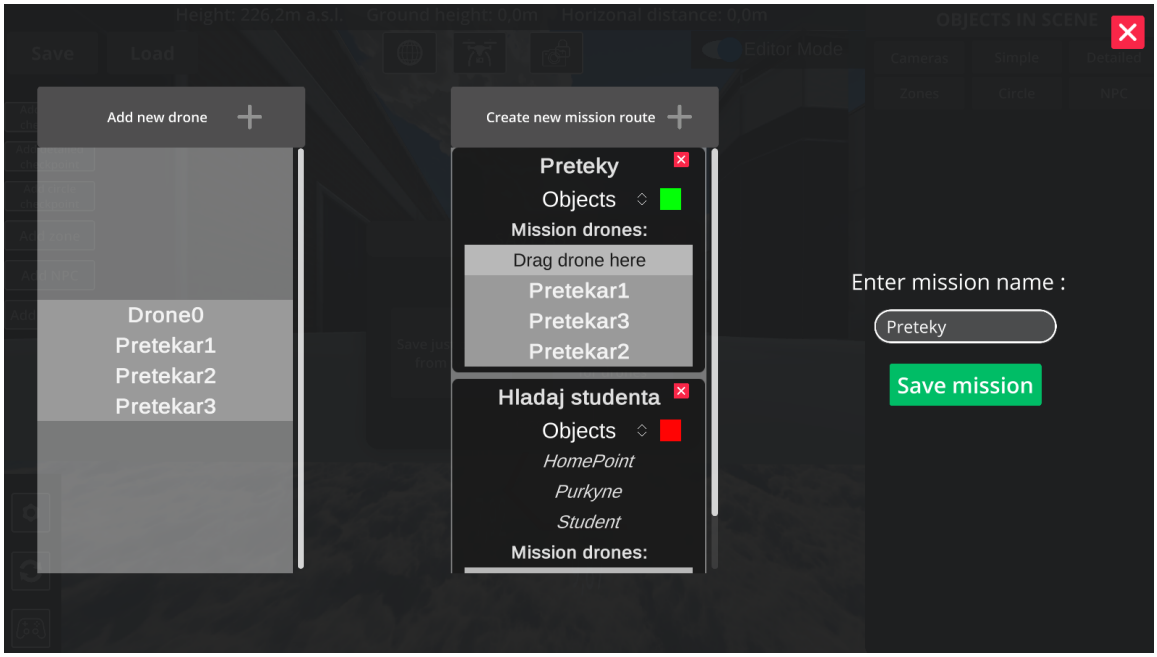
unity-ui-extensions[2] bohate rozširuje možnosti základných listov z Unity a takisto pridáva možnosť voľne presúvať objekty v rámci, ale aj mimo listov a meniť ich poradie. Umožňuje povoliť či zakázať presun alebo vkladanie elementov. Je možné zvoliť si, či bude element pri presune zduplikovaný alebo nie. Kombináciou týchto možností je vytvorený celý systém na tvorbu misie. Rozšírenie tiež implementuje možnosť volať vlastné funkcie pri rôznych akciách s elementom, ako napríklad pri presune, chytení alebo vložení. Možnosti, ktoré toto rozšírenie prináša sú zobrazené na obrázku 4.4.



Obr. 4.4: Možnosti ktoré rozšírenie prináša

Aby misia bola misiou, musí obsahovať drony, pre ktoré je určená. Bolo vytvorené nové grafické rozhranie (obrázok 4.5, poskytujúce túto funkcionalitu. Lavý list obsahuje drony dostupné v misii. Pomocou tlačidla vo vrchnej časti tohto listu je možné pridať nové drony. Pri pridávaní dronu je možné upraviť jeho názov, url adresu, odkiaľ sa získavajú dáta s aktuálnou polohou a súradnice na ktorých sa dron na začiatku inicializuje. Súradnice sú predvyplnené s aktuálnou pozíciou prvého (nášho) dronu. Tieto pridané drony nespadajú pod NPC objekty, ale stará sa o ne samostatná trieda *drones*.
V druhom liste sa nachádzajú body zoradené v logických postupnostiach - cestách. Na začiatku je prázdny. Bolo nutné vytvoriť mechanizmus, ktorý by umožnil tvorbu ciest z existujúcich orientačných bodov pomocou drag&drop systému.

