České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra počítačové grafiky a interakce

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Roman Janovský

Studijní program: Otevřená informatika
Obor: Počítačová grafika a interakce

Název tématu: Operátorské stanoviště pro vizualizaci a řízení autonomních bezpilotních prostředků

Pokyny pro vypracování:

Navrhněte a implementujte systém pro sledování a řízení více autonomních bezpilotních prostředků (UAV). Systém bude realizován v 3D enginu a bude umožňovat: vizualizaci prošlé a budoucí trajektorie jednotlivých UAV, zjednodušený 3D model prostředí a terénu, infromace o stavu jednotlivých prostředků, přepínání pohledů, výběr prostředků a jejich řízení/přiřazení k misi, definici bezletových zón. Systém napojte na simulační / řídicí systém implementovaný v rámci projektu AgentFly na katedře počítačů FEL a průběžně konzultujte funkcionalitu implementovaného systému s vybraným zástupcem této skupiny.
Proveďte uživatelské testování dle scénáře navrženého ve spolupráci s pověřenou osobou podílející se na vývoji systému AgentFly.

Seznam odborné literatury:

Jan Balata, Ladislav Čmolík, Zdeněk Míkovec. On the Selection of 2D Objects Using External Labeling. Proceedings of SIGCHI Conference on Human Factors in Computing Systems, pp. 2255-2258, 2014.

Vedoucí: Ing. David Sedláček, Ph.D.

Platnost zadání: do konce zimního semestru 2018/2019

prof. Ing. Jiří Žára, CSc.
vedoucí katedry

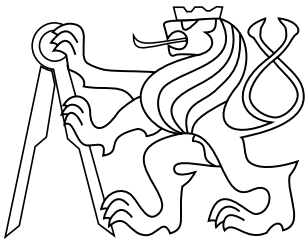
prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 3.4.2017

diplomová práce

Operátorské stanoviště pro vizualizaci a řízení autonomních bezpilotních prostředků

Janovský Roman



Květen 2017

Sedláček David Ing., Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra počítačové grafiky a interakce

Poděkování

V prvé řadě bych chtěl poděkovat svému vedoucímu práce Ph.D. Davidu Sedláčkovi za asertivní vedení, poskytnutí vhodných podkladů a za čas strávený při kontrole mé práce. Dále bych rád poděkoval Ing. Martinu Seleckému, který si vždy našel čas, aby mi vysvětlit cokoli týkající se simulátoru bezpilotních prostředků.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 21. 5. 2017 _____

Abstrakt

Cílem mé diplomové práce je vytvoření vizualizace bezpilotních (leteckých) prostředků, která jsou malá a rychle se pohybují, a napojit tuto vizualizaci na AgentFly simulátor. Také zjišťuji, jestli je vhodné využít webových služeb poskytujících satelitní snímky, výškové mapy a další informace jakými jsou budovy.

Klíčová slova

vizualizace uav; bezpilotní prostředky; tile services; procedurální terén; výběr malých objektů

Abstrakt

The goal of my master thesis is to create visualisation of unmanned (aerial) vehicles, which are small and fast objects, and connect it to AgentFly simulator. It also tests how useful would be use of web services providing satellite imagery, heightmaps and other information like buildings and landuse.

Keywords

uav visualisation; unmanned aerial vehicles; tile services; procedural terrain; picking small objects

Obsah

1 Úvod **1**
1.1 Struktura . . . 1

2 Analýza **2**
2.1 Uživatelské požadavky . . . 2
2.2 Webové služby . . . 3
2.2.1 Mercatorova projekce . . . 3
2.2.2 Výškové mapy . . . 5
2.2.3 Satelitní snímky . . . 6
2.2.4 Vektorové dlaždice . . . 6
2.3 Simulátor a síťová komunikace . . . 9
2.3.1 Hlavička dat . . . 10
2.3.2 Informace . . . 10
2.4 Příkazy . . . 11
2.4.1 Pořadí bytů . . . 13
2.5 Unity . . . 13
2.5.1 Unity terén . . . 13
2.5.2 Pořadí volání vnitřních funkcí Unity . . . 14
2.5.3 Komponentový návrh . . . 15

3 Návrh řešení **16**
3.1 Vizualizace ve 3D . . . 16
3.2 Bezpilotní prostředky . . . 16
3.3 Uživatelské rozhraní . . . 17
3.4 Kamery . . . 18
3.5 Počítání v pohyblivé řádové čárce . . . 19
3.6 Svět . . . 19

4 Implementace **20**
4.1 Unity . . . 20
4.1.1 Paralelní exekuce . . . 20
4.2 Generování světa . . . 21
4.2.1 World.txt . . . 21
4.2.2 Dlaždice terénu . . . 22
4.2.3 Budovy . . . 23
4.2.4 Třída ThreadedJob . . . 23
4.2.5 Třída TileCache . . . 24
4.2.6 Stahování dlaždic . . . 24
4.3 Objekty spjaté se simulací . . . 25
4.3.1 Trajektorie . . . 25
4.3.2 Zóny . . . 26
4.3.3 Agenti . . . 27
4.4 Třída Pickable . . . 28
4.4.1 Třída Agent a třída Zone . . . 28
4.5 Štítky objektů . . . 28
4.6 Třídy pro správu jednotlivých součástí programu . . . 29
4.7 Kamera . . . 29

4.8 Logování . . . 30
4.9 Vytváření objektů . . . 30
4.9.1 MissionCreator . . . 31
4.9.2 WaypointsCreator . . . 31
4.9.3 SACreator . . . 31
4.9.4 NFZCreator . . . 31
4.10 Ovládání . . . 31
4.11 Uživatelské rozhraní . . . 32
4.11.1 Mise . . . 33
4.12 Nastavení . . . 34
4.13 Port na Android . . . 36

5 Testování **37**
5.1 Performance testing . . . 37
5.1.1 Nastavení testů . . . 38
5.2 Uživatelské testování . . . 38
5.2.1 První testování . . . 39
5.2.2 Druhé testování . . . 40
5.2.3 Společné problémy participantů . . . 43

6 Závěr **44**

Literatura **45**

Příloha A - Dotazníky pro testování **46**

Příloha B - Screenshoty z aplikace **47**

Příloha C - Obsah přiloženého CD **49**

Zkratky

Použité pojmy:

NASA	The National Aeronautics and Space Administration
USGS	United States Geological Survey
ODbL	Open Database License
UAV	Unmanned Aerial Vehicle
FPS	Framerate per Second (Počet snímků za vteřinu)
SDK	Software Development Kit

1 Úvod

Bezpilotní prostředky se dnes používají v krizových situacích nebo ve špatně dostupných místech pro monitorování oblasti, jelikož pořízení není natolik nákladné, jako ztráta osoby. Zároveň se stávají velmi populárními jako forma volnočasové zábavy. Samozřejmě by tedy bylo vhodné, kdyby existovala možnost, jak rychle a přehledně informovat o stavu bezpilotních prostředků a stavu jejich okolí, což by umožnilo rychlejší a kvalitnější rozhodnutí ze strany operátora, obzvláště v kritickém čase.

Ve spolupráci s lidmi z Agent Technology Center, kteří se zabývají simulací řízení letového provozu jak normálních civilních letadel, tak i bezpilotních letových prostředků, jsme se rozhodli, že takovou vizualizaci vytvoříme. Jelikož i oni sami postrádají možnost vhodného řešení, tj. zatím mají pouze vizualizaci ve 2D prostoru bez jakékoli informace o okolí, tak bychom napojili tento vizualizační program na jejich simulátor, čímž by se získaly jak testovací data, tak potřebné podněty pro zlepšení. Při napojení na simulátor bude potřeba kromě pouhé vizualizace i vytvoření ovládání, které by umožnilo operátorovi vytvářet mise pro UAV v simulaci, nechat je hlídkovat na nějakou oblastí nebo proletět ve vzduchu trasu.

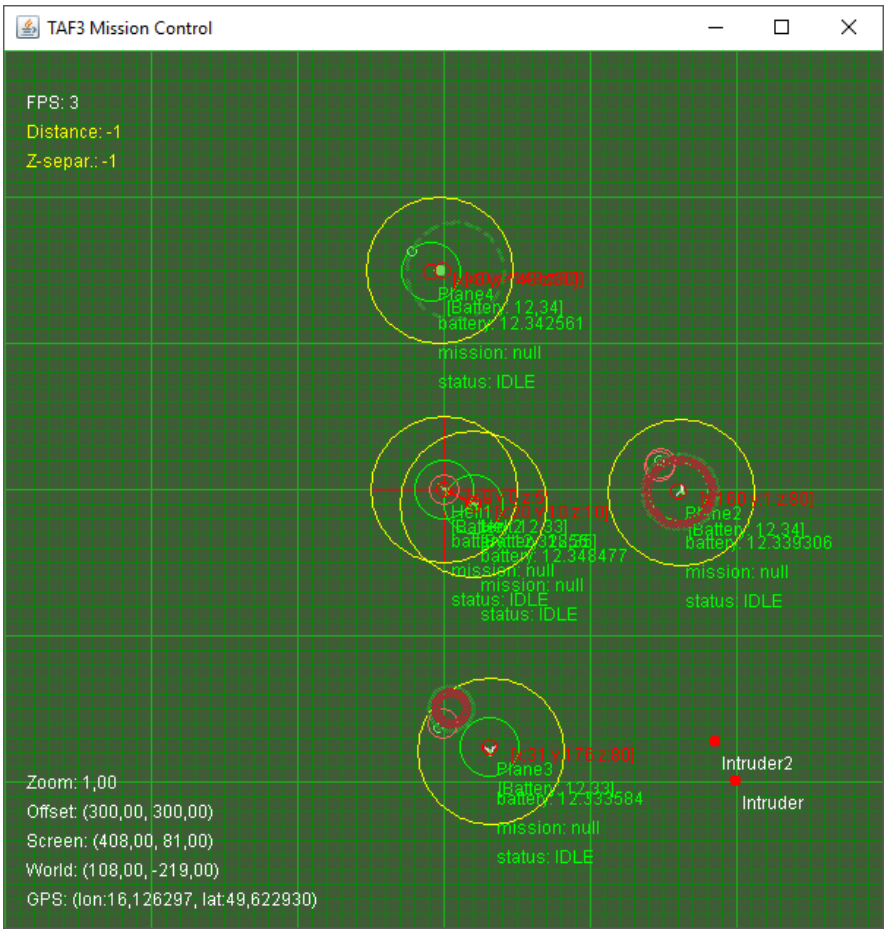
1.1 Struktura

Práce je rozdělena se skládá ze tří kapitol. Kapitola Analýza je rozbor problému, uživatelské požadavky a využitelné služby. V kapitole Návrh řešení vysvětlím, jak by program měl fungovat, na což navážu v kapitole Implementace, kde již budou konkrétní detaily. V kapitole Testování shrnu výsledky výkonu různých řešení pro vytváření světa a výsledky z uživatelského testování. V Závěru nabídnu možnosti rozšíření a shrnu výsledky práce.

2 Analýza

2.1 Uživatelské požadavky

Nynější ovládání simulátoru je založené na Javě a je pouze ve 2D a zpočátku není snadné se v něm orientovat. Tato práce nemá přímo nahradit stávající ovládání, ale především zlepšit vizuální stránku a přehlednost. Na těchto základních požadavcích jsou postaveny všechny ostatní.



Obrázek 1 Stávající vizualní stránka simulátoru od AgentFly

Program by však měl poskytovat možnost základního plánování misí. Měly by tedy jít vytvořit mise a přiřadit je k bezpilotním prostředkům. Mise se mohou skládat ze sledování vytyčené oblasti, průlet navigačními body vytyčnou trasou, sledovat pohybující se cíl a to vše za podmínky, že se vyhnou bezletovým zónám. Tyto požadavky plynou z možností existující síťové komunikace, která je popsána v části 2.3 Simulátor a síťová komunikace.

2.2 Webové služby

Není možné, aby celá planeta byla popsaná jediným souborem nebo obrázkem. Představte si, že by služby jako Google Street View nebo Google Earth s detailními snímky míst při každé žádosti o zobrazení místa na mapě musely stáhnout texturu s rozměry několika milionů pixelů. Vezmeme-li v úvahu jen samotnou velikost takové textury, tak by brzy byly jejich servery přetížené.

Proto takové služby poskytují dava ve formátu čtvercových dlaždic. Dlaždice může být representována jako rastrový obraz, např. pro satelitní snímky, mapy nebo například nadmořské výšky terénu či normálové mapy terénu [1]. Samozřejmě nemá smysl dělit pouze textury, ale i ostatní - vektorová - data. Ty obsahují mimo jiné informace o budovách, silnicích, zástavbě a využití území nebo hranice států a jiné. I zde by totiž nemělo smysl zobrazovat nejdetalnější budovy při pohledu z velké vzdálenosti.

Terén v Unity je implementován pomocí výškové mapy. Tuto výškovou mapu získávám ze služby poskytnuté Mapboxem[2]. Satelitní snímky jsou také získané z jedné z mnoha služeb Mapboxu[2]. Mapbox využívá dat od NASA, USGS, DigitalGlobe a jiných komerčních poskytovatelů. Množství stažených dat Mapbox limituje na 50000 za měsíc a umožňuje je ukládat pro offline použití. Vzhledem k tomu, že na pokrytí většiny Londýna stačí 6000 snímků, tak to netvoří problém pro bezpilotní prostředky s doletem maximálně 10 kilometr.

Vektorová data s informacemi o budovách a územním využití ve formátu GeoJSON získávám od Mapzenu. Ten obsahuje informace z OpenStreetMap, které jsou dostupné pod ODbL. Mapzen limituje množství stažených vektorových dat na 50000 požadavků za měsíc.

2.2.1 Mercatorova projekce

Jak již bylo výše zmíněno, data se získávají ve formě dlaždic. Tyto dlaždice jsou čtvercového tvaru kvůli snažšímu dělení a přístupu. To však neznamená, že oblast zachycená je také čtvercová. Služby využívají válcové projekce, Mercatorovy projekce, která mapuje Zemi do čtvercové sítě. To ale znamená, že je rozdíl ve velikosti mezi dlaždicemi kolem rovníku a u severního pólu. Zachycená oblast v jedné dlaždici je ve skutečnosti lichoběžníkového tvaru, ale pokud se pohybujeme v rozmezí desítek kilometrů, pak je tato odchylka zanedbatelná.

Mercatorova projekce je válcová projekce, kde poledníky i rovnoběžky jsou rovné a navzájem kolmé. Jelikož poledníky vycházejí z pólů, tak při jejich narovnání dochází k oné deformaci, která se zvětšuje s rostoucí zeměpisnou šířkou směrem od rovníku. Mercatorova projekce je konformní zobrazení, a tedy kolem každého bodu zachovává úhly.

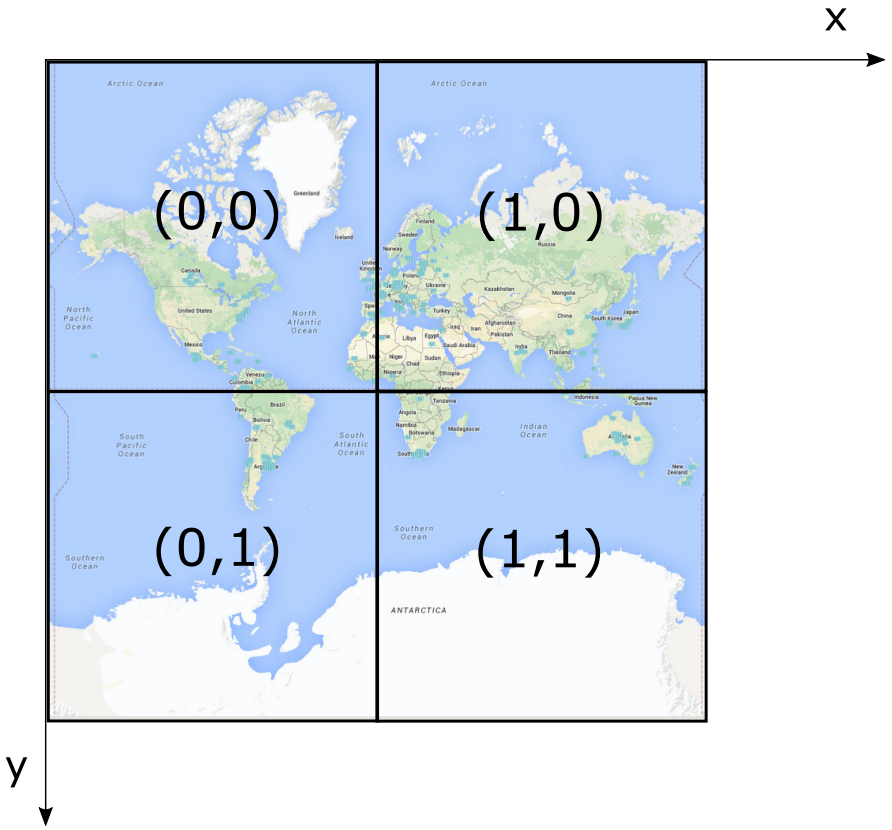
Webové služby využívají variantu Mercatorovy projekce nazývanou Web Mercator[3]. Ta je omezena na 85.051129°severní i jižní zeměpisné šířky. Tato zeměpisná šířka je zvolena tak, že získaná projekce je čtvercová. Sice chybí jižní a severní pól, ale vzhledem ke smyslu mapových služeb nebo vizualizace UAV, to není nijak limitující. Dalším rozdílem je, že se počátek přesune do levého horního rohu a velikost světa se změní na 256×256.

