TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Aplikace pro autonomní navigační systém

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Ondřej Kubíček**
Vedoucí práce: Ing. Igor Kopetschke

Liberec 2017

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Application for autonomous navigation system

Master thesis

Study programme:	N2612 – Electrical Engineering and Informatics
Study branch:	1802T007 – Information Technology
Author	**Bc. Ondřej Kubíček**
Supervisor:	Ing. Igor Kopetschke

Liberec 2017

Technická univerzita v Liberci
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2016/2017**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Ondřej Kubíček**
Osobní číslo:	**M15000176**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Aplikace pro autonomní navigační systém**
Zadávající katedra:	**Ústav nových technologií a aplikované informatiky**

Zásady pro vypracování:

1. Seznamte se s API pro získávání dat ze systémů ArcGIS a OpenStreetMap.
2. Na základě bodu 1) navrhněte způsob pro automatické získávání mapových a datových podkladů.
3. Na základě získaných dat modelujte formou preprocesingu reálné 3D prostředí, zdůvodněte potřebu dat z obou mapových systémů.
4. Pomocí jazyku Java a OpenGL vytvořte vzorovou aplikaci, na které budete demonstrovat výsledek své práce.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **40 - 60 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] SHREINER, Dave. OpenGL: průvodce programátora. Vyd. 1. Přeložil Jiří FADRNÝ. Brno: Computer Press, 2006. DTP & grafika. ISBN 80-251-1275-6.
[2] GELETIČ, Jan. Úvod do ArcGIS 10. 1. vyd. Olomouc: Univerzita Palackého v Olomouci, 2013. Skripta. ISBN 978-80-244-3390-5.
[3] ArcGIS REST API. ArcGIS REST API [online]. Český úřad zeměměřický a katastrální, 2016 [cit. 2016-10-12]. Dostupné z: http://ags.cuzk.cz/arcgis/sdk/rest/index.html
[4] OpenStreetMap API. OpenStreetMap [online]. OpenStreetMap Wiki, 2016 [cit. 2016-10-12]. Dostupné z: https://wiki.openstreetmap.org/wiki/API

Vedoucí diplomové práce: **Ing. Igor Kopetschke**
Ústav nových technologií a aplikované informatiky

Datum zadání diplomové práce: **20. října 2016**

Termín odevzdání diplomové práce: **15. května 2017**

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

L.S.

prof. Dr. Ing. Jiří Maryška, CSc.
vedoucí ústavu

V Liberci dne 20. října 2016

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15.5.2017

Podpis:

Poděkování

Rád bych tímto poděkoval panu Ing. Igoru Kopetschkemu za vedení a věcné připomínky k mé diplomové práci.

Abstrakt

Tato práce se zabývá návrhem zpracování dat ze služeb OpenStreetMap a ArcGIS Server od Českého úřadu zeměměřičského a katastrálního. Pro získaná data je v práci navržen datový model, který bude použitý v autonomním navigačním systému (dron nebo pozemní vozidlo) pro umožnění detekce kolizí a orientace v prostředí, kde se pohybuje. Výsledkem práce je desktopová aplikace, která poskytuje mapu České republiky. Tato aplikace slouží pro výběr oblasti pohybu pro vybrané cílové zařízení spolu se všemi náležitostmi, jako je zarovnání oblasti do navrženého mřížkového systému, vyhledávání míst v mapě nebo výběr způsobu, jakým se bude zařízení v oblasti pohybovat. Dále je v práci popsána implementace aplikace, která realizuje 3D vizualizaci zvolené oblasti. Pro vi-zualizaci je využit navržený datový model a reflektuje tak reálný vzhled krajiny spolu se zájmovými entitami.

Klíčová slova: 3D vizualizace, ArcGIS, autonomní navigační systém, Java 3D, OpenStreetMap

Abstract

This thesis focuses on design of processing data from OpenStreetMap and ArcGIS Server provided by Czech Office for Surveying, Mapping and Cadastre. For the data obtained from services, a data model is designed to be used in autonomous navigation system (drone or ground vehicle) to enable collision detection and orientation of the device in real enviroment. The result of this thesis is a desktop application that provides map of Czech Republic. This application is used to select area where the device should move, along with all the necessary features, such as aligning the area to designed grid system, search for locations in map, or select how the device should move in selected area. Thesis also describes implementation of application which creates 3D visualization of given area. To render the scene, the designed data model is used. Created scene reflects the real appearance of landscape along with entities of interest.

Keywords: 3D visualization, ArcGIS, Autonomous Navigation System, Java 3D, OpenStreetMap