[2]Rozšírenie dostupné z https://bitbucket.org/UnityUIExtensions/unity-ui-extensions/src/release/



Obr. 4.5: Obrazovka na vytvorenie misie

Pri vytváraní cesty sa opäť zobrazí nové okno (obrázok 4.3) s dvomi listami. V pravom sú zobrazené všetky objekty vytvorené na scéne, konkrétne ich názov a typ. Objekty je možné filtrovať pomocou filtra umiestneného nad zoznamom. Filter je vyriešený pomocou *Dropdown Multi Select* zoznamu z **Modern UI Pack**. Za filtrovanie je zodpovedný **GuiController**, konkrétne funkcia *MissionFilter*. Tá po kliknutí v zozname, buď zobrazí alebo skryje objekty daného typu.

Pre vloženie objektu do cesty ho stačí jednoducho „pretiahnuť“ z listu objektov do listu reprezentujúceho vytváranú cestu. Po pridaní do cesty, však potrebujeme aj iné informácie ako názov a typ, preto sa musí vytvoriť nový panel, reprezentujúci presunutý objekt. Na to sa využíva funkcionalita rozšírenia **Unity UI Extension**, kde po vložení elementu do zoznamu aktuálnej cesty, vznikne udalosť „On Element Dropped“, ktorá zavolá funkciu *AddToRoute* z **GuiController**-u. Tá vytvára sa nový panel, ktorý poskytuje priestor pre vloženie informácií o danom objekte v danej ceste. Je možno vložiť popis, prípadne poznámku, ktorá sa zobrazí, keď bude daný objekt aktuálnym cieľom. Takisto sa dá nastaviť o aký cieľ ide (confirm, regular, atď.) a veľkosť detekčnej zóny, tak ako už bolo spomenuté pri ukladaných štruktúrach (obrázok 4.2). Vybratie typu cieľa je implementované pomocou *Horizontal Selectora*, taktiež z **Modern UI Pack**.

Počítame s tým. že užívateľ sa môže pri vytváraní pomýliť a bude chcieť daný bod z cesty odstrániť. Pridal som teda plochu s logom smetného koša, ktorá slúži na odstránenie už vloženého objektu z cesty a je implementovaná ako list, ktorý každý element, čo je do neho vložený, zmaže.

Cesty je treba nejakým spôsobom od seba odlíšiť. Najefektívnejším a najintuitívnejším spôsobom je výberom farby pre každú cestu. Preto je pri vytváraní cesty možné nastaviť jej farbu pomocou „color pickeru“, ktorý bol prebraný od open-source projektu z githubu[3].

[3] https://github.com/judah4/HSV-Color-Picker-Unity

Cestu môžeme pomenovať alebo jej ponechať vygenerované meno: route a počet vytvorených ciest plus jedna.

Po kliknutí na „Create route“ funkcia *CreateRoute* z **GuiControlleru** spracuje list. Keďže samotná cesta zatiaľ existuje len ako poradie elementov v liste s informáciami o objektoch, tak musíme pre ďalšie využitie, dané informácie odtiaľ získať. Vieme, v ktorom objekte užívateľského rozhrania sa tieto elementy nachádzajú, preto môžeme cyklicky prejsť cez jeho „deti“ v hierarchii Unity. Získané informácie sa uložia do inštancie triedy *RouteObject* a element sa odstráni z listu. Inštanciu tejto triedy je potom ešte nutné vložiť do *routeObjectList*. Po vložení všetkých cieľových bodov je možné vytvoriť danú cestu. Okrem samotnej cesty je treba ešte vytvoriť panel reprezentujúci danú cestu. Na toto slúži *CreateRoute* z **EditorController**-a . Tá ako parametre príma spomínaný list, názov, farbu a list dronov v ceste (zatiaľ prázdny). V *CreateRoute* najprv vzniká spomínaný panel reprezentujúci cestu, zobrazený v liste ciest (obrázok 3.3). Tento panel obsahuje názov, farbu, rozbaľovací zoznam cieľov(iba ich názvy), priestor na pretiahnutie, a teda vloženie dronu do danej cesty. Po paneli sa vytvára inštancia triedy *MissionRoute* a *MissionRouteData*, ktorá sa už ukladá do pamäti **DataCollectoru**.