Výpočet pozice ve Web Mercatoru z GPS souřadnic lze provést pomocí rovnice (1):

$$\begin{aligned} x &= \frac{128}{\pi} \times 2^{zoom} \times (\lambda + \pi) \\ y &= \frac{128}{\pi} \times 2^{zoom} \times (x - \ln(\frac{\pi}{4} + \frac{\phi}{2})) \end{aligned} \tag{1}$$

, kde (x, y) jsou reálné souřadnice v rozsahu $< 0, 256 > \times 2^{zoom}$, ϕ a λ jsou v tomto pořadí zeměpisná šířka a zeměpisná délka. Souřadnice (x, y) jednoznačně určují místo na mapě v dané úrovni.

Jak již bylo zmíněno, pro podrobné rozlišení by mapa musela být nepřiměřeně velká, a proto se ukládá v hierarchii čtvercových sítí, kde *zoom* uvádí, v jaké úrovni se nacházíme. První úroveň (*zoom* 0) obsahuje pouze jednu dlaždici o velikosti 256×256. Druhá úroveň (*zoom* 1) potom obsahuje 2×2 dlaždice, každá o velikosti 256×256. Pozici dlaždice lze jednoduše získat ze souřadnic (x, y) zaokrouhlením dolů.



Obrázek 2 Souřadný systém Web Mercatoru a rozložení dlaždic na druhé úrovni

Rozlišení v úrovni pro jeden pixel S $[\frac{m}{pixel}]$ je závislé na zeměpisné šířce a lze spočítat pomocí rovnice (2):

$$S = C \times \frac{\cos \phi}{2^{zoom+8}} \tag{2}$$

, kde C je obvod rovníku v metrech, což je 40075016.686 metrů, a kde přičítáme 8 v mocnině za velikost dlaždice 256×256. Pro přehled rozlišení se můžeme podívat na

tabulku 1, kterou můžeme použít, abychom se zbavili dodatečných výpočtů. Protože hodnoty v tabulce jsou uvedeny pro rovník, pak je potřeba je přizpůsobit pro konkrétní zeměpisnou šířku. Pro přehled můžeme obecně pro Českou republiku počítat se zeměpisnou šířkou 50°, což činí cos $50° = 0.6427$ a pro *zoom* 15 je tedy rozlišení $3.07[\frac{m}{pixel}]$.

Zoom	Rozlišení $[\frac{m}{pixel}]$
0	156543.03
1	78271.52
2	39135.76
3	19567.88
4	9783.94
5	4891.97
6	2445.98
7	1222.99
8	611.50
9	305.75
10	152.87
11	76.437
12	38.219
13	19.109
14	9.5546
15	4.7773
16	2.3887

Tabulka 1 Rozlišení v $[\frac{m}{pixel}]$ pro jednotlivé úrovně na rovníku[4]

2.2.2 Výškové mapy

Výškové mapy získané ze služby Mapboxu jsou kódované do červeného, zeleného a modrého kanálu ve formátu png. V každém pixelu je pomocí barvy uložena skutečná hodnota nadmořské výšky v metrech. Jelikož každý barevný kanál poskytuje 256 hodnot, je možné zachytit 16777216 jedinečných hodnot. Ty jsou mapovány po $0.1m$, což poskytuje dostatečnou vertikální přesnost pro kartografické a jiné 3D aplikace[5].

Dlaždice je možné ze služby dostat pomocí odkazu:

`https://api.mapbox.com/v4/mapbox.terrain-rgb/z/x/y.pngraw?access_token=api-klíč`,

kde důležitá část je $z/x/y$ a api-klíč. Hodnoty z, x, y znamenají v tomto pořadí zoom, pozici dlaždice na ose X, pozici dlaždice na ose Y. Zoom je potřeba zvolit již předem nebo dynamicky v závislosti na vzdálenosti od terénu podle potřeb aplikace a souřadnice dlaždice lze dopočítat pomocí rovnice (1), případně relativně k nějaké počáteční dlaždici, pokud je známa. K 11.5.2017 je možné získat až zoom 23.

Výšku H lze získat z barvy obrázku pomocí rovnice (3):

$$H = -10000 + ((R \times 256 \times 256 + G \times 256 + B) \times 0.1) \tag{3}$$

2.2.3 Satelitní snímky

Satelitní snímky jsou získávány podobným způsobem jako výškové mapy, pouze se mění adresa URL:

`https://api.mapbox.com/v4/mapbox.satellite/z/x/y.jpg?access_token=api-klíč`,

kde $z/x/y$ a api-klíč je totožný jako u výškových map. Pokud by se zvolily jiné než satelitní snímky, např. mapbox.streets, pak je možné specifikovat i formát a kompresi obrázku, např. jpg70. Jelikož však používám satelitní mapy a navíc mi záleží na detailech, takže komprese ani není potřeba.

2.2.4 Vektorové dlaždice

Při vizualizaci UAV nás mimo jiné zajímá, kudy prolétá. Například průlet budovou je nežádoucí. Celkově je lepší naplánovat trasu tak, aby se UAV vyhýbalo oblastem s velkým množstvím překážek, které tvoří především budovy či zdi a hradby. I kdyby s nimi simulace trajektorie nepočítala, pak se jim stále dá vyhnout naplánováním podrobnější trasy.

Webová služba Mapzenu, ze které získávám vektorová data, rozděluje informace do různých vrstev. Například budovy a zdi, které mě především zajímají, patří do dvou různých vrstev a to *buildings a landuse*. Data mají formát JSON a lze je získat z URL:

`https://tile.mapzen.com/mapzen/vector/v1/{požadované-vrstvy}/{z}/{x}/{y}.json`,

kde $z/x/y$ jsou opět souřadnice již výše zmíněné a požadované vrstvy mohou být z těchto 9 vrstev: *boundaries*, *buildings*, *earth*, *landuse*, *places*, *pois*, *roads*, *transit* a *water*. Můžeme buď vybrat specifickou kombinace, tj. třeba $landuse, buildings$, nebo požádat o všechny vrstvy, tj. zadat *all*[6].

Vrstvy buildings a landuse

Vrstvy $landuse, buildings$ také obsahují geometrii, která je častá především ve vrstvách *water*, *landuse*, *buildings*, a může být bodem, řetězcem segmentů nebo polygonem. Kdy budovy a zdi jsou polygonem a mohou být buď $Polygon$, nebo $MultiPolygon$. Důležitý je především $Polygon$. Každá z vrstev má velké množství atributů, mezi nimiž jsou pro budovy důležitá výška a pro územní využití druh zástavby a to *city_wall* (např. Vyšehradské hradby).

Za každou požadovanou vrstvu je zde jeden objekt stejného jména. Konkrétní popis budovy (ukázka 2.2.4) nebo zdi (ukázka 2.2.4) je pak uložen v poli se jménem $features$. U nich si pak vyhledám, jestli je typ geometrie polygon a potřebné atributy, které jsou v ukázkách také vypsány.

```
{
   "buildings": {
      "type": "FeatureCollection",
      "features": []
   },
   "landuse": {
      "type": "FeatureCollection",
      "features": []
   }
}
```

Ukázka 2.1 Příklad základní struktury GeoJSON pro landuse a buildings

```
{
   "geometry": {
      "type": "Polygon",
      "coordinates": [
         [
            [
               14.41974311,
               50.0677521
            ],
            [
               14.41979341,
               50.06783312
            ] ...
         ]
      ]
   },
   "properties": {
      "name": "Nádraží Praha-Vyšehrad",
      "height": 8,
      "id": 34067460
   }
}
```

Ukázka 2.2 Příklad struktur pro budovu

```
{
   "geometry": {
      "type": "Polygon",
         "coordinates": [
            [
               [
                  14.41955566,
                  50.06531011
               ],
               [
                  14.41987345,
                  50.06525018
               ] ...
            ]
         ]
   },
   "type": "Feature",
   "properties": {
      "name": "Vyšehradské hradby",
      "kind": "city_wall",
      "id": 8888301
   }
}
```

Ukázka 2.3 Příklad struktur pro budovu

Ostatní vrstvy

Zde stručně popíši ostatních 7 vrstev, které nevyužívám, mezi něž patří *boundaries*, *earth*, *places*, *pois*, *roads*, *transit* a *water*. Tyto dodatečné vrstvy by bylo možné využít o obohacení generované krajiny, automatické generování bezletových zón (například nad letištěm nebo elektrárnami). Kompletní přehled můžete vidět na stránkách Mapzenu[6].

Boundaries Reprezentuje administrativní hranice států, krajů nebo jiných územních celků. Data jsou kombinací dat z OpenStreetMap pro *zoom* $>= 8$ a pro zbytek jsou získaná z NaturalEarth. Data také obsahují geometrii a tou jsou segmenty.

Earth Vrstva obsahuje kontinenty a přírodní hranice jako jsou útesy, údolí nebo ostrovy. Data jsou opět tvořena geometrií, kde země tvoří polygony, kontinenty, ostrovy nebo poloostrovy jsou pouze body myšlené pro umístění označení a útesy nebo údolí jsou tvořeny řetězcem úseček.

Places Obsahuje územní celky v podobě bodů. Patří sem kontinenty, státy, mesta a jiné územní lokality. Také je z této vrstvy možné určit počet obyvatel nebo hlavní město územního celku.

POIs - Points of Interest Bodové označení zajímavého místa. Spadají sem parky, železnice, nemocnice, školy, autobusové zastávky a nebo prostě jen místa pro relaxaci, jako je lavička v parku nebo strom.

Roads Sem nepatří pouze cesty, ulice, dálnice nebo silnice pro motorová vozidla, ale i vzdušné cesty, železnice nebo cesty trajektů. Dále jsou poskytnuty informace, jestli cesta vede tunelem, jestli je jednosměrná nebo vede přes most.

Transit Tato vrstva je podobná vrstvě *roads*, jen sem spadají cesty hromadných dopravních prostředků, tedy tramvají, lanovek, metra nebo jednokolejek. Daty jsou opět řetězce úseček. Může být uvedeno, kdo je poskytovatelem hromadné dopravy.

Water Vrstva *waters* obsahuje vodní plochu - řeky, kanály, oceány, jezera. Řeky jsou tvořeny úsečkami a vodní plochy jako jezera jsou ohraničené polygony. Vrstva také využívá bodových dat, která slouží jako popisky.

Api klíč

Již jsem zmiňoval, že pro stažení dat jsou potřeba api klíče. Jak Mapzen, tak Mapbox nebo i Google požadují, aby přístup k jejich službám nebyl anonymní, a pokud se o to pokusíte, tak místo dat dostanete jako odpověď *bad request*. Jelikož využívám služby Mapzenu a Mapboxu, tak jsem u nich také registrován jako vývojář a api klíče mám. Pro potřebu aplikace se nacházejí v souboru world.txt. Pro testování a vývoj využívám své klíče, ale jiný uživatel se musí sám zaregistrovat, jinak by brzy mohlo dojít k naplnění kvót za měsíc.

Api klíč je Vaším identifikátorem pro přístup ke službám. Díky němu provozovatel může sledovat, jak často stahujete a případně Vás informovat o překročení limitů nebo Vám vytvořit statistiku Vašeho přístupu.

Získání takového klíče je jednoduché a bezplatné. Samozřejmě je možné si zaplatit navýšení měsíčního limitu, ale to nebude nejspíše potřeba. Registraci je možné provést na stránkách Mapzenu[7] nebo zde pro Mapbox[8].

2.3 Simulátor a síťová komunikace

Stávající simulátor je založen na programovacím jazyku Java, a tak veškerá vnitřní komunikace probíhá pomocí serializace objektů. Avšak Unity využívá C#. Nejde zde tedy využít serializace objektů, protože se jedná o komunikaci mezi různými platformami, a tak je potřeba vytvořit nový protokol pro posílání dat. Ten byl založen na posílání paketů bytů, kdy se nejdříve pošle hlavička obsahující velikost a druh dat, a poté následují data převedená na pole bytů.

Ke komunikaci slouží příkazy a informace, které reprezentují různé události. Každý z příkazů má své vlastní identifikační číslo, podle kterého lze poznat, jakým způsobem se poslaná data mají načíst. Např. $CommandSetSurveillanceArea$ přiřadí vybraným UAV danou oblast, nad kterou mají dohlížet, nebo $InfoUAVTelemetry$ informuje o stávající poloze UAV. Příkazy mění stav simulace a informace o něm pouze informují.

Komunikace mezi simulátorem a vizualizací probíhá přes TCP na portu 30000. Ve stávající verzi není možné nastavit adresu, pod kterou lze simulátor najít, používá se *localhost*.

2.3.1 Hlavička dat

Hlavička dat se skládá z 12 bytů, kde byty 0..3 značí délku následujícího paketu, byty 4..7 jsou číslo sekvence, byty 8..9 reprezentují kód příkazu nebo informace a byty 10..11 značí verzi. Když se posílá hlavička s daty o délce 61 bytů, pak je v délce hlavičky uvedena délka 69, jelikož se přidává 8 bytů jako vložka.

2.3.2 Informace

Existuje 5 informací, které se posílají: informace o pozici pozemního cíle (chodec), informace o pozici a stavu UAV (křídlo, kvadrokoptéra, vozidlo), informace o přiřazení UAV k misi, informace o aktuálně vykonávané akci pro UAV a predikce trasy UAV. V následující části bude podrobný popis dat, která lze získat. Jejich pořadí je vypsáno takové, jak je potřeba ho zapsat nebo přečíst do/z pole bytů.

Pozemní cíl

`string` Jméno pozemního cíle.

`Vector3d` Pozice cíle - pozice v rovině, výška nad zemí (dále stejná notace).

Telemetrie UAV

`string` Jméno UAV.

`Vector3d` Pozice UAV.

`int` Autopilot.

`double` Směr natočení.

`double` Pozemní rychlost [m/s].

Vykonávání mise

`string` Jméno UAV.

`string` Jméno mise.

`string` Stav mise.

Predikce trajektorie UAV

`string` Jméno UAV.

`List<Vector3d>` Seznam bodů na budoucí trajektorii.

Predikce trajektorií simulátor posílá každou minutu nebo poté, co se trajektorie přeplánuje. Telemetrie je posílána každých 500 ms.

Přiřazení UAV k příkazu

`string` Jméno UAV.

`Příkaz` Příkaz, který má UAV vykonávat. Buď sledování cíle, nebo oblasti.

Obecně platí, že jakýkoli list (tedy svým způsobem i řetězec znaků - string), nejdříve zapíše svoji délku, a pak jednotlivé položky (znaky). Pokud by nastal nějaký problém (ukazatel na list nebo string je null), pak se zapíše jako délka -1, a nemá tedy smysl pokoušet dále číst. Vector3d se zapisuje a čte stejně, jako by se četl třikrát double.

2.4 Příkazy

Příkazy slouží k vynucení změny v simulátoru, kterou může být přiřazení UAV k misi nebo vytvoření nové bezletové zóny. Jinými slovy: slouží k vytváření misí a plánování letových tras.

UAV je možné slučovat do skupin. Tyto skupiny neslouží k ničemu jinému, než ke snazšímu přiřazení společných misí. Jakmile se přiřadí a naplánuje mise, pak se ze skupiny vymažou a skupina se zruší.

Přiřazení UAV do skupiny

`string` Jméno UAV.

`string` Jméno skupiny.

Odebrání UAV ze skupiny

`string` Jméno UAV.

`string` Jméno skupiny.

Aby se bezletová zóna projevila na letovém plánu UAV v simulátoru, musí se znovu přeplánovat mise, kterou zrovna provádí.

UAV ignoruje bezletovou zónu

`string` Jméno UAV - Je možné použít jméno jako "*", kdy se přiřadí všem.

`string` Jméno bezletové zóny.

Přidat bezletovou zónu pro UAV

`string` Jméno UAV.

`string` Jméno bezletové zóny.

`byte` Typ bezletové zóny - pouze válcovitá bezletová zónu.

`Vector3d` Střed bezletové zóny.

`double` Poloměr bezletové zóny.

`double` Výška bezletové zóny.

Průlet UAV navigačním bodem

`string` Jméno UAV.

`string` Jméno navigačního bodu.

Průlet UAV navigačním bodem je zároveň informací, zároveň příkazem, jelikož je možné říct UAV, že tímto bodem už nemusí letět (již jim prolétl).

Přiřaď navigační body k více UAV

`string`	Jméno mise.
`List<string>`	Jména UAV přiřazená k misi.
`List<Vector3d>`	Pozice navigačních bodů, kterými mají UAV proletět.

Přiřaď navigační body k jednomu UAV

`string`	Jméno UAV.
`List<CartesianWaypoint>`	Navigační body, kterými mají UAV proletět.

CartesianWaypoint obaluje informace o místu, kterým má UAV proletět. Kromě pozice uvádí i požadovanou rychlost a dobu, kdy má místem proletět, případně zpoždění.

CartesianWaypoint

`Vector3d`	Pozice navigačního bodu.
`Vector3d`	Zapíše tři 0.
`double`	Požadovaná rychlost průletu.
`double`	Požadovaná doba průletu.
`double`	Maximální zpoždění průletu.

Misi mohou být přiřazeny různé úkony pro UAV. Momentálně jsou povoleny sledování oblasti, sledování cíle a průlet navigačními body a musejí být uvedeny v tomto pořadí, i kdyby se neměly vykonávat, tj. pošle se prázdný seznam.

Přiřaď misi skupině

`string`	Jméno skupiny.
`string`	Jméno mise.
`Příkazy`	Příkazy, ze kterých se mise skládá, uvedeny ve správném pořadí.

Sledovaná oblast je obdelníkového tvaru a je definována levým dolním rohem oblasti a horním rohem oblasti.

Sleduj oblast

`Vector3d`	Počátek sledované oblasti.
`Vector3d`	Konec sledované oblasti.
`long`	Čas začátku sledování.
`long`	Čas konce sledování.

Sleduj cíl

`List<string>`	Jména sledovaných cílů.
`List<string>`	Jména sledovaných cílů.
`long`	Čas začátku sledování.
`long`	Čas konce sledování.

2.4.1 Pořadí bytů

Pořadí bytů je důležités pro síťovou komunikaci mezi různými počítači (nebo mezi různými aplikacemi). Je potřeba vědět, jak se s byty nakládá. Java má vždy Big-endian, tj. nejvýznamnější byte se uloží na paměťové místo s nejnižší adresou. Jelikož je u simulátoru díky Javě zaručeno, že má vždy Big-endian, pak tuto část musí řešit pouze vizualizační nástroj.

Pokud zařízení, na kterém běží vizualizace, používá Big-endian, tak se nemusí provést žádné změny. Pokud ale používá Little-endian, pak je potřeba obrátit pořadí bytů a teprve potom je odeslat. Pro čtení dat je postup obdobný.

2.5 Unity

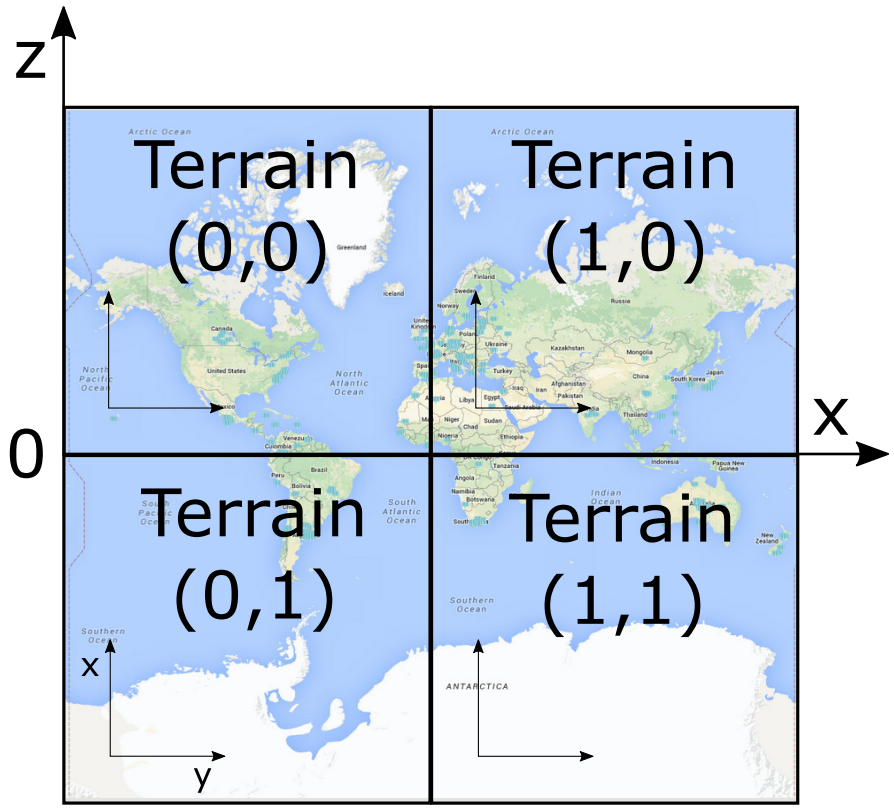
Pro samotnou vizualizaci využívám herního enginu Unity 3D. Unity nabízí rozsáhlou škálu funcí, z nichž jedna právě umožňuje vytváření terénu pomocí výškových map a zároveň umí podle vzdálenosti kamery terén zjednodušit. V této části přiblížím některé funkce a nastavení, která využívám pro vizualizaci.

2.5.1 Unity terén

Terén v Unity je tvořen pomocí výškové mapy. Velikost této mapy lze nastavit, ale musí být o rozměrech $2^n + 1$ a minimálně o velikosti 33×33. Jelikož stažená výšková mapa má velikost 256×256, nemá smysl nastavovat větší rozlišení výškové mapy terénu na více než 257. Navíc je potřeba výškovou mapu převzorkovat, aby měla požadované rozlišení 257×257[9].

Další důležitou součástí terénu je *splatmap*, tj. jedna nebo více textur, které se mapují na terén. Mapování se dělá pomocí *alphamap*. *Alphamap* je taktéž textura (nebo více), která udává jakým způsobem se mixují dohromady *splatmaps*. Každý pixel *alphamap* je barva v RGBA, jejíž součet jednotlivých kanálů musí být 1. Takto se určuje, jakým způsobem se michají jednotlivé *splatmaps* dohromady. Pokud by se použilo 5 *splatmap*, pak je potřeba mít dvakrát *alphamap*. Mně však stačí pouze jedna *splatmap* o velikosti celého terénu, kterou je satelitní snímek odpovídající dané dlaždici/terénu.

Dále je potřeba nastavit samotnou velikost terénu. Výšky jsou reálná čísla ve výškové mapě v rozmezí $< 0, 1 >$ a z nich se dopočítá výška terénu právě ze zadané velikosti. Můžeme předpokládat, že výška terénu nebude více jak $10000m$, dokud zůstaneme na Zemi, pak chybí pouze velikosti pro délku a šířku terénu. Ty jsou závislé na GPS poloze a můžeme ji získat pomocí rovnice 2, kterou pouze vynásobíme 256, což je velikost jedné dlaždice.



Obrázek 3 Souřadný systém terénu v Unity včetně odpovídajících dlaždic z Mercatorovy projekce a souřadný systém pro výškovou mapu terénu

Pozice terénu v rovině XZ je určena jeho levým dolním rohem. Při porovnání se souřadným systém Web Mercatoru (viz. Obrázek 2) je vidět, že si odpovídají osy z,y, ale jsou otočeny v opačném směru. Počátek se tedy musí přesunout z levého horního rohu (Web Mercator) do levého dolního rohu pro Unity a musí dojít k posunutí o velikost jednoho terénu. Na obrázku 3 je ukázka mapování projekce pro *zoom* 1 do terénu v Unity.

Důležité je si také uvědomit, že souřadný systém pro výškovou mapu terénu má oproti souřadnému systému obrázků/textur prohozené osy. Bude-li se tedy přistupovat ke staženému obrázku jako k textuře (v Unity *Texture2D*), pak se musí při tvorbě výškové mapy prohodit osy.

Stažená textura výškové mapy je 256×256, ale textury na sebe nenavazují přesně, tzn. nemají stejné výšky na okrajích. Takže kromě toho, že se musejí převzorkovat na 257×257, je potřeba okolních 8 dlaždic, z nichž se získají výšky na okrajích.

2.5.2 Pořadí volání vnitřních funkcí Unity

Mezi jednu z vnitřních funkcí Unity patří metoda *Update*, která se volá každý snímek. Dále sem patří metody *Start*, *Awake*, *OnApplicationQuit*, *OnEnable*, *LateUpdate*, *OnPostRender*, *OnDestroy* a další. Zmiňuji pouze ty, které aktivně využívám, pro kompletní přehled se můžete podívat do manuálu Unity[10].

Při načítání scény se v tomto pořadí zavolá *Awake*, *OnEnable*. Metoda *Awake* se používá pro základní inicializaci proměnných. *OnEnable* je dále volána pokaždé, když se změní stav objektu ve scéně na aktivní. Před prvním snímkem se zavolá metoda *Start*, která slouží podobně jako *Awake*. Rozdíl mezi *Awake* a *Start* je ten, že *Start* se zavolá pouze na vytvořený aktivní objekt, protože není nutné, aby objekt byl ve scéně vytvořen jako aktivní. Tímto způsobem se dá odložit náročnější inicializace na vhodný čas nebo v metodě *Awake* vytvořit reference mezi objekty a ty poté použít v metodě *Start*.

Metody *Update*, *LateUpdate* se vykonávají každý snímek v tomto pořadí. *LateUpdate* zaručuje, že všechny metody *Update* byly dokončeny a hodí se například k implementaci kamery sledující objekt.

Poté, co se vykonají metody *Update*, *LateUpdate*, se provedou metody související s renderováním, z nichž se hodí *OnPostRender*, ve které se dá přímo volat vykreslování čar na grafické kartě.

Zbylé dvě uvedené metody se volají při vypnutí aplikace - *OnApplicationQuit* - nebo při smazání objektu ze scény - *OnDestroy*. Jsou pochopitelně určené ke správě (smazání) alokovaných zdrojů.

2.5.3 Komponentový návrh

Unity je postaveno na komponentovém a objektovém programování, tj. využívá principy dědičnosti, ale nové chování je tvořeno především vytvářením a přidáním nových komponent a málokdy tak vznikají hluboké hierarchie dědičnosti[11]. Naopak častější jsou hierarchie o málo úrovních, ale s velkým množstvím potomků.

Každá komponenta v Unity musí dědit ze základní třídy MonoBehaviour, která definuje všechny funkce zmíněné v předchozí části, např. *Update*, *Start* a další. Budeme-li chtít vytvořit třeba UAV, pak se nejprve vytvoří objekt ve scéně, který sám o sobě nic neobsahuje. Tomu se přiřadí komponenty, které slouží k vykreslování, a aby UAV něco dělalo, je potřeba mu přiřadit komponentu, která ho bude ovládat.

3 Návrh řešení

3.1 Vizualizace ve 3D

Za závažný problém stávající vizualizace považuji její provedení pouze ve dvou rozměrech. Chybí zde vizuální informace nejen o nadmořských výškách, ve kterých se objekty nacházejí, ale i další informace o oblasti, nad kterou se nachází. Simulátor při plánovaní tras využívá informace o nadmořské výšce terénu, která však není nikde zobrazena. Přestože není možné, alespoň prozatím, ze simulátoru informaci o terénu získat, navrhuji využití webových služeb, které takové údaje poskytují. Tyto služby jsem již popsal v sekci 2.2 Webové služby.

Jako další zdroj informací o prostoru, ve kterém se objekty nachází, lze využít satelitních snímků. Tyto snímky sice nemusí být aktuální, ale webové služby se je snaží v rámci možností aktualizovat, ale i tak se dozvíme ty nejdůležitější informace, jako že se nacházíme poblíž lesa, v zástavbě nebo ve volném prostranství na poli. V případě, že si ve stávající simulaci potřebují naplánovat misi, je potřeba, aby si nejprve lokaci našli na mapách, vzali souřadnice, změnili nastavení simulátoru a znovu pustili simulátor. Moje představa je taková, že využitím satelitních snímků přesná lokace nebude potřeba, protože vždy budou vědět, že se UAV zrovna nachází nad polem. Vzhledem k podmínkám použití pro Google Maps, nelze využívat jejich satelitní snímky takovým způsobem, který potřebuji. Využiji tedy sateltiních snímků od Mapboxu, přestože jejich snímky nejsou tak kvalitní jako z Google Maps - na satelitních snímcích se nacházejí mraky a nejsou tak přesné.

V simulaci se pro plánování nepoužívá zástavba. Přesto považuji za vhodné, aby se ve vizualizaci budovy nacházely, už jen proto, aby operátor lépe viděl zástavbu, přestože je možné ji rozpoznat na satelitních snímcích, kde však chybí informace o výšce budov. Jelikož budovy se v simulaci nenachází a UAV by tak mohly startovat uvnitř budov nebo jen prostě mohou překážet ve výběru, tak by zde měla být možnost vypnout a zapnout jejich zobrazení.

3.2 Bezpilotní prostředky

Vizualizace UAV má zásadní problém a tím je velikost bezpilotních prostředků. Vzhledem k velikosti UAV - nejvýše kolem 2 metrů v průměru - a jejich rychlosti je obtížné je sledovat případně vybírat do misí.

Bylo by možné zobrazit informace o vybraných objektech vedle nich ve scéně, tak jak je vidět na obrázku 1. Samozřejmě pak dochází k překrývání informací mezi sebou a také k zakrývání jiných objektů. Pokud by byl vybrán pohyblivý cíl s velkou rychlostí, pak by bylo i obtížné tyto informace vůbec přečíst.

Navrhuji využití billboardů s danou vzdáleností od objektů. Tyto billboardy by se však mohly překrývat a negativně ovliňovat vnímání scény. Výhodou této metody je, že zároveň ukazuje polohu objektu v prostoru.

Druhou možností je vytvořit štítky v prostoru obrazu, které by byly spojené s objekty úsečkou. Toto řešení bych postavil na uspořádání grafu modelováním přitažlivých a odpudivých sil. Vycházel bych ze článku On the Selection of 2D Objects Using External Labeling[12].

Řešení zvolím v závislosti na reakci participantů při testování, ale očekávám, že druhá možnost vyjde lépe právě kvůli překryvům u billboardů. Také je zde možnost zvětšovat objekt v závislosti na vzdálenosti od kamery. Zde by mohly nastat problémy s překryvem a při velkém zvětšení se pohyb špatně vnímá. Navrhuji tedy dát možnost uživateli zvětšování s měnící se vzdáleností vypnout a zapnout podle potřeby.

3.3 Uživatelské rozhraní

Jak jsem již zmínil v předchozí sekci, součástí uživatelského rozhraní musí být štítky označující malé objekty jako jsou UAV. Jelikož nemá smysl stejným způsobem neoznačit i velké oblasti jako bezletové zóny, každý vybratelný objekt ve scéně musí mít vlastní štítek. Pro zjednodušení orientace mezi objekty, navrhuji jejich rozdělení podle barev. Barvy by měly být vizuálně odlišné, a tak navrhuji červené odstíny pro bezletové zóny (červená značí nebezpečí a zákaz), pro sledovanou oblast modrý odstín (sledovaná oblast se promítá na terén a zemský povrch se skládá převážně z odstínů zelené a hnědé, vyjma vodních ploch, nad nimiž však výrobci kvadrokoptér a jiných dronů nedoporučují létat), pro navigační body v trase letu žlutý odstín (jelikož se nachází nad zemí, je možné se na ně dívat směrem na nebe, kde díky žluté barvě bude dobře vidět) a pro pohyblivé objekty jako UAV navrhuji zelenou barvu (budou tak lépe vidět oproti nebi) a pro sledované cíle navrhuji fialovou barvu (pouze proto, aby byla vizuálně odlišná od ostatních). Obdobně odlišuji prolétlou trasu a její predikci. Predikce trasy je modrá, zatímco prolétlá trasa je červená, kde barvy byly zvoleny tak, aby se co nejvíce od sebe odlišovaly.

K osvětlení navrhuji pouze směrové světlo, které vrhá stíny. Stíny se hodí pro zlepšení vnímání tvaru a pozice objektů. Aby stíny nemátly, tak směrové světlo bude mířit kolmo na terén, a tak vznikne stínová projekce objektů ve vzduchu na zem a půjde tak snáze určit poloha při pohledu z boku. Za problémové však považuji odlesky materiálů, kdy materiály pro například bezletové zóny jsou částečně transparentní, aby bylo vidět, co se v nich nebo za nimi děje. Odlesky znepřehledňují a zakrývají informaci za poloprůhlednými objekty, tudíž všechny mé materiály nepoužívají odlesky. Naopak za užitečné považuji zvýrazněné obrysy objektů. Nejenže jsou objekty lépe odlišitelné od terénu, navíc poskytují možnost zvýraznění při výběru objektu.

U poloprůhledných objektů se však hůře vnímá pozice v prostoru, a proto navrhuji pro všechny objekty (i neprůhledné, aby se zachovala stejná konvence pro všechny objekty) spojit jejich střed s jeho průmětem na terén úsečkou.

Co se týče grafického uživatelského rozhraní, je potřeba alespoň jedno vlastní okno pro mise a pro informace o výběru. Mise totiž nejsou objektem ve scéně a tudíž není možené je jakkoli vybrat. Mohlo by se říct, že by stačilo pouze poslat UAV na misi a

nikde ji nezobrazovat. Ale tak by brzy uživatel ztratil přehled o probíhajících misích a neměl by možnost je znova pustit nebo třeba pozastavit. Okno pro informace o výběru souvisí s problémem zmiňovaným u štítků. Informace by nebylo vhodné ponechat přímo u objektu, jak je ve stávající vizualizace, protože by informace byly nečitelné, když by se překryly.

Dále navrhuji dvě další okna: jedno pro zóny a druhé pro agenty. Mezi agenty patří pozemní cíle a UAV, protože mají vlastní inteligenci a pohybují se. Mezi zóny samozřejmě patří bezletové zóny a sledované oblasti, ale také navigační body, které sem patří proto, že svým způsobem tvoří kruhovou zónu kolem bodu v prostoru, kterou musí UAV proletět. Pro tyto okna by se však hodila možnost zda-li je zobrazovat nebo ne.

Pro prvky grafického uživatelského rozhraní považuji za vhodné, aby se jejich velikost neměnila v závislosti na rozlišení obrazovky, tj. stálá fyzická velikost v pixelech, protože by tak zmizela výhoda větších displejů, ty by takto měly mít více volného místa. Zároveň by se hodilo, kdyby se dala okna pozicovat a měnit jejich velikost.

Obrázek 4 Uživatelské rozhraní se všemi okny zobrazenými při rozlišení 700×315 *px*

Součástí uživatelského rozhraní by mělo být i informování o stavu aplikace, tj. není připojen simulátor k vizualizaci, UAV prolétlo bezletovou zónou, přestože nemělo apod. Na obrázku 4 je možné vidět upozornění na nepřipojení k simulátoru.

3.4 Kamery

Navrhuji použití čtyř kamer, všechny perspektivní. Jedna kamera by měl být plně volný pohyb po scéně a měla by umožňovat rotaci. Druhá kamera by byl pohled shora bez možnosti rotace a jako třetí by byla kamera, která je umístěna pod objektem a umožňuje rotaci, ale relativní vzhledem ke směru natočení objektu. Čtvrtá kamera je kamera z pohledu chodce.

Pohled shora by měl zjednodušit vytváření zón, jelikož souřadné osy terénu budou odpovídat souřadnicím (okrajům) obrazovky, a tím ulehčit vnímání vzdáleností a pozic. Volná kamera je samozřejmě určena především k pohybu ve scéně přesnějšímu zaměření se na jednotlivé objekty či pohled z jiného úhlu. Kamera z pohledu objektu je určena především pro UAV, kdy se uživatel může podívat z pohledu bezpilotního prostředku.