Po vytvorení cesty, je potrebné aby bola zobrazená aj inak, ako v podobe jedného panelu. Toto som vyriešil vykresľovaním čiar medzi jednotlivými bodmi cesty. Stará sa o to **RouteDrawer**, ktorý na vykresľovanie využíva **LineRenderer**. Problémom je, ak je daný objekt presunutý po pridaní do cesty. A teda čiara by ukazovala k nesprávnemu miestu. Z tohto dôvodu existuje *Update* funkcia, ktorá sleduje či sa niektorý bod nepohol. Tu som využil atribút z komponenty transform. Do *hasChanged* sa pri zmene polohy nastaví true a ostáva true, až kým sa táto hodnota nezmení v niektorej funkcii. Keďže bod môže byť vo viacerých cestách, musí byť *hasChanged* zmenené na false až po skončení *Update* vo všetkých cestách. Toto je docielené zavolaním korutiny, ktorá po skončení aktuálneho framu nastaví hodnotu na false.

4.4 Užívateľské rozhranie

Bolo potrebné upraviť existujúce rozhranie aby zobrazovalo nové dostupné informácie. Okrem toho bolo potrebné vytvoriť nové editačné rozhranie na pridávanie a prácu s orientačnými bodmi.

Pôvodné rozhranie

Toto pôvodné, len rozšírené rozhranie sa skladá zo štyroch hlavných častí.

- Menu nastavení - zmena ROS adresy, súradníc mapy, veľkosti mapy...Táto časť ostáva nezmenená.
- Tlačidlá na ovládanie kamery - slúžia na otáčanie/približovanie kamery pomocou triedy **CameraController**. Zväčša nezmenená, bola pridaná len možnosť presúvať panel s nimi voľne po obrazovke.
- Minimapa - implementovaná pomocou **MinimapScript** a ďalšej kamery, taktiež pridaná len možnosť voľne presúvať po obrazovke
- Informačný panel - slúži na získavanie informácií o aktuálnej misii a vzdialenosti ostatných dronov. Zaberá jednu pätinu obrazovky, keďže je však tento panel zväčša nevyužitý, bolo pridané tlačidlo na minimalizáciu/maximalizáciu okna. Pri minimalizácii

zostáva zobrazená len horná lišta. Zároveň som odstránil triedu **cameraScaler**, ktorá zmenšovala obraz kamery len na štyri pätiny obrazovky, a to práve kvôli oknu s informáciami. Toto zmenšenie obrazu kamery spôsobilo, že pri pohybe okien mimo daného zmenšeného priestoru ostávali objekty zaseknuté (kamera sa v danej časti obrazovky vôbec neobnovovala). Takisto bolo na spodnú časť okna umiestnené prepínanie medzi informačným a navigačným oknom. Navigačné okno zobrazuje skrolovateľný zoznam všetkých objektov, ktoré aktuálne existujú. Každá položka obsahuje názov, popis a typ objektu. Po kliknutí sa zobrazí možnosť zapnúť si navigáciu k objektu alebo ju ukončiť. O navigáciu sa stará **NavigationController**, ktorý už taktiež existoval a bol len upravený, aby lepšie odpovedal využitiu. Podotýkam, že ide o inú triedu ako tá, ktorá sa stará o označovanie cieľov v rámci misie. Je to najmä z toho dôvodu, že pilot niekedy potrebuje aj počas misie odletieť k inému predtým neznámemu cieľu, a práve preto je to oddelené do dvoch samostatných tried.

Editačné rozhranie

Pozostáva z troch hlavných častí. Snažil som sa zachovať rozmiestnenie sebe podobných prvkov rovnaké, aby bol dojem zmeny pri prepnutí čo najmenší.

Zoskupenia tlačidiel

Táto sekcia sa nachádza v ľavej hornej časti a dá sa rozčleniť na horizontálnu a vertikálnu časť. Horizontálna časť uľahčuje prácu s kamerou. Bolo potrebné implenentovať ovládanie kamery a prepínanie medzi viacerými kamerami. Toto je implementované pomocou tried **CameraController** a **ChangeCamera**, tie umožňujú prepínať medzi kamerami alebo s nimi pohybovať. Pohyb je podobný ako pri pohybe napr. na Google Maps. A teda ťahavým pohybom myši so stlačeným ľavým tlačidlom. Pri pohybe k sebe sa kamera v priestore posúva dopredu. Pri tomto pohybe by bolo veľmi ťažké určiť či sa chce užívateľ priblížiť k objektu alebo posunúť vpred. Preto tento pohyb neumožňuje pohyb po osi Y, čo znamená, že sa pohybujeme v stále tej istej výške. Na približovanie a vzďaľovanie bolo teda potrebné nájsť iné riešenie. Keďže sa pohybujeme pomocou myši, vhodným riešením pre priblíženie/-vzdialenie bolo využitie kolečka myši. Tento pohyb je počítaný pomocou „forward vektora“, ktorý označuje vektor kam objekt (kamera) smeruje,a teda priblíženie a vzďaľovanie sa deje na rovnobežke s daným vektorom.