Kamera chodce by měla sloužit k pohledu z pozice operátora, aby si naplánoval trasu tak, aby mu UAV neodletělo z dohledu třeba za horizont.

3.5 Počítání v pohyblivé řádové čárce

Simulátor pracuje se souřadnicemi s dvojitou přesností (*double*) a to proto, že je zde důležitá především přesnost. To pro vizualizaci neplatí. Tato přesnost je v milimetrech a moje simulace by neměla žádné nové stavy UAV vytvářet, takže nevadí, jestli bude UAV umístěn o pár milimetrů jinde, než podle simulátoru je. Proto budu využívat pro ukládání pozic pouze jednoduchou přesnost ($float$), který je navíc použit pro pozicování v Unity.

Pozice v jednoduché pohyblivé čárce může způsobovat problémy při transformaci GPS souřadnic, kdy poloha z úhlů je přepočtena na metry. Takové hodnoty nedokáže jednoduchá pohyblivá čárka zachytit všechny, především měly-li by pokrývat celou planetu. Proto navrhuji využití relativních pozic ke středu světa, kdy střed bude vytvořen na libovolné GPS souřadnici a všechny ostatní pozice se budou odvíjet od ní.

3.6 Svět

Střed světa by se tedy měl nacházet v počátku souřadné soustavy a měl by se skládat ze čtvercových dlaždic s relativní pozicí vzhledem ke středu světa. Každá dlaždice má svůj vlastní satelitní snímek a výškovou mapu a na ni patřící budovy. Navrhuji použít data z úrovně 15 a 16, kdy přesnost na jeden pixel bude menší než 3 metry, viz. sekce 2.2.1 Mercatorova projekce. Kdy větší důraz kladu na přesnost satelitních snímků, protože dlaždice terénu nebude mít pro každý pixel textury vlastní vrchol v 3D trojúhelníkové síti, a tak dojde ke ztrátě detailnější informace, přičemž se zlepší výkon.

4 Implementace

Tato kapitola se zabývá konkrétními implementačními detaily, mezi které patří i způsob generování světa, stahování dat z webových služeb nebo komunikace mezi simulátorem a vizualizací.

4.1 Unity

4.1.1 Paralelní exekuce

Generování terénu i budov nebo stahování obrázků z internetu je pomalé, tak by bylo vhodné, kdyby bylo možné tyto náročné operace provádět v jiném vlákně. Unity je ale herní engine, kdy veškeré operace běží pouze v jednom vlákně a grafický kontext také existuje pouze v tomto jediném vlákně. Jakékoli operace nad objekty vytvořené v Unity je tedy potřeba volat pouze z hlavního vlákna Unity.

Každý objekt v Unity, který potřebuje každý snímek měnit své hodnoty nebo něco vypočítat, potřebuje metodu *Update*, která je volána každý vykreslovaný snímek. Kdyby se však v této metodě vždy prováděly náročné operace, tak se celá vizualizace zastaví na jednom snímku a je potřeba počkat než se dokončí. Proto existuje konstrukt nazývaný *Coroutine*, který umožňuje rozložit práci přes několik snímků tím, že uloží svůj stav a pozastaví své vykonávání a později pokračuje od uloženého stavu.

Coroutine je v základu funkce, která vrací *IEnumerator* a obsahuje příkaz *yield return*[13]. *Coroutine* se spustí, když se zavolá *StartCoroutine* s názvem potřebné funkce a polem případných argumentů funce. Stejně jako vlákno, coroutine nezaručuje pořadí volání ani dokončení.

Coroutine však není novým vláknem, stále se vykonává v rámci hlavního vlákna Unity. Pokud bychom tedy potřebovali spustit výpočet v samostatném vlákně, musí se oddělit akce, která ovlivňuje stav objektů v Unity od výpočtu. Jestliže je potřeba výpočet promítnout zpět do hlavního vlákna, je potřeba pravidelně kontrolovat, jestli samostatné vlákno doběhlo, a pokud ano, může se provést požadovaná akce. Čekání opět musí být ve funkci *Update*. *Coroutine* nezaručuje, že pokud se zavolá první, pak také první skončí.

```
private IEnumerator WaitForData(Vector2i tilePos, System.Action<bool>
    callback)
{
   // Wait on data download around tilePos and then execute callback
}
```

Ukázka 4.1 Ukázka použití coroutine

```
StartCoroutine(WaitForData(tilePos, (success) =>
   {
      if (success)
      {
         // Create terrain tile
      }
   }
));
```

Ukázka 4.2 Ukázka spuštění coroutine s lambda funkcí jako parametrem

Nevýhodou *Coroutine* je, že stále pracují pouze na jednom vlákně a nejsou stavěné na zpracování událostí. Textury v Unity se navíc okamžitě po vytvoření nahrávají na grafickou kartu a to zabírá nějaký čas. Při stažení přes API Unity (třída *WWW*) se okamžitě tvoří textura, a tak dochází k znatelnému poklesu počtu snímků za vteřinu (dále již FPS). *Coroutine* tedy využívám k animacím rozloženým přes několik snímků a k čekání na stažení potřebných okolních výškových map.

4.2 Generování světa

Svět, ve kterém UAV létají, se skládá ze čtvercových dlaždic. Tyto dlaždice jsou vytvořeny jako Unity *Terrain* s výškovou mapou získanou od Mapboxu s texturou satelitních snímků získaných taktéž od Mapboxu. Svět je generován na základě GPS souřadnic, které jsou stejně jako api-klíče v souboru world.txt. Tato GPS souřadnice se stane středem světa, tj. počáteční dlaždici s relativními souřadnicemi $(0, 0)$. Střed světa je levým dolním rohem umístěn v počátku souřadného systému Unity. Terén se generuje v kruhovém rozsahu daném uživatelským nastavením v aplikaci.

4.2.1 World.txt

Tento soubor slouží ke specifikování počátečních GPS souřadnic (ve tvaru zeměpisná šířka a délka) a také obsahuje Vaše api-klíče.

MAPZEN_KEY ?api_key=mapzen-xxxxxx

MAPBOX_KEY ?access_token=xxxxxx

GPS 49.6209556586354 16.1247969873662

IP IP adresa v4

Klíč je od hodnoty oddělen 1 mezerou a stejně tak jsou rozděleny i hodnoty, pokud je pro klíč více jak jedna. Pokud simulátor beží na stejném stroji jako vizualizace, doporučuji záznam IP odstranit, a tak se použije adresa na *localhost*.

Pomocí těchto algoritmů lze dopočítat, o kolik se bude lišit velikost dlaždice po 10 km. Jako počátek jsem zvolil souřadnice 49.61784° severní šířky a 16.11694° východní délky s rozlišením 3.09514 metrů na pixel. Po 10 km na sever a východ jsou souřadnice 49.70758° severní šířky a 16.2556° východní délky s rozlišením 3.089436 metrů na pixel. Z toho plyne, že po 10 km se dlaždice zmenší o plných 1.46 metrů.

Data: (x,y,z) pozice ve světě relativně k počátku
Result: GPS souřadnice ve tvaru zeměpisná šířka, délka

1 $(x_1, y_1) = (x, z) + Souřadnice\ počátku\ světa$;
2 $y_1 += -y_1 + Velikost\ dlaždice \times\ Rozlišení\ dlaždice\ na\ pixel$;
3 $(x_1, y_1) /= Rozlišení\ dlaždice\ na\ pixel \times 2^{zoom}$;
4 **return** *Aplikuj inverzi vzorce* 1 *na* (x_1, y_1);

Algoritmus 1: Výpočet pozice gps souřadnic z pozice ve světě.

Data: GPS souřadnice ve tvaru zeměpisná šířka, délka
Result: (x,z) pozice ve světě relativně k počátku

1 $(x_1, z_1) = Promítni\ GPS\ souřadnice\ pomocí\ vzorce1$;
2 $z_1 += -z_1 + Velikost\ dlaždice \times Rozlišení\ dlaždice\ na\ pixel$;
3 $(x_1, z_1) -= Počátek\ světa$;
4 **return** (x_1, z_1) 1;

Algoritmus 2: Výpočet pozice ve světě z GPS souřadnic.

Data: (x,y,z) pozice ve světě relativně k počátku
Result: Souřadnice dlaždice v Mercatorově projekci

1 $(x_1, y_1) = (x, z) + Souřadnice\ počátku\ světa$;
2 $y_1 += -y_1 + Velikost\ dlaždice \times Rozlišení\ dlaždice\ na\ pixel$;
3 $(x_1, y_1) /= Rozlišení\ dlaždice\ na\ pixel \times Velikost\ dlaždice$;
4 **return** $(x_1, y_1) zaokrouhlené\ dol$ů;

Algoritmus 3: Výpočet souřadnic dlaždice z relativní pozice ve světě.

4.2.2 Dlaždice terénu

Již v kapitole o webových službách bylo zmíněno, s jakou přesnost na pixel mají výškové při dané hodnotě *zoom*. Pro zopakování, pro Českou republiku můžeme říci, že se tato přesnost pohybuje kolem 3 $[\frac{m}{px}]$. Tuto hodnotu považuji za dostatečnou, vzhledem k tomu, že málokdy se bude uživatel pohybovat přímo u země a navíc terén v Unity využívá dynamického LOD a s velkou pravděpodobností stejně nebude využívat veškerou dostupnou přesnost. Problém však nastává se satelitními snímky. Pro přehlednost a míru detailů jsem se rozhodl vzít satelitní snímky z úrovně vyšší.

Jelikož na terén máme 4 textury a jen jednu výškovou mapu, po stažení se výšková mapa nejdříve převzorkuje, a poté rozdělí na 4 části, z nichž se teprve vygenerují 4 dlaždice terénu. Toto rozhodnutí také pomohlo zrychlit generování, jelikož je možné rozdělit tvorbu každé části do vlastního snímku a nedojde k takovému poklesu FPS. Pro referenci: generování terénu s rozměry 257×257 zabere v průměru 200 ms a terén 129×129 pouze 50 ms.

Jelikož terén v Unity vyžaduje rozlišení výškové mapy $2^n + 1$ a stažená výšková mapa je rozměru 256×256, musíme ji převzorkovat. Takto vzniklé terény na sebe ale nebudou pasovat a to proto, že okraje výškových map se nepřekrývají. Dá se také říct, že mezi nimi bude chybět jeden pás trojúhelníků spojující hrany terénů. Aby se tomuto předešlo, tak pro tvorbu jednoho terénu je potřeba znát i ostatních 8 okolních výškových map a s

využitím těchto map se interpolací dopočítají hraniční hodnoty, aby výšky sousedících stran map měly stejnou hodnotu. Tak se zbavíme na sebe nepasujících okrajů. Dalším důvodem pro čekání na okolní výškové mapy je, že nedochází k opakovanému upravování terénu, protože veškerá data jsou již dostupná.

Po vytvoření upravené výškové mapy ji rozdělíme na 4 části a z nich pak vytvoříme vlastní dlaždici terénu, kterému se přiřadí stažený satelitní snímek. Je důležité pro texturu satelitních snímků nastavit *wrap mode* na *clamp*, jinak by docházelo na okrajích kvůli nepřesnostem floatu a bilineární interpolace k získávání barev s opačné strany, což je především viditelné na místech, kde například pole přechází do lesa.

4.2.3 Budovy

Pro každou dlaždici se stáhnou informace o budovách a využití půdy (viz. ukázka 2.2.4 a ukázka 2.2.4). Z dostupných dat se využívají pouze GPS souřadnice rohů budov a jejich výška. V případě, že výška není definována, pak je použita výška 15 metrů. Budovy jsou navíc protáhlé další dva metry pod úroveň terénu, aby se nestalo, že přestože rohy budou na zemi, tak nějaká hrana zůstane ve vzduchu nad proláklinou v zemi. Každá budova má pouze jednoduchou rovnou střechu vytvořenou triangulací pomocí algoritmu ořezávání uší[14] a to jeho nejjednodušší variantou, která může běžet až $O(n^3)$, což nemá vzhledem k počtu rohů budov příliš význam. I nekonveční budovy budou mít nejvíce kolem několika desítek rohů. Navíc, pokud má budova složitý půdorys, pak je rozložena do základních stavebních částí, např. do 5 obdelníků, které se mohou i překrývat.

Pro dlouhé objekty jako zdi (např. Vyšehradské hradby) se umisťuje každý roh zvlášť, zatímco u budov se pouze najde výška středu budovy, vzhledem k dnes populární praxi, kterou je stavění na úrovnané ploše. Hledání nadmořské výšky rohu/středu budovy se provádí vrháním paprsku na terén (jako naprostá většina získávání výšek). Tím se však u budov může stát, že budova ležící mimo terén není umístěna do správné nadmořské výšky. Takové bývají budovy, které se nacházejí na rozmezí jednotlivých dlaždic, přestože se to stává velmi zřídka, jelikož se nejprve generuje terén a až poté tvoří budovy.

4.2.4 Třída ThreadedJob

Abych odstranil problémy se stahováním na jednom vlákně, vytvořil jsem třídu ThreadedJob, která spustí funkci ve vlastním vlákně a po dokončení této nezávislé akce, se zavolá funkce, která se již provede na hlavním vlákně Unity. Třída potřebuje pravidelně kontrolovat, jestli se akce ve vlákně již dokončila. Tato kontrola lze provádět právě pomocí *Coroutine* nebo v *Update*.

PopulateTileJob

PopulateTileJob slouží ke stažení dat z webových služeb a každé stahování - 4 pro satelitní snímky, 1 pro budovy a 1 pro výškové mapy - probíhá asynchronně. Stáhnutá data se mohou uložit na disk, aby se při příštím spuštění zkrátila doba načítání.

V rámci stahování výškové mapy také dojde k jejímu převzorkování. Tvorba textur, již prováděna v hlavním vlákně, je rozložena přes několik snímků, aby nedocházelo k

přílišné zátěži hlavního vlákna Unity. Po dokončení se vyvolá událost, která informuje o dokončení stahování dat pro danou dlaždici.

TileCreationJob

TileCreationJob se spustí po stažení všech potřebných dat, viz. ukázka 4.1.1. Nepjrve se sešijí okraje výškových map, aby hodnoty na krajích odpovídaly, a pak se výšková mapa rozdělí na 4 menší. Po dokončení úprav výškové mapy se generuje samotný terén. Jeho tvorba je rozložena do 4 snímků, protože při využití terénu v Unity je medián doby jeho tvorby 40 ms a nejvyšší hodnoty se pohybují kolem 200 ms. Pokud by se tedy vytvářel terén naráz, došlo by k znatelnému poklesu FPS. Po úspěšném vygenerování terénu se zavolá vytváření budov.

BuildingCreationJob

BuildingCreationJob se spouští dvakrát: jednou pro budovy a jednou pro využití půdy. Ve vlastním vlákně se zpracuje GeoJSON s daty a vygenerují se vrcholy a trojúhelníky pro budovy, případně zdi. Ty se pak přiřadí v hlavním vlákně Unity.

4.2.5 Třída TileCache

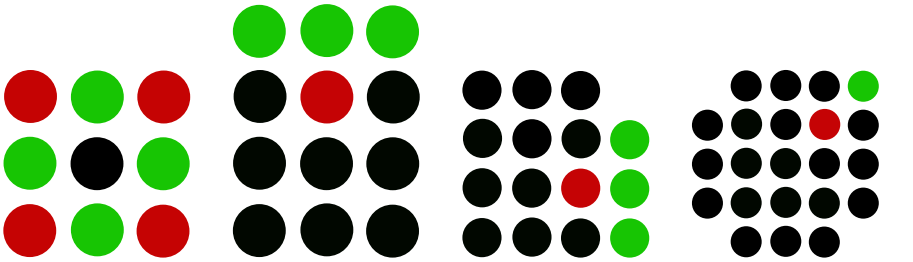
Od struktury, která by obsahovala dlaždice, potřebuji především rychlou kontrolu, zda-li je daná dlaždice již vytvořena nebo se právě vytváří.

Souřadnice dlaždic jsou závislé na úrovni, z které se berou. Používám-li úroveň 15, pak dlaždice má obě souřadnice nejvýše 2^{15}. Z toho se velmi snadno vytvoří jednoznačná hashovací funkce, která vezme souřadnici x, posune ji o 16 bitů vlevo a přičte se souřadnice y. Za využití této hashovací funkce, můžeme pro ukládání dlaždic využít hash mapu, která splňuje moje požadavky na rychlý přístup - v čase O(1).

Třída TileCache obsahuje hash mapy právě pro stažené dlaždice, stažená data, data právě se stahující a data, která na stáhnutí teprve čekají. Dlaždice se může vytvořit pouze v případě, že ještě nebyla vytvořena a zároveň o její vytvoření není zažádáno. Pokud lze vytvořit, tak se označí jako čekající na stáhnutí, zažádá se o 8 dlaždic okolo a dlaždice se přidá do seznamu čekajících na zpracování. Žádosti o vytvoření dlaždice se zpracovávají po jednom vzhledem k nejmenšímu vlivu na FPS a zpracovávané dlaždice jsou označeny jako právě se stahující. Když se stáhnou data pro dlaždici, označí se data jako stažená, a v případě že jsou všechny okolní dlaždice stažené, vytvoří se terén a dlaždice je označena jako vytvořená.