Kamera je v hierarchii dieťa objektu dronu, preto sleduje jeho pohyb. Na to aby sa kamera mohla voľne pohybovať v priestore musíme túto väzbu odstrániť. Po aktivácii voľného pohybu si GameObject kamery nastaví svojho rodiča v Unity hierarchii na *null*, a tým pádom sa môže pohybovať nezávisle od drona. Pri ukončení módu sa len znovu nastaví ako rodič GameObject dronu a kamera sa presunie na jeho súradnice a začne sledovať jeho pohyb. Rýchlosť pohybu kamery sa dá meniť pomocou posuvného slideru a triedy **CameraController**. Tento panel je viditeľný len ak je aktivovaný mód voľného pohybu kamery a nachádza sa v hornej časti obrazovky vedľa spomínaných tlačidiel. Ďalej je možné zapnúť 2D mód, ktorý poskytuje pohľad zhora. Pri jeho zapnutí sa zobrazí aj možnosť vypnúť si zobrazenie budov, a teda prepnúť na čistú 2D verziu mapy. Pôvodné objekty sú príliš malé aby boli viditeľné z väčšej výšky. Pre tento mód sú vytvorené zväčšené kópie objektov aj s ich názvami, ktoré sú viditeľné len počas neho.

Tlačidlá vo vertikálnej časti slúžia na pridávanie objektov. Po kliknutí na niektoré z nich sa spúšťa proces pridania objektu daného typu. Kliknutie vyšle signál do **GuiControlleru**, ktorý následne komunikuje s ostatnými triedami a buď daný objekt priamo pridá,

začne proces jeho pridávania zobrazením okna pre zadanie informácii alebo pošle správu ďalej **MapClickControlleru**, ktorý následne pošle súradnice kliknutia naspäť do **GuiControllera**.