4.2.6 Stahování dlaždic

Dlaždice se stahují v závislosti na pozici kamery. Generují se postupně od středové dlaždice a to po směru hodinových ručiček - nejprve levá, pak horní, pravá a dolní. Po dokončení těchto čtyř se začne levou horní a dále po směru hodinových ručiček. Pokaždé, když se vygeneruje dlaždice, uloží se do fronty, z které se po jednom odebírá a terén se přestane generovat, jakmile se fronta vyprázdní. Tímto způsobem je kromě první dlaždice potřeba stáhnout nejvýše 3 další okolní dlaždice a v případě rohových pouze 1, čím se generování stane plynulejší a rychlejší.



a) Generování dlaždic. Nejprve zelné, a pak červené a to po směru hodinových ručiček.

b) Potřeba stáhnout zelené dlaždice pro horní čerený bod.

c) Potřeba stáhnout zelené dlaždice pro pravý červený bod.

d) Pro rohový bod zbývá stáhnout pouze 1 dlaždice.

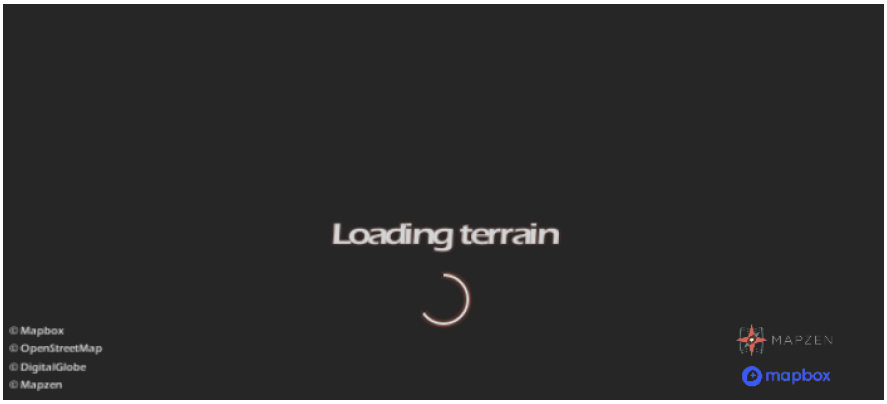
Obrázek 5 Postupné generování dlaždic.

4.3 Objekty spjaté se simulací

V této sekci se zabývám popisem objektů, které uživatel může vytvořit nebo které vznikají na základě informací poskytnutých *AgentFly* simulátorem. Patří sem například bezpilotní prostředky, bezletové zóny nebo trajektorie.

4.3.1 Trajektorie

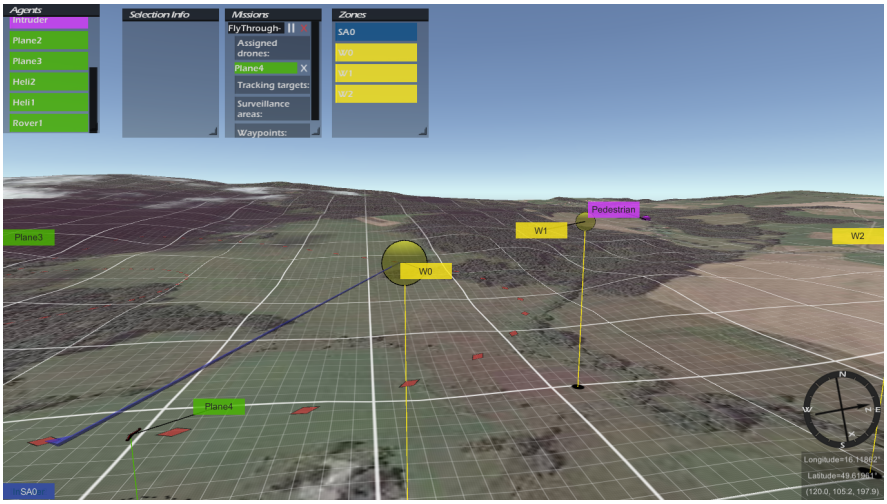
Jelikož ze simulátoru přicházejí pozice relativní vzhledem k výšce terénu (viz. 2.3), je potřeba jakékoli body nejprve posunout v závislosti na dané výšce terénu. Pozici získávám vrháním paprsku ve směru projekce bodu na terén, z čehož plyne podmínka, že takový terén musí existovat. Existenci počátečního terénu zajišťuji tím, že komunikace se simulátorem je navázána až poté, co se vygeneruje terén okolo počáteční lokace. Během této doby má uživatel přístup pouze do nápovědy a než se vytvoří první dlaždice je zobrazena načítací obrazovka, kde se i nachází loga a seznam použitých služeb. Další terén se pak generuje zároveň s pohybem uživatele, aniž by došlo k omezení ovládání.



Obrázek 6 Obrazovka před vytvořením první dlaždice.

Prolétlá trajektorie se získává ukládáním pozice pokaždé, když ze simulátoru přijde nová aktuální pozice UAV, tedy každých 500 ms. Prolétlá trajektorie je tvořena jednotlivými body reprezentující pozici UAV a je označena červenou barvou. Z rozestupů

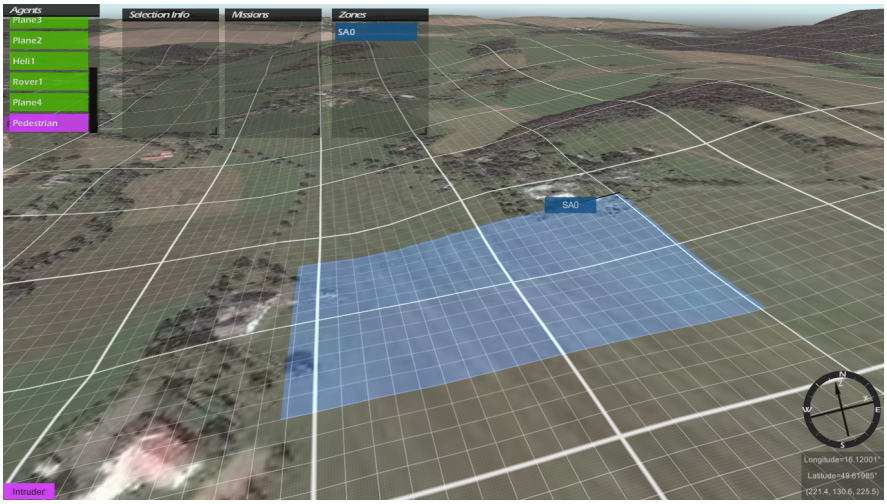
bodů tedy lze odhadnout jak rychle v porovnání s ostatními UAV se pohybuje. Predikce trajektorie je označenou barvou modrou a má tvar kvádru zakončeného jehlany.



Obrázek 7 Ukázka trajektorií. Prolétlá červeně a predikce modře. UAV míří k navigačnímu bodu W0.

4.3.2 Zóny

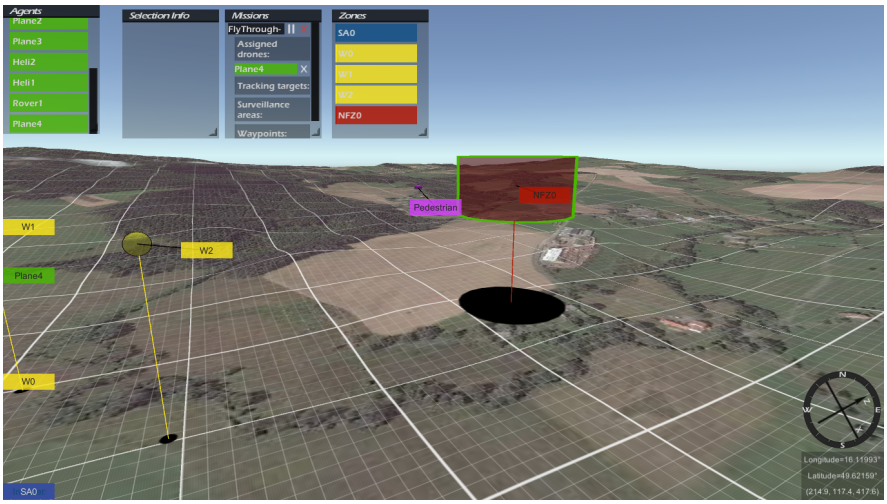
Sledovaná oblast je řešena jako obdelníkový projektor modré barvy. Projektor je kamera, která aplikuje přiřazenou texturu na všechny objekty, které jsou v ní viditelné. Využívám tedy kameru ortografickou, u které lze pomocí jejího aspektu a šířky projekce přesně nastavit, jakou plochu bude zabírat. Pro texturu, kterou kamera promítá, platí, že nesmí mít nastavené opakování textury a že je potřeba nechat alespoň jeden pixel na okraji transparentní, jinak by se textura promítla na celý objekt, kterého se kamera dotýká a ne jen na jeho viditelnou část.



Obrázek 8 Modře zbarvená sledovaná oblast vytvořená pomocí projektoru.

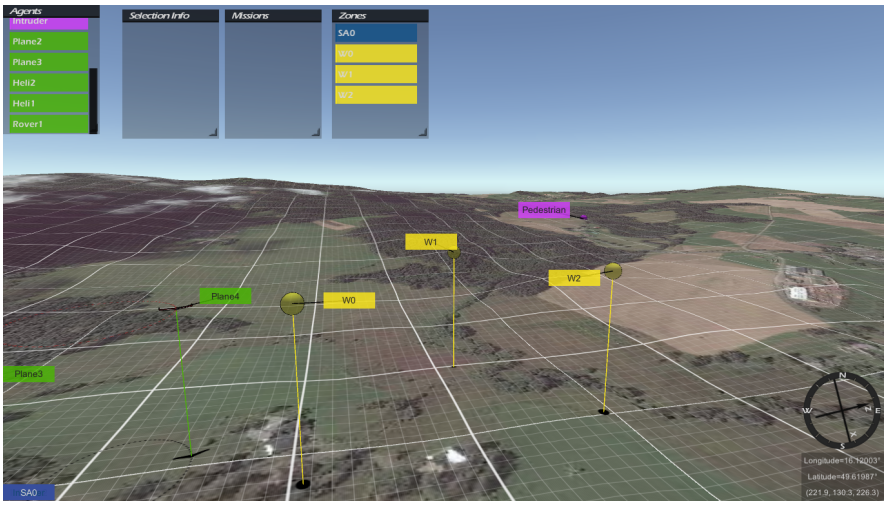
Bezletová zóna je červený poloprůhledný válec, jlikož simulátor momentálně neumožňuje vytvoření jiné než válcovité bezletové zóny. Pro to, aby se bezletová zóna promítla

do letového plána v simulátoru, je potřeba znovu spustit misi, ve která jsou UAV, která onou bezletovou zónou mají být ovlivněna. Tento požadavek je dán vlastností simulace a já ho nemůžu ovlivnit.



Obrázek 9 Červená poloprůhledná válcovitá bezletová zóna.

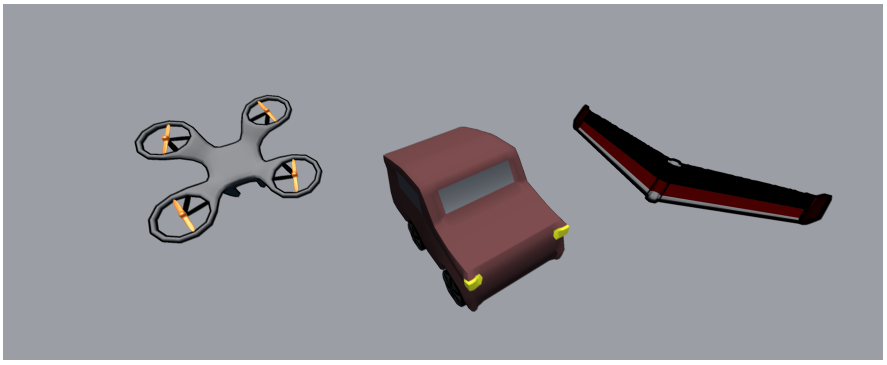
Navigační bod také spadá pod zóny z důvodu, který jsem již vysvětloval v kapitole 2 Návrh řešení, tj. navigační bod tvoří oblast (zónu) s daným poloměrem, kterou UAV musí proletět. Tento poloměr lze nastavit v konfiguraci simulace, avšak tuto informaci se já nedozvím, takže navigační body ve vizualizaci mají stále stejnou velikost a nemění se, ani když je povoleno škálování vzhledem k vzdálenosti kamery.



Obrázek 10 Žlutě zbarvený navigační bod. Všimněte si, že je spojen se svojí projekcí na terén.

4.3.3 Agenti

Za agenty označuju všechny objekty, které jsou řízené simulací a mohou se pohybovat. Sem tedy patří bezpilotní prostředky jako křídla, kvadrokoptéry a vozidla. Bezpilotní prostředky mají svůj vlastní model a mají různé zbarvení.



Obrázek 11 Kvadrokoptéra s průměrem 70 cm, křídlo s rozpětím 135 cm a jednoduchý automobil se šířkou 2 m a délkou 3,5 m. Poměry velikostí nejsou na obrázku zachovány.

4.4 Třída Pickable

Třída *Pickable* slouží pro sjednocení ovládání všech objektů, které lze ve scéně vybrat. Obsahuje funkce na přidání štítků, na manipulaci při výběru a vyvolává události, které informují, jestli se objekt ve scéně zrovna nachází před kamerou.

4.4.1 Třída Agent a třída Zone

Z těchto tříd dědí třídy GroundTarget, Drone a NoFlyZone, SurveillanceArea, TargetWaypoint v tomto pořadí. Každá z nich slouží k obsluze jednoho z objektů definovaných v síťové komunikaci - pozemní cíl, UAV, bezletoví zóna, sledovaná oblast a navigační bod.

4.5 Štítky objektů

Každá instance třídy Pickable má vlastní štítek (třída Label), který tvoří vrchol grafu. Tento graf se skládá z koncových bodů a kotev, které spojují instanci Pickable s jejím štítkem. Tento graf je pozicován podle působících odpudivých a přitažlivých sil. Výpočet působení těchto sil probíhá ve třídě LabelManager. V případě, že se objekty nachází na obrazovce, pak výpočet probíhá pomocí algoritmu 4.

Algoritmus 4 je založen na řešení [12]. Od autorů se odlišuji v tom, že nepovažuji velikost štítků za zanedbatelnou a možnou nahrazení jedním bodem. Proto jsou moje působící síly dvourozměrné, kdy na každou osu působí jiná síla, čímž dochází k méně překryvům mezi štítky. Dále jsem přidal sílu, která přitahuje štítky k pozici myši, což slouží ke snažšímu výběru, protože do určité vzdálenosti jsou štítky přitahovány k myši a je tak snažší je vybrat, protože se neustále nebudou pohybovat ve směru pohybu UAV.

Pro objekty mimo obrazovku probíhá pouze zjednodušený výpočet a to takový, že se vypočítá pouze přitažlivá síla ke kotvám. Štítky se drží na okraji obrazovky a mohou se i překrývat. Jsou zde proto, aby objekty, které jsou po stranách kamery, měly alespoň určený směr, kterým se nacházejí. Je důležité poznamenat, že aplikování působících sil na objekty ve scéně by se mělo provést, až po doběhnutí všech iterací algoritmu a to z toho důvodu, že při každém přiřazení pozice dojde k přepočtu hranic pro prvky Unity *Canvas*, který slouží k vykreslování uživatelského rozhraní.

Data: pozice středů štítků a pozice odpovídajících kotev pro objekty na obrazovce
Result: síly působící na koncové body

```
while neproběhl dostatečný počet kroků do
    for objekt na obrazovce do
        if objekt je vybrán then
            pozice se nemění, pokračuj dál;
        else
            F1 = výpočítej přitažlivou sílu koncového bodu ke kotvě;
            F2 = součet odpudivých sil mezi koncovými body navzájem;
            F3 = součet odpudivých sil mezi koncovým bodem a všemi kotvami;
            F4 = vypočítej přitažlivou sílu k středu obrazovky;
            F5 = vypočítej přitažlivou sílu koncového bodu k jeho předchozí
             lokaci;
            F6 = vypočti přitažlivou sílu k pozici myši;
            přičti součet všech sil k síle působící na objekt;
        end
    end
    aplikuj síly působící na koncový bod;
end
```

Algoritmus 4: Výpočet sil působících na body v grafu.

4.6 Třídy pro správu jednotlivých součástí programu

Všechny tyto třídy jsou implementovány jako návrhový vzor singleton. Patří sem GuiManager, AgentManager, ZoneManager, SimulatorClient a již zmíněný LabelManager. Každá z těchto tříd zprostředkovává komunikaci mezi souvisejícími třídami a mezi sebou navzájem. GuiManager má na starosti uživatelské rozhraní, AgentManager obsluhuje UAV, pozemní cíle, trajektorie a mise, ZoneManager ukládá sledované oblasti, bezletové zóny a navigační body a SimulatorClient poskytuje komunikaci po TCP se simulátorem.

4.7 Kamera

Kamera je tvořena skriptem PickerCamera, kterým se obsluhuje uživatelský vstup. Všechny operace nad kamerou jsou implementovány v *LateUpdate*, kdy je vypočítána pozice všech objektů. Tímto způsobem lze předejít poskakování kamery při sledování cíle. Kamera má k sobě připevněný *CharacterController*, který umoňuje pohybovat objektem v rámci fyzických možností a nestane se tak, že kamera vletí do budovy. Součástí *CharacterController* je také obálka v podobě kapsule. Zde nastává problém: pokud by kamera byla nastavena z pozice UAV, které má také obálku, dojde k jejich kolizi a fyzikální model v Unity se bude snažit pozici srovnat, a tak dojde k poskakování kamery. Tomuto lze předejít dočasným vypnutím obálky pro *CharacterController* v rámci výpočtu pozice kamery.

Kamera se může nacházet v jednom ze čtyř stavů - *WalkerState* pro kameru z pohledu chodce, *CompassState* pro pohled z UAV, *TopState* pro pohled shora a *FreeState* umožňující volný pohyb.

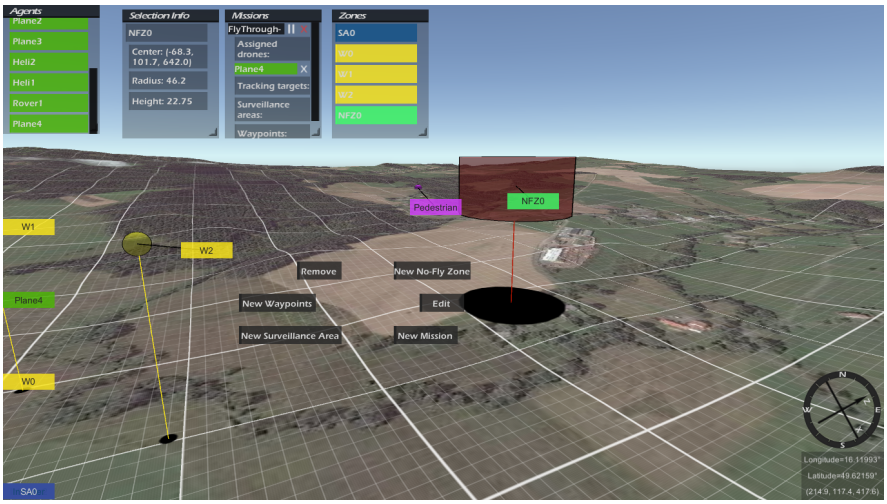
4.8 Logování

K zobrazování událostí ve vizualizaci slouží třída Logger. Ta umožňuje z jakéhokoli vlákna vytvořit zprávu a poté ji zobrazit. Avšak vytváření objektů ve scéně, kterým taková zpráva je, lze volat pouze z hlavního vlákna Unity, což není možné například z vlákna určeného pro síťovou komunikaci. Proto se zprávy přidávají do seznamu, který se při každém volání *Update* vyprázdní a zprávy se zobrazí.

Zprávy jsou rozděleny podle závažnosti na informace (zelené, zobrazené 5 s), varování (oranžová, zobrazené 10 s) a chyby, které jsou červené a mohou být zobrazeny 15 s nebo dokud se nezruší. Každá zpráva lze vytvořit dvěma způsoby: první je pouze předáním řetězce znaků pro zobrazení, taková zpráva nelze ze skriptu zrušit ani obnovit, druhou možností je vytvořit objekt Message, který lze znovu používat a zprávu lze ze skriptu zrušit nebo obnovit.

4.9 Vytváření objektů

Vytváření objektů je možné z kontextového menu, které lze zobrazit stisknutím klávesy Alt. Menu se zobrazí kolem stávající pozice myši. Možnosti Edit a Remove jsou v menu vidět, pouze pokud je vybrán alespoň jeden objekt, který je možno upravit nebo smazat.



Obrázek 12 Kontextové menu v okolí myši při vybrané bezletové zóně (světle zelený štítek).

Každý objekt, který je možno vytvořit, má vlastní třídu, která zpracovává jeho vytvoření - *NFZCreator*, *SACreator*, *MissionCreator* a *WaypointsCreator*. Každý z těchto tříd má kontrolu, zda-li jsou vstupní pole správně vyplněna, a pokud ne, tak se označí červeným obrysem. Vytváření lze ukončit klávesou Escape nebo dokončit klávesou Enter. Každé vstupní pole, jež obsahuje číselné hodnoty, lze měnit kliknutím levého tlačítka myši a tahem vlevo a vpravo. Pokud je při vytváření potřeba zobrazit nadmořská výška, pak je zde uvedena jak od moře, tak relativní od země.

4.9.1 MissionCreator

Slouží k vytváření misí, které vyžaduje pouze zadání jména. Do tohoto dialogu pro vytvoření mise se lze dostat i přetažením jednoho objektu na druhý, kdy nezáleží, jestliže to bylo v uživatelském rozhraní nebo ze štítků. Po zadání jména se automaticky vytvoří mise s přiřazenými objekty, které během vytváření byly vybrány. Jméno mise se automaticky generuje v závislosti na výběru nebo se použije jméno *Mission* s přidaným unikátním číslem.

4.9.2 WaypointsCreator

Jak již množné číslo v anglickém názvu naznačuje, lze je možno vytvořit libovolné množství bodů najednou, kdy nově vytvořený bod sdílí nadmořskou výšku s předchozím bodem. Každý nově vytvořený bod je unikátně pojmenován. Pokud bychom chtěli změnit jeden již vytvořený bod, lze ho kliknutím na něj znovu vybrat.

Nový bod se vytváří kliknutím na terén. Pokud klikneme na již existující navigační bod a nepustíme levé tlačítko myši, můžeme s tímto bodem pohybovat. Při pohybu je zachována nadmořská výška. Změna nadmořské výšky lze provést držením klávesy Shift a levého tlačítka myši a táhnutím po obrazovce směrem nahoru či dolů.

4.9.3 SACreator

Vytváří obdelníkovou sledovanou oblast, k jejímuž vytvoření je potřeba její jméno, které se také generuje automaticky, a pozici počátečního rohu a koncového rohu. Sledovaná oblast se vytváří postupně, nejprve se potřebuje dvakrát kliknout na terén, čímž se vytvoří pomocné body, kterými lze kliknutím a táhnutím oblast měnit.

4.9.4 NFZCreator

Posledním objektem, který lze vytvořit je válcovitá bezletová zóna, která potřebuje jméno, taktéž automaticky generované, pozici středu, poloměr a výšku. Při aplikování poloměru a výšky je nutné si uvědomit, že základní válec v Unity má poloměr 0.5m a výšku 2m. Tedy při změně měřítka se musí výška dělit 2 a poloměr naopak 2 násobit.

K zjednodušení vytváření bezletové zóny existují tři pomocné body, kterými lze táhnutím měnit pozice středu zóny, její poloměr a výška. Nadmořská výška se mění stejně jako u navigačních bodů a to držením tlačítka Shift a levého tlačítka myši a táhnutím nahoru/dolů.

4.10 Ovládání

Důležité je říci, že pokud se vyplňuje nějaké vstupní pole (jméno zóny, pozice), pak je vypnutý pohyb kamery, aby se zabránilo pohybu zbytečnému kamery při psaní. Také najede-li uživatel myší nad uživatelské rozhraní, nelze vybírat objekty ve scéně.

Ovládání je rozděleno mezi myš a klávesnici. Na myši lze použít pravé tlačítko myši k rotaci kamery, levé tlačítko k výběru nebo při tvorbě a prostřední tlačítko při zmáčknutí umožňuje pohyb do stran a při rolování se mění vzdálenost od body, kam ukazuje myš, a tato vzdálenost také určuje, jak rychle se kamera bude přibližovat/vzdalovat. Vybírání lze provést s držením klávesy Shift, čímž lze vybrat více jak jeden objekt.

a) Vytváření navigačního bodu.

b) Vytváření sledované oblast.i

c) Vytváření bezletové zóny.

Obrázek 13 Přehled vytváření objektů.

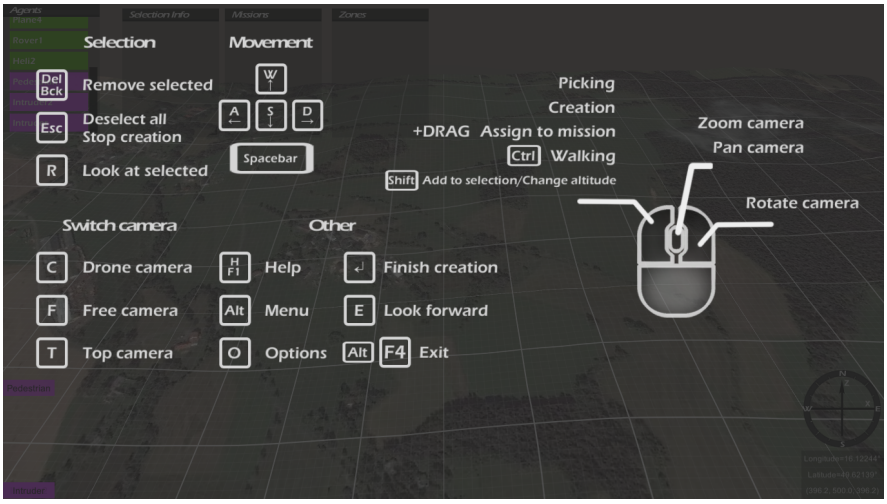
Klávesa Escape slouží k odebrání všech objektů z výběru, smazat objekty (pokud je to dovoleno) lze klávesami Delete nebo Backspace a kontextové menu lze zobrazit pomocí klávesy Alt. Nápověda se zobrazuje stisknutím H/F1 a nastavení pomocí O. Mezi kamerami je možné přepínat pomocí kláves C, F, T pro pohled z kamery UAV, volné kamery a kamery shora. Pro přepnutí do řežimu chodce je potřeba držet Ctrl a kliknout na terén, po čemž se kamera přesune na pozici červeného bodu pod kursorem myši. Klávesou R se lze zaměřit na vybraný objekt. Dále pro pohyb v slouží klávesy W,A,S,D, šipky a mezerník, který slouží k pohybu nahoru a dolů (pokud se drží Shift). Při pohledu z kamery UAV je možné se otáčet libovolným směrem, a kdyby se uživatel chtěl natočit směrem, kam UAV míří, může stisknout E.

Během načítání, kdy je zobrazen nápis Loading screen, není možná žádná interakce s vizualicí, kromě klávesové zkratky Alt+F4 pro ukončení. Jakmile se vytvoří první terén, načítací obrazovka se zpoloprůhlední a je možné vstoupit do nápovědy. Ovládání je povoleno až po načtení celého prvního terénu.

Nápověda obsahuje přehled klávesových zkratek, ale neobsahuje všechy možnosti, kterými lze například tvořit objekty. Mým cílem bylo vytvořit jednoduchou nápovědu pro ovládání, kde by vše bylo na jednom místě a nebylo nutné rolování, aby se uživatel podíval, jakým způsobemse ovládá kamera. Pro podrobnější informace je zde manuál.

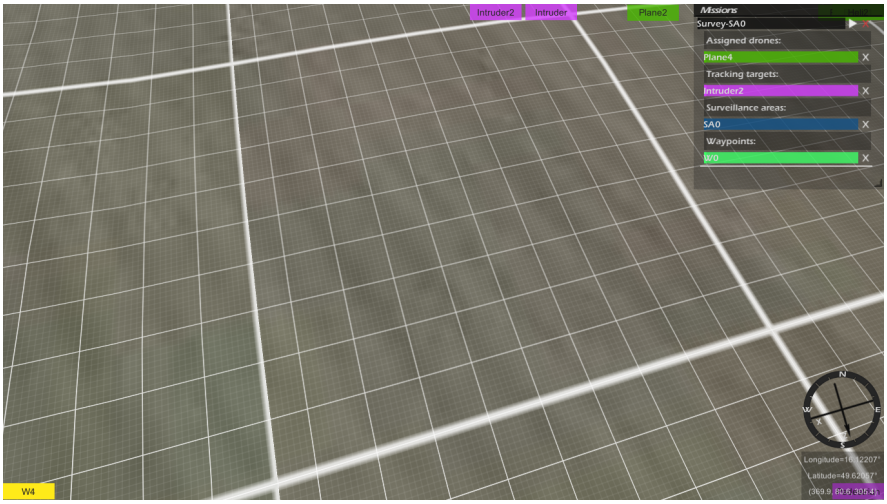
4.11 Uživatelské rozhraní

Uživatelské rozhraní se skládá z několika částí - již probrané štítky označující objekty, 4 oken, kompasu a pozice ve scéně.



Obrázek 14 Nápověda pro ovládání vizualizace.

Kompas a pozice ve scéně byly vytvořeny v reakci na testování, kdy uživatelům chyběla informace, jak jsou natočeni. Kompas také ukazuje souřadné osy. Dále jsem překryl terén čtvercovou mřížkou pro lepší určení vzdáleností. Vzdálenosti jsou po 100 m, 10 m a 1 m.



Obrázek 15 Čtvercová mřížka pro určení vzdáleností. Záběr na jeden čtverec s hranou 100 m.

Okna je možno kliknutím na hlavičku a táhnutím přetáhnout na požadovanou pozici a táhnutím pravého dolního rohu lze okno zvětšit/zmenšit. Jak pozice, tak velikost se ukládají do souboru panels.txt. V případě, že by okna nebyla vidět, můžete tento soubor smazat a okna se objeví v jejich základní poloze a velikosti. Tato okna mají stálou fyzickou velikost v pixelech.

4.11.1 Mise

Mise se skládá z horního panelu, které obsahuje název mise a vedle něj je tlačítko pro spuštění/pozastavení mise, a pak křížek, kterým lze mise odstranit. Kliknutím na název mise se obsah mise dá skrýt.

Mise obsahuje místo pro všechny objekty - zvlášť jsou navigační body, UAV, sledované cíle a sledované oblasti. Bezletové zóny působí na všechny UAV. Každé z těchto tlačítek (náležitá barva podle typu objektu) lze pomocí křížku odstranit. Pro navigační body navíc platí, že pokud na jeden klikneme a táhnutím přesuneme na jiný, pak se táhnutý navigační bod přesune na místo před bodem, kde jsme ho pustili. Že lze objekt pustit značí šipka s kružnicí.

Obrázek 16 Jak vypadá mise.

Do mise lze jednoduše přidávat objekty táhnutím a puštěním nad jednou z položek v misi nebo na její název. Objekt se automaticky přiřadí, kam je potřeba. Že lze objekt pustit se opět značí šipkou s kružnicí.

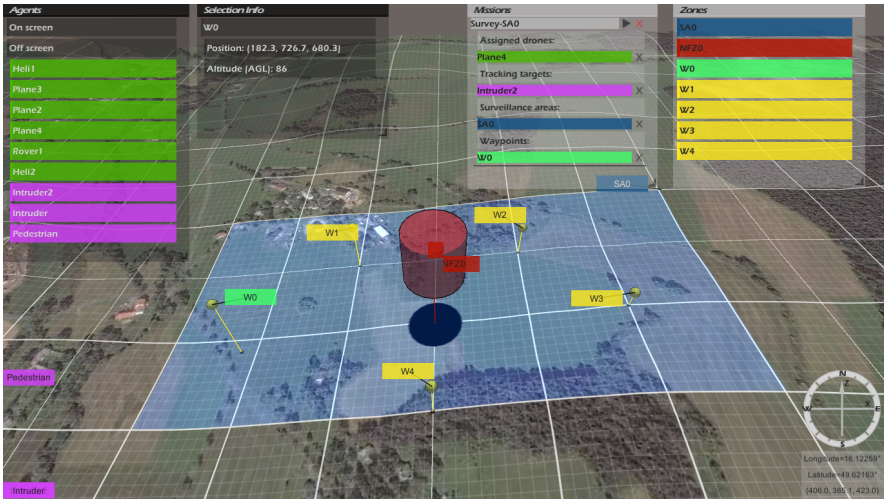
4.12 Nastavení

Pod klávesou O skrývající se nastavení aplikace nabízí tyto možnosti: vypnutí a zapnutí změny velikosti UAV v závislosti na vzdálenosti od kamery, jestli se mají smazat soubory v cache po ukončení aplikace, jestli se stahovaná data mají ukládat do cache, jestli se mají zobrazit budovy a jestli má být zobrazeno pouze okno s misemi. Také zde lze nastavit velikost generovaného terénu, kdy přednastavené velikosti jsou *small*, *medium* a *large*, kdy dlaždice jsou generovány v kruhu o poloměru 1500 m, 3500 m a 5500 m, zároveň se mění i vzdálenost, s jakou se terén maže a tato vzdálenost je o 3000 metrů větší než poloměr generovaného kruhu.

Jako poslední možností nastavení je barevné schéma, jsou vytvořeny dvě a to světlé a tmavé. Při nižším osvětlení je pro oči šetrnější tmavé schéma zatímco pro osvětlené

Obrázek 17 Přehled nastavení.

prostředí se hodí světlé schéma. Zároveň se tmavé schéma hodí v případě využití s průhledovými brýlemi pro rozšířenou realitu, kdy je tmavá, resp. černá barva vnímána jako průhledná. Taktéž ve virtuální realitě, za použití technologií jako HTC Vive nebo Oculus Rift, se více hodí tmavší barvy, protože z příliš jasných barev může často dojít k bolestem hlavy.



Obrázek 18 Dvě různá barevná schémata.

Pro ukládání nastavení využívám třídu v Unity $PlayerPrefs$, která ukládá data do hashovací tabulky pomocí klíče - řetězce znaků. $PlayerPrefs$ nejsou určena k uchovávání velkého množství dat, už jen vzhledem k místu, kam se ukládají, nejsou vhodná k častému zápisu a ani tak nebyla tato třída zamýšlena, jak při testování ukázala práce PreviewLabs [15]. Na Windows se ukládají do registrů pod HKCU/Software/jméno společnosti/jméno programu, kde jména jsou nastavitelná v Unity Project Settings. Na Linuxu se ukládají do home/.config/unity3d/jméno společnosti/jméno programu a na Androidu jsou uložena v /data/data/pkg-name/shared_prefs/pkg-name.xml.

4.13 Port na Android

Vzhledem ke snadné přenositelnosti mobilních zařízení, jsem vytvořil port na Android. Umožňuje pouze pohled shora a tak stačí generovat nejmenší terén. Nezaznamenal jsem viditelné sekání a pohyb nad terénem byl plynulý. Ověřil jsem také funkčnost napojení na simulátor bežící na jedné síti. Samozřejmě byla potřeba udělit vyjímka pro firewall, aby se vizualizace mohla připojit.

Nebylo potřeba mnoho změn. Bylo potřeba změnit načítání dat ze souboru, jelikož cesta k datům je dostupná pouze v rámci funkcí Unity *MonoBehaviour* a také byl potřeba upravit uživatelský vstup. Ten jsem vyřešil zavedením nové třídy *InputModule*, která poskytuje abstraktní rozhraní k funkcím simulátoru a záleží na konkrétní implementaci, jak to bude řešit. Pro stolní počítač jsem ponechal využití klávsenice s myší a pro Android jsem ovládání přepsal na použití dotykového displeje.