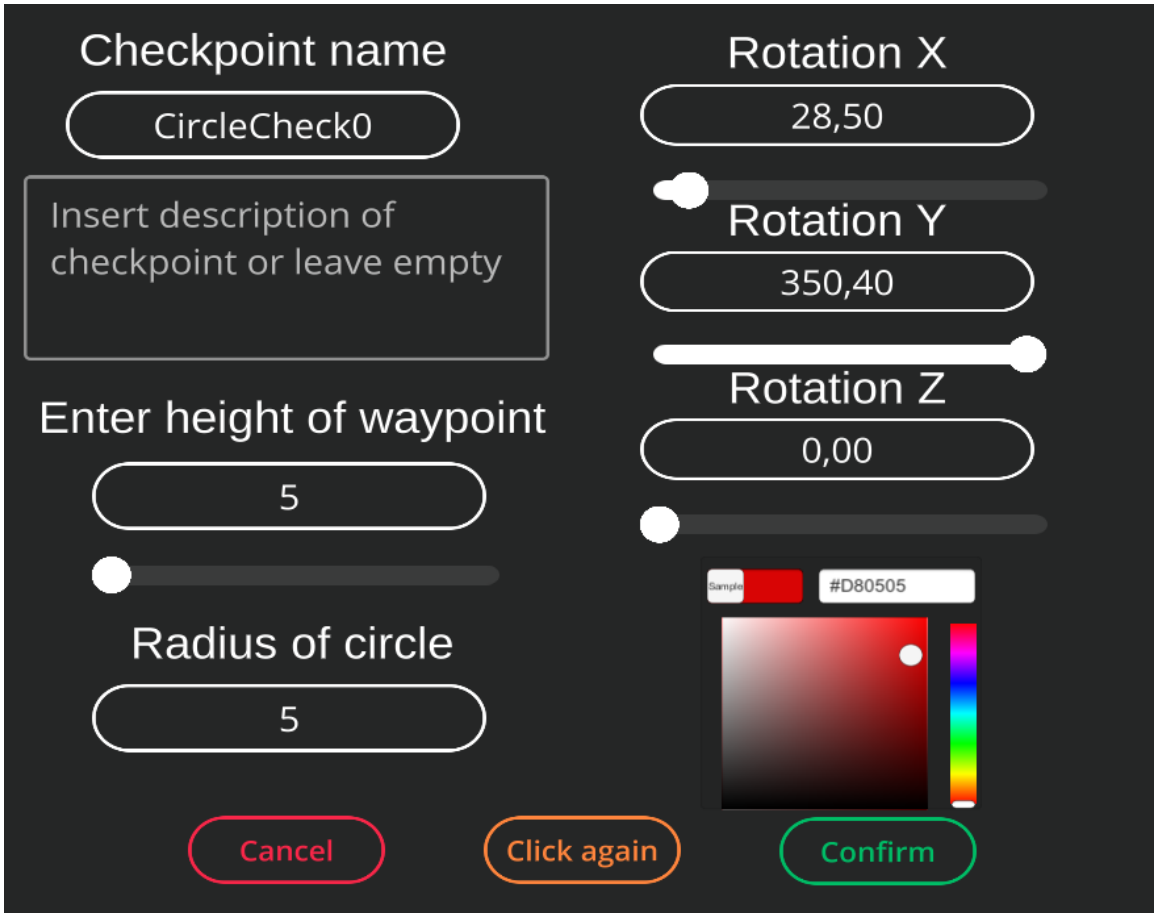
Okná na zadanie informácií o objekte

Objekty majú rôzne špecifické informácie, ktoré užívateľ môže chcieť upraviť alebo zadať. Tieto parametre sa líšia od objektu k objektu, avšak riešenie pre ich zadávanie je takmer rovnaké vo všetkých bodoch. Preto opíšem detailnejšie len jedno okno. Kruhový cieľový bod obsahuje najviac parametrov, ktoré môžu byť užívateľom zmenené, preto použijem tento objekt ako ukážkový. Okno (obrázok 4.6) obsahuje textové polia (typ *TMP_InputField*) pre úpravu názvu, ktorý je predvyplnený, pridanie popisu a textové polia s výškou, priemerom a rotáciou na osiach x, y, z. Pod textovým poľom s výškou je aj posuvný slider, ktorý slúži na jednoduchšiu úpravu výšky. Podobné slidery sú aj rotácii. To dovoľuje zadať číselné parametre bez nutnosti chytenia klávesnice. Pohyb so sliderom vyvolá funkciu z **GuiController**-u *SliderHandler*, ktorá nastaví aktuálnu hodnotu slideru do poľa. Existuje samozrejme aj funkcia pre opačný prípad, kedy je zmenená hodnota ručne a nie pomocou slideru. Tu sa pomocou *SetValueWithoutNotify()* posunie slider na zadanú hodnotu. Pri zadávaní rotácie máme prednastavené hodnoty totožné s rotáciou kamery v momente kliknutia, a teda vyvolania tohto okna. Posledným prvkom je ,už skorej spomínaný, color picker. Bol mierne upravený, teda lepšie povedané minimalizovaný, keďže pre naše potreby úplne stačí farebný štvorec, kde sa pomocou kliknutia vyberie farba.

Panel objektov

Je veľmi pravdepodobné, že vytvorené objekty budeme chcieť nielen vidieť, ale aj upraviť. Presne na toto slúži tento panel, ktorý pripomína panel informácií z druhého rozhrania. Pozostáva zo skupiny tlačidiel nachádzajúcej sa v hornej časti. Tlačidlá slúžia na prepínanie práve zobrazeného zoznamu objektov podľa ich typu. Je tam teda 6 tlačidiel pre jednotlivé typy - SimpleCheckpoint, DetailedCheckpoint, Camera, CircleCheckpoint, NPC a Zone. Pod nimi sú zobrazené objekty z práve aktuálne zvoleného typu. Pre každý objekt existuje panel s jeho názvom a popisom, pri niektorých typoch je to ešte rozšírené o výšku alebo priemer.