Přestože aplikace je takto funkční, až na problémy spjaté s uživatelským vstupem, bylo by v každém případě nutné vytvořit stabilnější a jednodušší ovládání, protože jsem ho pro stolní počítač založil na používání klávesových zkratek a omezeném počtu tlačítek. Přesto je však možné vytvořit a plánovat mise.

5 Testování

5.1 Performance testing

V této části se zabývám srovnáváním rychlostí tvorby terénu a počtu FPS v závislosti na velikosti terénu. Testování jsem prováděl při rozlišení 1366 × 768 a na Unity nastavení kvality *Fantastic*, které je nejvyšší možné. Oblast testování se nachází na GPS souřadnicích 49.620955° severní šířky a 16.124796° východní délky, což odpovídá oblasti nad vesnicí Kuklík.

Parameter	CPU	GPU
OS	Windows 10 Home 64-bit	
Chip	AMD A8-4500M	AMD Radeon HD 7640G + R5 M230
Frequency	1.9GHz	885MHz
RAM	8GB DDR3	4GB DDR3
Processing Units	4	-

Tabulka 2 Parametry počítače, na kterém se prováděly testy.

Testovala se rychlost stahování terénu v závislosti na velikosti terénu a použitém algoritmu - pomocí *Coroutine* a pomocí *ThreadedJob*. Dále se porovnávala procesem zabraná paměť a velikost uložených dat a jejich počet. Nakonec se porovná tvorba terénu, pokud se použije Unity *Terrain* nebo 3D síť, kde se porovnávají i různé velikosti 3D trojúhelníkové sítě.

Poloměr terénu	1500 m	3500 m	5500 m
Počet souborů v cache	150	606	1206
Velikost dat v cache	6.11 MB	25.1 MB	50.9 MB
Paměť procesu	203 MB	418 MB	606 MB

Tabulka 3 Velikost a počet uložených souborů a nároky procesu na paměť.

Rozlišení dlaždice terénu	17 × 17	33 × 33	65 × 65	129 × 129
Paměť procesu	418 MB	425 MB	469 MB	858 MB

Tabulka 4 Nároky procesu na paměť při velikosti terénu 3500 m a různým počtem vrcholů pro 3D síť.

FPS samozřejmě klesají při tvorbě terénu, ale při vykreslování se drží vždy alespoň na 60 snímcích za vteřinu. Hodnoty by se mohly lišit, kdyby byl vypnutý VSync. Při stahování z webových služeb platí, že první dlaždice je stažena nejpozději do dvou vteřin, což pro stahování pomocí *Coroutine*, které stahuje dlaždice postupně, znamená

Generování terénu	Maximální čas [ms]	Průměrný čas na dlaždici [ms]
Unity Terrain	72	45.22
3D mesh 17×17	57	3.53
3D mesh 33×33	44	12.86
3D mesh 65×65	102	56.25
3D mesh 129×129	480	304.2

Tabulka 5 Průměrný a maximální čas na vytvoření jedné dlaždice terénu.

výrazné zpomalení. U *Coroutine* se mi nepovedlo vytvořit rozumnou paralelizaci a toto řešení poskytovalo nejméně viditelný pokles FPS. Toto není takový problém u *PopulateTileJob*, kde je plná paralelizace, jak pro stahování jednotlivých dat, tak pro spuštění více prací najednou. *Coroutine* se daří lépe při načítání dat ze souboru, kdy doba načítání pro jednu dlaždici je až 10 ms, zatímco pro *PopulateTileJob* trvá načítání alespoň 150 ms.

5.1.1 Nastavení testů

Je možné si vytvořit jednoduché vlastní testy přidáním argumentů programu v příkazové řádce nebo využít a upravit existující spustitelné testy - soubory .bat začínající run-test. Aby program ukládal data z testování, je nutné aby byl specifikován argument +test. Mezi další parametry patří +mesh-size, které určuje, jak velká bude generovaná 3D trojúhelníková síť a musí být ve tvaru $2^n + 1$ a maximální velikosti 129. Pokud se zvolí +unity-terrain, pak se ignoruje předchozí příkaz mesh-size a k vykreslení dlaždice se použije Unity *Terrain*. V základu se používá ke stahování dat *PopulateTileJob*, ale je možné pomocí +unity-coroutine přikázat použití *coroutine* pro stahování dat. Dále lze specifikovat poloměr generovaného terénu pomocí +terrain-*, kde * je nahrazena jednou z velikostí: small, medium nebo large.

Výpisy z testů se ukládají do složky test pojmenované podle zvolených argumentů testu. Složka test obsahuje také skript parsetest.py v Pythonu, který po spuštění načte výpisy a převede je na srozumitelný text.

Samozřejmě i samotný výstup je čitelný. Pouze obsahuje velké množství dat, které je vhodné zpracovat. Data jsou ve formátu klíč=hodnota. Klíče můžou být "Count", "Populate", "Terrain". Populate a Terrain reprezentují informace o rychlosti stažení/-vytvoření. Count je použit, pokud aplikace stahuje pomocí PopulateTileJob a zapíše počet stažených dlaždic. Každý výstup testu je zapsán pouze jednou a to po ukončení generování prvního kruhu terénu.

5.2 Uživatelské testování

Prováděl jsem kvalitativní testování s 5 osobami. První dva testy probíhaly s dostatečným následným časovým rozmezím, aby po nich bylo možné zásadní nedostatky nebo chyby opravit. Se zbylými třemi participanty jsem prováděl testování hned po sobě. Díky tomuto rozdělení jsem měl možnost opravit velké množství drobných, ale i závažných chyb.

Pro participanty jsem měl připraveno 7 úkolů, které se měli pokusit vyřešit, kdy jako jedinou nápovědu dostali, jak funguje základní ovládání, jak lze vyvolat menu a nápověda. Úkoly se týkaly ovládání a vytváření misí a zahrnuly všechny možnosti, které komunikace se simulátorem umožňovala - jednalo se tedy především o vytváření bezletové zóny, přiřazení UAV k misím, sledování oblasti a pohyblivého cíle, nechat proletět UAV přes vytvořené navigační body a nakonec měli změnit různá nastavení. Seznam úkolů je dodán v příloze A 6.

5.2.1 První testování

Pro první dva participanty úkoly byly obecného rázu a zaměřovaly se spíše na to, jak budou schopni vytvářet jednotlivé mise a celkově interagovat s programem. Průběh byl volnější a nedržel se striktně rozdělení dotazník před testem, provádění úkolů, dotazník po testu, takže jsem uvítal, pokud měli v průběhu poznámky a nápady na vylepšení, které jsem s nimi rozebíral a případně použil, než jsem se pustil do dalšího testování.

První participant

První participant byl velmi výřečný, neměl potíže s mluvením a pečlivě komentoval, co provádí, nad čím uvažuje a dokonce i co mu vadí nebo co se mu zrovna líbí. Uživatel má praxi s 3D nástroji jako Blender, Maya nebo Unity, na nichž jsem založil ovládání.

Nejprve si při plnění úkolů osahal ovládání a často využil nápovědy. S ovládáním neměl problémy, pouze se několikrát tvářil zmateně, když se spletl při použití modifikátorů Shift a Control. Měl problémy při vytváření bezletové zóny, u které bylo ovládání při vytváření odlišné od ostatních, tj. postrádalo táhla, kterými by šlo měnit hodnoty. Dále měl problémy s pochopením názvů os a nevěděl, jestli nadmořská výška je počítána od země nebo od úrovně moře.

Při tomto testu jsem používal k označení objektů ve scéně billboardy inspirované z her, tj. billboard je ukotven nad objektem, natáčí se ke kameře, ale nijak není zabráněno překrývání objektů. Toto překrývání participantovi velmi vadilo: nejen překrývání objektů, ale zároveň i štítků. Také mu vadilo, že není dostatečně přesně v uživatelském rozhraní rozlišené, kde končí jedna mise a začíná druhá. Postrádal možnost editace již vytvořených objektů.

V závislosti na některé přípomínky, problémy a moje pozorování, byly před testováním s druhým participantem provedeny následující úpravy (vyjma oprav chybného nebo nepředpokládaného chování):

1. Nápověda je možná zobrazit již v okamžiku, kdy je načtena první dlaždice a začne se objevovat terén.
2. Přetáhnout objekt jde do celé mise, nejen na její jméno.
3. Kamery sdílí pozici a rotaci, pokud to má smysl. Například nemá význam, aby si kamera pro chodce udržela pozici kamery ve vzduchu.
4. Při dialogu pro vytváření je automaticky umístěň $focus$ na vstup pro název objektu.
5. Dialog pro vytváření je nyní možné ukončit pomocí klávesy Escape nebo potvrdit pomocí klávesy Enter.

6. Při vytváření bezletové zóny jsou nyní umístěna tři táhla, pro výšku, pro poloměr a pro pozici.

7. Po každé změně nebo při pokusu o vytvoření objektu je provedena kontrola a jsou znázorněna pole, která jsou chybně vyplněná nebo jsou ještě potřeba vyplnit.

8. Vybrané objekty jsou nyní znázorněny i v oknech se seznamy UAV, misí a zón.

9. Prvkům v oknech mají barvy podle barevného označení jednotlivých objektů ve scéně.

10. Jednotlivé mise v okně jsou odděleny horizontální čárou barvy textu a kliknutím na text mise je možné všechny data o ní schovat.

11. Přidáno možnost zvětšit/zmenšit hodnoty ve vstupních polích pomocí táhnutí vpravo/vlevo.

12. Změna popisků objektů. Popisky jsou nyní pouze v prostoru obrazu, a aby se nepřekrývaly, využívá se algoritmu založenén na použití odpuzujících a přitažlivých sil v grafu[12].

Druhý participant

Druhým participantem je uživatel stávajícího ovládání pro simulátor, je tedy také pravděpodobným uživatelem mojí vizualizace. S druhým participantem jsem v průběhu konzultoval, na jaká značení je zvyklý nebo se v letectví používají. V průběhu testování pečlivě komentoval, o co se snaží a co dělá.

Participant pracoval ve větší vzdálenosti, aby vždy viděl celou scénu, a nebo alespoň její větší část. Během vytváření navigačních bodů si nevšiml, že je možné jich vytvořit více najednou. Omylem objevil možnost schování informací v mise, když se překlikl, ale pak ji začal používat. Do mise vždy přiřazoval objekty po jednom. Během načítání zkoušel rotovat kameru. Také mu, jako prvnímu participantovi, scházela možnost dodatečně upravit objekt ve scéně.

V závislosti na přípomínky ke konvencím v letectví, problémy a moje pozorování, byly provedeny následující úpravy (vyjma oprav chybného nebo nepředpokládaného chování):

1. Přidána možnost zpětné úpravy již vytvořených objektů.

2. Možnost změnit pořadí navigačních bodů v misi.

3. Možnost přidat více stejných navigačních bodů do jedné mise.

4. Při vytváření přidána možnost zadání nadmořské výšky relativně k zemi.

5.2.2 Druhé testování

Pro zbylé participanty jsem změnil úkoly v testu. Byly více specifické a jejich součástí bylo i určování vzdáleností mezi navigačními body nebo od předchozí pozice. Také jsem již striktně dodržoval rozdělení pre-test dotazník, úkoly a post-test dotazník. V dotazníku před testem mě především zajímalo, co je součástí jejich práce a jestli obsahuje využívání stávající vizualizace. V dotazníku po testu jsem se ptal, co se jim

líbilo, či nelíbilo a zaměřoval jsem se také na rychlost jejich odpovědi. Pokud dokázali odpovědět okamžitě, pak samozřejmě aplikace zanechala silnější dojem a odpověď tak má větší váhu, než když se pomalu snažili najít nějaké pro a proti. Což pochopitelně neznamená, že by se pomalé odpovědi měly úplně zanedbat a to především negativní, protože někteří participanti se budou snažit říct alespoň něco dobrého, aby neurazili.

První participant

Participant nevyužívá stávající vizualizaci. Plánuje mise před letem a při testech venku kontroluje let UAV a v případě potřeby převezme jeho ovládání.

Během testu nabízel řešení a odkazoval se na odlišné nástroje, které při plánování misí používá. V průběhu testování komentoval, o co se snaží a jestli mu něco nedává smysl.

U ovládání ocenil, že jeho rozložení je na jednom místě. Když v nápovedě viděl, že je možné přepínat kamery, tak nevěděl, co si má představit, jestli se například neobjeví nové okno s pohledem z dané kamery.

Při plnění úkolů si vybíral vzdálenější místa, kam zpočátku měl problém se dostat. Také měl problémy s orientací v prostoru, kdy nevěděl, kde je sever, tak by se mu hodil kompas nebo přehledová mapa. Zkoušel vytvářet mise tím, že přetáhnul sledovanou oblast na letadlo a naopak. Po vytvoření sledované oblasti nevěděl, jak pokračovat dál. Věděl, že simulátor stále neplánuje, ale nevěděl proč, a tak potřeboval upozornit, že sledovaná oblase neznamená její přiřazení k misi. Nebyl zvyklý na pojmenování os.

Při vytváření jednotlivých navigačních bodů podotkl, že by se mu hodilo, kdyby body začínaly ve stejné výšce nad zemí, protože málokdy se mění letová hladina, a pokud ano, pak je to jednorázové a UAV/letadlo pak na změněné zůstává. Tvrdil, že by se mu hodilo, mít možnost si nějak seskupit vytvořené navigační body, ale po přiřazení do mise, říkal, že tu skupinu tvoří právě ona mise.

Dříve než se pustil do samotné tvorby objektů, tak si nejprve vyplnil název, aby si udržel objekty zorganizované.

Téměř okamžitě říkal, že se mu nelíbí pohyb po mapě, protože na něj není zvyklý. Také postrádal možnost uložení mise, kdy by si ji mohl vytvořit v kanceláři, a pak ji nahrát do simulátoru přímo na místě testování.

Líbila se mu možnost vytvoření mise z právě vybraných objektů a také označení a přehled o letadlech.

Druhý participant

Druhý participant zodpovídá za manuální ovládání UAV. Vizualizici využívá zřídka, uváděl, že jednou do měsíce. Participant byl v průběhu testování tišší, ale komentoval to, co dělá, přestože mu občas nebylo rozumět.

Ocenil zoom a pohyb pomocí prostředního tlačítka myši, ale čekal, že pohyb vpřed nebude totéž jako zoom, ale že se bude pohybovat nad terénem ve stejné výšce. Jako předchozí participant očekával, že by mohlo fungovat přetáhnutí objektu na UAV, aby se vytvořila mise. Taktéž potřeboval nápovědu, že vytvořená sledovaná oblast neznamená, že je pro ni naplánovaná mise.

Při vyváření navigačních bodů poznamenal, že by očekával, že lze vytvořit více navigačních bodů najednou, když se funkce jmenuje *New Waypoints*, ale přestože se mu při vytváření objevovaly při kliknutí na terén další navigační body, tak si neuvědomil, že tak funkce skutečně funguje. Také měl problémy s pozicováním kontext menu v závislosti na pozici myši. Také pro přehlednost vyplňoval nejdříve názvy objektů. Jelikož často chyboval, když si nechtěně vytvořil objekt pomocí enteru nebo změnil velikost za využití jednoho pomocného táhla, tak by uvítal možnost funkce *Zpět*.

Nelíbilo se mu, že potřeboval pochopit koncept misí a že si potřeboval zvyknout na kontextové menu schované pod tlačítkem Alt.

Líbilo se mu, že přepínání kamer zachovává pozici a je velmi jednoduché. Za snadné také považoval vybírání a líbila se mu možnost hromadného výběru za držení tlačítka Shift. Také považoval za dobré zobrazení sledované oblasti, protože nepřekáželo ve výhledu.

Třetí participant

Náplní práce třetího participanta byla tvorba plánování pohybu UAV. Stávající simulaci viděl, ale už si nepamatuje, co v ní vlastně vidí.

Na ovládání se mu nelíbíla nastavená rychlost pro zoom a vertikální pohyb pomocí mezerníku. Taktéž očekával, že je možné vyskočit z nápovědy pomocí Escape. U kamery z pohledu drona očekával, že kamera si bude držet směr pohledu v závislosti na rotaci UAV nebo že bude zpoza UAV, aby viděl případné kolize.

Při psaní do vstupních polí byl chvíli zmaten, že nemůže změnit kameru a neuvědomil si, že právě píše do vstupního pole. Také si myslel, že stačí vytvořit oblast, aby se začalo plánovat její sledování, takže potřeboval nápovědu jako předchozí dva participanti.

Přišlo mu nelogické a nepřehledné, že při vytváření se neukazuje zvýraznění objektu, když se na ně najede myší, a že by se mu hodilo, aby se pod myší ukazoval objekt, který je možné vytvořit. Jelikož součástí otázek byl i odhad vzdálenosti, tak mu chybělo měřítko a kompas pro orientaci. Také by uvítal, kdyby se mu předvyplnil název vytvářených objektů.

Neuvědomil si, že přepnutí kamery zároveň znamená i změnu ovládání, a byl tak chvíli zaseklý v pohledu shora, kdy se mu nedařilo nic dělat.

Uvedl dvě věci, které se mu nelíbily, a to, že zoom pomocí kolečka byl pomalý a že je zvykem pro mapy využívat k pohybu levého tlačítka myši a ne prostředního.