Po kliknutí na panel, sa zobrazia možnosti práce s objektom. Ide o vycentrovanie kamery na objekt, zmazanie objektu, nová poloha pomocou kliknutia a manuálna úprava parametrov. Napríklad pri objekte typu kamery sa môže zdať logické, že aktuálnou kamerou sa stane tá, ktorú chceme zobraziť. Avšak nie je to pravda. Nejde o novú kameru, ale len o uložené súradnice, rovnako ako pri ostatných typoch objektov, na ktoré sa kamera po kliknutí premiestni. Presnejšie povedané presunie sa niekoľko metrov pred objekt pomocou odčítania vektoru získaného z atribútu Unity objektu *transform.forward* vynásobeného piatimi. Pri kliknutí na novú polohu, posiela **EditorController**, GameObject kliknutého objektu **GuiControlleru**, ktorý potom získané dáta z kliknutia nastaví ako novú polohu objektu. Pri ručnej zmene dát sa zobrazí užívateľovi nový väčší panel s poliami na vkladanie textu, ktoré obsahujú aktuálne hodnoty. Unity používa na umiestnenie objektov do scény 3D súradnicový systém s osami x,y,z, avšak užívateľ nevie ako sa daná poloha prepočítava do reálneho sveta. Hodnoty teda musíme previesť na geografickú šírku a dĺžku. Našťastie nám toto veľmi jednoducho umožňuje trieda **AbstractMap** obsahujúca funkcie *WorldToGeoPosition* a *GeoToWorldPosition*. Funkcia *GeoToWorldPosition* však vracia hodnotu na osi y ako nadmorskú výšku daného bodu na zemskom povrchu. Z tohto dôvodu je po-



Obr. 4.6: Dialógové okno zobrazujúce dostupné možnosti pri pridávaní kruhového checkpointu

trebné pripočítavať resp. odčítavať užívateľom zadanú hodnotu výšky objektu pri každom prepočte.

4.5 Testovanie a budúce možnosti rozšírenia aplikácie

Hlavnou náplňou DroCa je uľahčiť pilotovi orientáciu v teréne, avšak pridané rozšírenia budú používané najmä počas prípravy na let. A teda, na overenie funkčnosti aplikácie, nie je potrebné priamo lietať s reálnym dronom a ku všetkým činnostiam nám stačí len ten simulovaný.

Vo väčšej miere prebiehalo testovanie mnou. Všetky technické časti testovania boli robené mnou, priamo počas implementácie kódu a pri užívateľskom testovaní bolo cieľom upraviť vzhľad DroCa, tak aby bol intuitívny a jednoduchý na používanie.

Užívateľské testovanie bolo navrhnuté, tak aby postupne zoznámilo testera s užívateľským prostredím spolu s jeho možnosťami pridávania nových bodov alebo vytvárania misie. Zvolil som teda testovanie pomocou troch dopredu pripravených scenárov. Pred samotnými scenármi bolo vysvetlené, na čo DroCo slúži a čo všetko dokáže. V prvom scenári bolo sledované, či sa nový užívateľ v aplikácii dokáže rýchlo zorientovať a prepnúť rozhranie, aby si pridal nejaký orientačný bod. Toto testovanie ukázalo nedostatky v skorších verziách užíva-

teľského rozhrania, ktoré boli následne odstránené. V druhom kroku sa sledovalo, či tester chápe rozdiel medzi misiou a scenárom len s bodmi. Následne mal vytvoriť jednoduchú misiu. Pri poslednom testovacom scenári bola pripravená misia s chybami. Bol vytvorený zoznam chýb nachádzajúcich sa v nej, ktoré mal tester opraviť a následne misiu znovu uložiť. Scenár bol zameraný tak, aby umožnil využitie každej pridanej funkcionality. Napr. bolo potrebné nastaviť zónu, aby bola v tvare štvorcu, teda predpokladalo sa, že tester sa prepne do 2D módu a upraví ju. K dispozícii boli uložené kamery a bolo sledované, či sa na ne prepne alebo sa bude posúvať manuálne a teda nevyužije tieto zjednodušenia práce, čo by mohlo indikovať, že sú zle umiestnené. Výsledky druhého a tretieho testovacieho scenára boli uspokojivé. Tester síce zo začiatku úplne nechápal ako fungujú módy kamery a jej prepínanie, avšak akonáhle tento proces pochopil dokázal využiť všetky jeho možnosti naplno.

Možnosti na zlepšenia

Momentálny stav aplikácie určite nie je najdokonalejší a dá sa v mnohých ohľadoch zlepšiť. Jedným z hlavných nedostatkov, ktorý by bolo treba odstrániť je načítavanie misie. Mojim cieľom v tejto práci bolo pridanie editačného rozhrania a umožniť tak užívateľom pridávať orientačné body a zoraďovať ich do misií. Dovolím si tvrdiť, že toto som splnil úspešne. Takisto som pridal možnosť body načítať mimo misie, načítať a spustiť základné misie. Avšak, pokročilejšie misie z hľadiska rôznych druhov cieľových bodov alebo spracovanie dronov, ktoré sú vo viacerých cestách, v rámci misie, nie je plne funkčné. Je potrebné doplniť aj funkčnosť niektorých rozširujúcich bodov ako je NPC alebo Circle Checkpoint, ktoré sa zatiaľ tvária rovnako základné body. Takisto je podľa môjho názoru vhodné pridať možnosť vytvoriť misiu aj jednoduchším spôsobom ako terajším. Napr. začať vytváranie misie, ktoré bude sledovať pohyb dronu a následne len ukladať cieľové body do danej štruktúry.

Kapitola 5

Záver

Cieľom tejto práce bolo navrhnúť a implementovať systém na vkladanie orientačných bodov a ich spájanie do misie do už existujúcej aplikácie na uľahčenie pilotovania dronov.

Začal som štúdiom problematiky pilotovania dronov, ich využitia v rôznych oblastiach a nedostatkov existujúcich riešení. Následne som sa oboznámil s architektúrou existujúcej aplikácie. Svoje poznatky som analyzoval, aby som neskôr z nich vytvoril zoznam vylepšení, ktoré aplikácia potrebuje.

Po identifikácii problémov som navrhol ich riešenia. Jednalo sa najmä o užívateľské rozhrania pre prácu s novými objektmi. Je potrebné aby boli tieto rozhrania prehľadné a rýchle na používanie. Preto som počas vývoja niekoľko krát upravil návrh a rozmiestnenie prvkov. Mojím hlavným cieľom bolo odbremeniť užívateľa od repetetívneho klikania. Vytvoril som niekoľko orientačných bodov, ktoré majú uľahčiť navigáciu pilota v akejkoľvek situácii. Vytvorené bolo aj rozhranie, ktoré uľahčuje pridávanie týchto bodov. Takisto sú pridané možnosti na následnú úpravu týchto bodov. Niektoré scenáre sa môžu odohrávať na veľkých plochách a z tohto dôvodu bolo pridané rýchle prepínanie medzi uloženými kamerami. Medzi bodmi môže, ale aj nemusí existovať prepojenie. Preto som vytvoril mechanizmus, ktorý dovoľuje uložiť a načítať orientačné body. Body môžu byť spojené do misie, ktorú je možné spustiť a odohrať. V závere svojej práce som prepracoval načítavanie misie tak, aby bolo možné načítať a spustiť novú zložitejšiu štruktúru misie.

Ako ďalšie kroky aplikácie vidím dokončenie chýbajúcich funkcionalít pri hraní misie, ktoré by kontrolovali špeciálne podmienky splnenia určitých bodov. Zároveň by bolo vhodné smerovať aplikáciu k podpore multiplayeru, kde by sa dokázalo do misie napojiť viacero dronov.

Myslím, že aplikácia má veľký potenciál do budúcna a verím, že ďalší vývoj prinesie riešenie spomenutých problémov a umožní reálne využitie aplikácie v praxi.

Literatúra

[1] *UNITY documentation.* [cit. 2022-4-7]. Dostupné z: https://docs.unity3d.com/Manual/.

[2] *Easy access rules for Unmanned Aircraft Systems (regulation (EU) 2019/947 and regulation (EU) 2019/945).* Sep 2021 [cit. 2022-4-7]. Dostupné z: https://www.easa.europa.eu/document-library/easy-access-rules/easy-access-rules-unmanned-aircraft-systems-regulation-eu#group-publications.

[3] Agrawal, A., Abraham, S. J., Burger, B., Christine, C., Fraser, L. et al. The Next Generation of Human-Drone Partnerships: Co-Designing an Emergency Response System. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems.* New York, NY, USA: Association for Computing Machinery, 2020, s. 1–13. ISBN 9781450367080. Dostupné z: https://doi.org/10.1145/3313831.3376825.

[4] Ahirwar, S., Swarnkar, R., Bhukya, S. a Namwade, G. Application of drone in agriculture. *International Journal of Current Microbiology and Applied Sciences.* 2019, zv. 8, č. 01, s. 2500–2505.