Výsledky druhého testování

V závislosti na přípomínky, problémy a moje pozorování, byly provedeny následující úpravy (vyjma oprav chybného nebo nepředpokládaného chování):

1. Přidána možnost vyskočit z nápovědy a nastavení pomocí klávesy Escape, kdy pro nastavení Escape ukončí bez uložení a klávesa Enter ukončí s s uložením.
2. Rychlost přibližování pomocí kolečka myši je nyní ovlivňena vzdáleností k bodu, kam se uživatel kouká.
3. Přidáno automatické generování jmen pro bezletovou zónu a sledovanou oblast.
4. Vytvářet misi nyní lze i přetažením štítku jednoho objektu na druhý. Taková mise si zároveň vygeneruje jméno podle objektů, které se přiřadí do mise.
5. Rozdělena vstupní pole pro nadmořskou výšku a pozici.
6. Přidán kompas, který určuje, kterým směrem se nachází sever a jak jsou natočené světové osy.
7. Nové navigační body se vytváří se stejnou nadmořskou výškou, jako poslední vytvořený bod.
8. Volná kamera se nyní šipkami vpřed/vzad pohybuje ve stejné nadmořské výšce.
9. Terén je nyní pokryt čtvercovou mřížkou s rozlišením po 100, 10 a 1 metru.

5.2.3 Společné problémy participantů

Většina participantů si vůbec neuvědomila, že se automaticky k vytvořené misi přiřadí právě vybrané objekty, pokud je možno je do mise vůbec přiřadit. V případě, že si to uvědomili, pak pouze náhodou. Jen první z participantů si všiml v nápovědě, že je možné přiřadit do mise přetáhnutím všechny vybrané objekty. Někteří na toto přišli taktéž náhodou, když měli více objektů vybraných a přiřadili nějaký objekt misi. Pokud si však těchto možností všimli, velmi je uvítali.

V průběhu testování jsem zvažoval, jestli je potřeba vytvářet terén s takovým rozlišením, jelikož většina uživatelů málokdy využila možnost práce z pozice chodce a pracovala z větší výšky, aby měla přehled nad celou scénou. Snížení detailů by umožnilo zvětšit terén, aby pokrýval celý pohledový jehlan kamery, což by nejspíš potřebovalo generování detailnějšího terénu v závislosti na vzdálenosti od kamery.

6 Závěr

Cílem mé práce bylo navrhnout a naimplementovat aplikaci, která by vizualizovala pohyb bezpilotních leteckých prostředků a zároveň by umožňovala plánování misí pro jednotlivé prostředky, kdy plánování by prováděl simulátor od AgentFly na Katedře počítačů.

Aplikace byla otestována s lidmi z Agent Technology Center, a jelikož jsem opravoval všechny zásadní nedostatky, myslím, že jsem vytyčené zadání splnil, ale vidím zde prostor pro vylepšení. V rámci post testu jsem se s několika participanty bavil, jak vlastně plánují mise. Bylo mi řečeno, že si často naklikají misi v Google Maps nebo využijí softwaru jako je UgCS, kde si mapu připraví a až poté naklikají v simulátoru. Všechny tyto aplikace využívají formátu KML, který by šel v této práci taktéž využít. Bylo by tak třeba možné naklikat si navigační body na mapách, a pak přenést do mé vizualizace, nebo si misi naplánovat ve vizualizaci, kde by se uložila a až by se zkoušela ve venkovních podmínkách, stačilo by ji znovu nahrát.

V průběhu práce jsem narazil na beta verzi SDK pro Unity od Mapboxu, které právě umožňuje generování světa na základě jejich map. Generování řeší jiným způsobem než já: stahují vždy jen jednu dlaždici a lze vidět, že nejdříve se tvoří terén, a pak aplikují satelitní snímky, a když se vytvoří dlaždice okolo, tak se spojí nadmořské výšky, zatímco já si je nejdříve stáhnu a připravím. V ohledu stahování mi přišlo toto řešení rozumnější, ale nelíbí se mi změna výškových dat za běhu. Přivedlo mě to však na myšlenku, kdy by dlaždice terénu byly menší a zůstal by mezi nimi jakýsi volný pás, který by se doplnil, jakmile by se sousední dlaždice stáhla. Takto by nedocházelo k změně výškových map a navíc by mohlo být vytvořené LOD pouze pro část dlaždic terénu, protože by stačilo spojit nejbližší vrcholy u sousedních dlaždic.

Literatura

[1] Mapzen. *Terrain Tile Services*. URL: https://mapzen.com/documentation/terrain-tiles/ (cit. 06. 05. 2017).

[2] Mapbox. *Mapbox*. URL: https://www.mapbox.com/ (cit. 06. 05. 2017).

[3] Wikipedia. *Mercator projection - Wikipedia*. URL: https://en.wikipedia.org/wiki/Mercator_projection (cit. 16. 01. 2017).

[4] OpenStreetMap. *Slippy map tilenames - OpenStreetMap Wiki*. URL: http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames (cit. 11. 05. 2017).

[5] Mapbox. *Global Elevation Data | Mapbox*. URL: https://www.mapbox.com/blog/terrain-rgb/ (cit. 11. 05. 2017).

[6] Mapzen. *Layers - Vector Tile Service*. URL: https://mapzen.com/documentation/vector-tiles/layers/ (cit. 11. 05. 2017).

[7] Mapzen. *Get started*. URL: https://mapzen.com/documentation/overview/#get-started-developing-with-mapzen (cit. 12. 05. 2017).

[8] Mapbox. *Developers | Mapbox*. URL: https://www.mapbox.com/developers/ (cit. 12. 05. 2017).

[9] Unity. *Unity - Manual: Terrain Settings*. URL: https://docs.unity3d.com/Manual/terrain-OtherSettings.html (cit. 16. 01. 2017).

[10] Unity. *Unity - Manual: Execution Order of Event Functions*. URL: https://docs.unity3d.com/Manual/ExecutionOrder.html (cit. 12. 05. 2017).

[11] Robert Nystrom. *Game programming patterns*. Genever Benning, 2014.

[12] Jan Balata, Ladislav Čmolík a Zdeněk Míkovec. "On the Selection of 2D Objects Using External Labeling". In: *Proceedgings of SIGCHI Conference on Human Factors in Computing Systems*. 2014, s. 2255–2258.

[13] Unity. *Unity - Manual: Coroutines*. URL: https://docs.unity3d.com/Manual/Coroutines.html (cit. 16. 01. 2017).

[14] Ian Garton. *An O(kn) Time Algorithm for Finding an Ear*. URL: http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/97/Ian/algorithm1.html (cit. 12. 05. 2017).

[15] PreviewLabs. *PreviewLabs PlayerPrefs vs Unity PlayerPrefs*. URL: http://previewlabs.com/previewlabs-playerprefs-vs-unity-playerprefs/ (cit. 17. 05. 2017).

[16] AgentFly Technologies. *Tactical AgentFly installation plan.* Ver. 1.0. 2016.

Příloha A - Dotazníky pro testování

Pre-test dotazník

1. Co je náplní Vaší práce?
2. Používáte při práci stávající vizualizaci.

Post-test dotazník

1. Co se Vám líbilo na vizualizaci a jejím ovládání?
2. Co se Vám **ne**líbilo na vizualizaci a jejím ovládání?

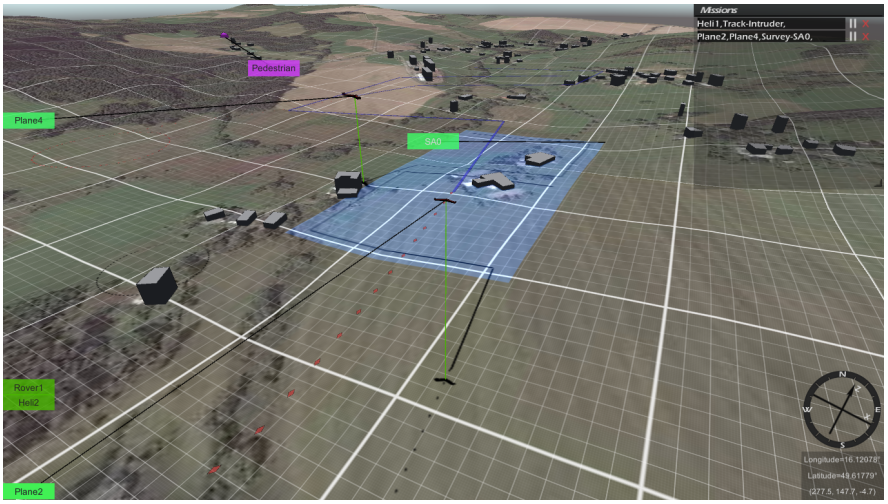
Úkoly pro první dva participanty

1. Pro Plane2, Plane3 a Plane4 vytvořte misi, kdy UAV budou hlídkovat nad Vámi vytvořenou oblastí.
2. Vytvořte misi, ve které budou Heli1, Plane2 a Plane3 hlídkovat nad jinou Vámi vytvořenou oblastí.
3. Do jedné z misí přiřaďte Heli2.
4. Nechte Plan4 proletět přes 4 Vámi vytvořené waypointy, které budou alespoň 50 metrů nad zemí.
5. Vytvořte bezletovou zónu.
6. Vypněte zobrazení budov.
7. Nastavte vymazání cache souborů pro ukončení programu.

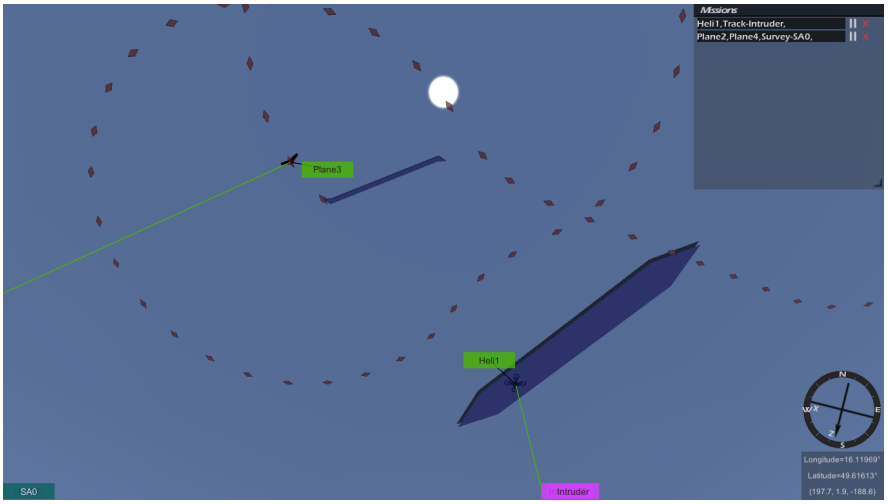
Úkoly pro participanty

1. Vytvořte misi pro Plane2 a pro Plane3, kdy letadla budou hlídkovat nad Vámi vytvořenou oblastí. Tato oblast by měla zahrnovat jednu z vesnic.
2. Vytvořte 4 waypointy a nechte jimi proletět Plane4. Waypointy by od sebe měly být vzdáleny kolem 80 metrů a měly by být výše jak 50 metrů.
3. Schovejte informace u obou misí.
4. Nechte Heli2 sledovat Intruder2.
5. Vytvořte bezletovou zónu, která bude zasahovat do oblasti, nad kterou hlídkují Plane2 a Plane3.
6. Upravte hlídkovanou oblast. Zvětšte ji o 50 metrů na každé straně.
7. Vypněte zobrazení budov.
8. Nastavte vymazání cache souborů pro ukončení programu.

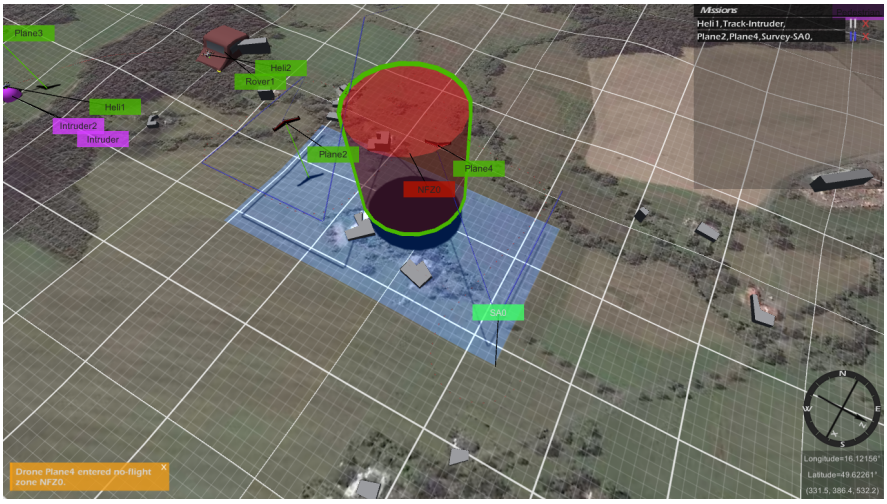
Příloha B - Screenshoty z aplikace



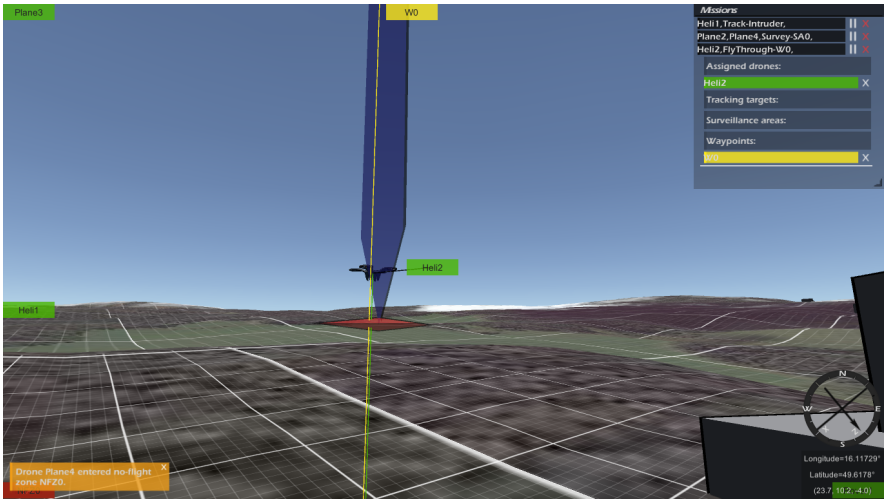
Obrázek 19 Plane4 a Plane2 míří k sledované oblasti SA0.



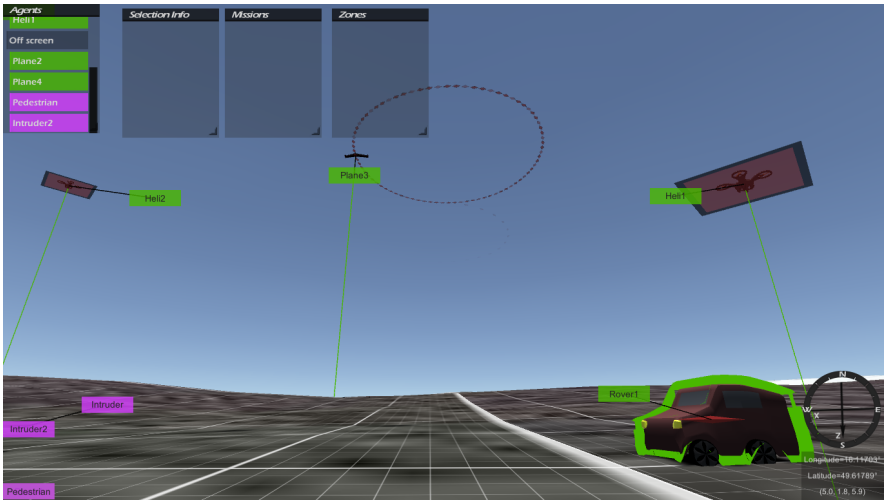
Obrázek 20 Plane3 a Heli1 sledují Intruder. Heli1 vysí ve vzduchu, zatímco Plane3 létá v kruzích.



Obrázek 21 Sledování oblasti SA0 tak, aby se vyhnulo zvýrazněné bezletové zóně. Zároveň se Plane4 snaží dostat z bezletové zóny, ve které se nachází.

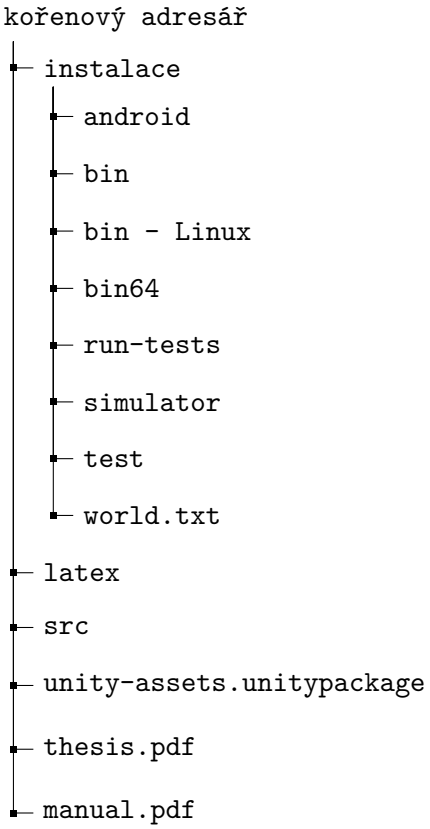


Obrázek 22 Heli2 pomalu stoupá k navigačnímu bodu W0.



Obrázek 23 Pohled na pozemní vozidlo, kroužící křídlo a dvě kvadrokoptéry.

Příloha C - Obsah přiloženého CD



Složka instalace obsahuje všechny potřebné soubory pro instalaci a spuštění. Jak se instalace provádí a aplikace ovládá lze najít v manuálu. Ve složce latex se nacházejí zdrojové kódy, ze kterých se vygeneroval manuál a diplomová práce. Unity-assets je soubor, který lze otevřít v Unity a obsahuje všechny soubory, které jsou potřeba pro vytvoření a spuštění projektu. Ve složce src se nacházejí zdrojové kódy k vizualizaci, pokud by někdo bez Unity chtěl vidět samotnou implementaci (a pokud ano, doporučuji začít World.cs v src/WorldScripts).