[5] Cleland Huang, J. a Agrawal, A. *Human-Drone Interactions with Semi-Autonomous Cohorts of Collaborating Drones.* arXiv, 2020. DOI: 10.48550/ARXIV.2010.04101. Dostupné z: https://arxiv.org/abs/2010.04101.

[6] Corrigan, F. *Quick drone parts overview along with Handy DIY Tips.* Aug 2020 [cit. 2022-4-7]. Dostupné z: https://www.dronezon.com/learn-about-drones-quadcopters/drone-components-parts-overview-with-tips/.

[7] Dieter Schmalstieg, T. H. *Augmented Reality: Principles and Practice.* Addison-Wesley Publishing Company, 2016. ISBN 978-0321883575.

[8] Funk, M. Human-Drone Interaction: Let's Get Ready for Flying User Interfaces! *Interactions.* New York, NY, USA: Association for Computing Machinery. apr 2018, zv. 25, č. 3, s. 78–81, [cit. 2022-4-7]. DOI: 10.1145/3194317. ISSN 1072-5520. Dostupné z: https://doi.org/10.1145/3194317.

[9] Hubinák, R. *Aplikace pro efektivní řízení dronu s využitím rozšířené virtuality.* Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22839/.

[10] Kyrkou, C., Timotheou, S., Kolios, P., Theocharides, T. a Panayiotou, C. Drones: Augmenting Our Quality of Life. *IEEE Potentials.* 2019, zv. 38, č. 1, s. 30–36. DOI: 10.1109/MPOT.2018.2850386.

[11] Li, Y. a Liu, C. Applications of multirotor drone technologies in construction management. *International Journal of Construction Management*. Taylor & Francis. 2019, zv. 19, č. 5, s. 401–412, [cit. 2022-4-7]. DOI: 10.1080/15623599.2018.1452101. Dostupné z: https://doi.org/10.1080/15623599.2018.1452101.

[12] Milgram, P., Takemura, H., Utsumi, A. a Kishino, F. Augmented reality: a class of displays on the reality-virtuality continuum. In: Das, H., ed. *Telemanipulator and Telepresence Technologies*. SPIE, 1995, sv. 2351, s. 282 – 292 [cit. 2022-4-7]. DOI: 10.1117/12.197321. Dostupné z: https://doi.org/10.1117/12.197321.

[13] Mogili, U. R. a Deepak, B. B. V. L. Review on Application of Drone Systems in Precision Agriculture. *Procedia Computer Science*. 2018, zv. 133, s. 502–509, [cit. 2022-4-7]. DOI: https://doi.org/10.1016/j.procs.2018.07.063. ISSN 1877-0509. International Conference on Robotics and Smart Manufacturing (RoSMa2018). Dostupné z: https://www.sciencedirect.com/science/article/pii/S1877050918310081.

[14] Poljak, M. *Drone Parts And Components Overview With DIY Tips – Drone Tech Planet*. 2020 [cit. 2022-4-7]. Dostupné z: https://www.dronetechplanet.com/quick-drone-parts-overview-with-diy-tips/.

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

[16] Sedlmajer, K., Bambušek, D. a Beran, V. Effective Remote Drone Control Using Augmented Virtuality. In: *Proceedings of the 3rd International Conference on Computer-Human Interaction Research and Applications 2019*. SciTePress - Science and Technology Publications, 2019, s. 177–182. DOI: 10.5220/0008349401770182. ISBN 978-989-758-376-6. Dostupné z: https://www.fit.vut.cz/research/publication/12006.

[17] Shahmoradi, J., Talebi, E., Roghanchi, P. a Hassanalian, M. A Comprehensive Review of Applications of Drone Technology in the Mining Industry. *Drones*. 2020, zv. 4, č. 3, [cit. 2022-4-7]. DOI: 10.3390/drones4030034. ISSN 2504-446X. Dostupné z: https://www.mdpi.com/2504-446X/4/3/34.

Príloha A

Plagát

Príloha B

Obsah priloženej SD karty

doc/technicka-sprava.pdf technická správa vo formáte PDF
doc/video.mp4 prezentačné video
doc/plagat.pdf plagát vo formáte PDF
doc/latex zdrojové súbory technickej správy vo formáte jazyka Latex
src/ zdrojové súbory Unity projektu
build/ skompilovaná aplikácia pre platformu Windows