Obsah

Úvod ... 10
1 Autonomní navigační systémy ... 11
2 Webové mapové služby ... 12
2.1 Geografický informační systém a platforma ArcGIS ... 13
2.2 REST služby poskytované ČÚZK ... 14
2.2.1 DMP 1G ... 14
2.2.2 DMR 5G ... 15
2.2.3 Ortofoto ... 15
2.3 Projekt OpenStreetMap ... 16
2.4 Shrnutí ... 16
3 Java 3D ... 18
3.1 Struktura Java 3D aplikace ... 18
3.1.1 Typy objektů ve scéně ... 19
3.2 Silné a slabé stránky Java 3D aplikací ... 20
4 Návrh řešení ... 21
4.1 Komunikační schéma ... 21
4.2 Datový model ... 22
4.2.1 Konstrukce datového modelu ... 23
4.2.2 Objekty datového modelu ... 24
4.3 Režimy pohybu ANS ... 27
4.3.1 Režim náhodného pohybu ... 27
4.3.2 Režim pohybu po předem stanovené trase ... 28
4.3.3 Režim pohybu do cílového bodu ... 28
4.4 Aplikace pro výběr oblasti pohybu ... 28
4.4.1 Návrh grafického rozhraní ... 29
4.4.2 Zobrazení a nastavení mapy ... 30
4.4.3 Zvýšení efektivnosti cachování ... 31
4.5 Aplikace pro vizualizaci vybrané oblasti ... 31
4.5.1 Postup vykreslení scény ... 32
5 Serverová komponenta ... 34
5.1 Konstrukce dlaždic ... 34

5.1.1 Získání výškových profilů ... 35
5.1.2 Stažení satelitního snímku ... 36
6 Realizace aplikace pro výběr oblasti ... 37
6.1 Vytvoření mřížky pro zarovnání dlaždic ... 38
6.2 Výběr oblasti ... 39
6.2.1 Specifikace trasy pohybu ... 39
6.3 Vyhledávání v mapě ČR ... 40
6.4 Zobrazení oblastí se zakázaným pohybem ... 41
6.5 Získání dat ze serveru ... 41
6.5.1 Formát odpovědi ... 42
7 Vykreslení 3D scény ... 43
7.1 Příprava dat pro popis terénu ... 43
7.1.1 Vykreslení drátěného modelu terénu ... 45
7.1.2 Aplikování textury na terén ... 46
7.2 Vytvoření a vykreslení budov ... 47
7.2.1 Korekce vrcholů z OSM služby ... 48
7.3 Vizualizace bodů definujících trasu pohybu ... 49
8 Závěr ... 50
Seznam použité literatury ... 52
A Obsah přiloženého CD ... 55

Seznam obrázků

Obrázek 1: Ukázka grafu scény v Java3D 19
Obrázek 2: Komunikační schéma 22
Obrázek 3: Class diagram datového modelu 25
Obrázek 4: Drátěný model aplikace pro výběr oblasti pohybu 29
Obrázek 5: Reprezentace krychle v Java 3D 32
Obrázek 6: Ukázka výběru oblasti a specifikace trasy 40
Obrázek 7: Realizace trojúhelníkového pásu 44
Obrázek 8: Vymodelování terénu ve formě drátěného modelu 46
Obrázek 9: Terén s aplikovanou texturou 47
Obrázek 10: Ukázka dopadu uspořádání vrcholů na vykreslení entit 48
Obrázek 11: Vymodelovaná scéna se všemi implementovanými typy objektů 49

Seznam tabulek

Tabulka 1: Typy dotazů na WMS 12
Tabulka 2: Operace nad službami DMP 1G a DMR 5G 15

Seznam zdrojových kódů

Zdrojový kód 1: Tělo dotazu pro získání výškových vzorků 36
Zdrojový kód 2: Zarovnání výběru do mřížkového systému 39
Zdrojový kód 3: Formát JSON souboru definující zakázané oblasti 41
Zdrojový kód 4: JSON formát odpovědi ze serveru 42

Seznam zkratek

ANS	Autonomní Navigační Systém
ČÚZK	Český úřad zeměměřický a katastrální
DMP 1G	Digitální model povrchu České republiky 1. generace, ArcGIS služba od ČÚZK
DMR 5G	Digitální model reliéfu České republiky 5. generace, ArcGIS služba od ČÚZK
GUI	Grafické Uživatelské rozhraní
REST	Representational State Transfer, architektura rozhraní, navržená pro distribuované prostředí
S-JTSK	Systém jednotné trigonometrické sítě katastrální, souřadnicový systém využívaný v České republice a na Slovensku
SOAP	Simple Object Access Protocol, protokol pro výměnu zpráv založených na XML po síti
WMS	Web Map Server, webový mapový server dle specifikace Open Geospatial Consortium

Úvod

V posledních letech se čím dál více setkáváme s bezpilotními zařízeními, která se pohybují po zemi nebo ve vzduchu (tzv. drony) [1]. Dříve se tato zařízení hojně využívala především ve vojenské sféře [2]. Nyní se však dostávají i do civilní oblasti, kde nacházejí mnohá využití. Obvykle jsou tato zařízení charakteristická tím, že je pro jejich pohyb potřebná lidská obsluha, která pomocí dálkového řízení udává směr pohybu. Spolu s příchodem těchto zařízení se začínají vymýšlet způsoby, jak minimalizovat nebo úplně odstranit potřebu dálkového řízení člověkem a nechat orientaci v prostoru na samotném zařízení.

Tato práce se zabývá způsobem, jak reprezentovat data popisující vybranou oblast tak, aby mohla být následně využita k realizaci autonomní navigace zařízení v rámci této oblasti. To se stává obzvláště důležitým v oblastech s vysokým výskytem budov nebo přírodního porostu, jako jsou například města. V práci jsou popsány požadavky, které by měly informace o oblasti splňovat, aby mohlo zařízení rozpoznávat své okolí a rozhodovat se tak o svém pohybu. Na základě stanovených požadavků je navržen datový model, který slouží k popisu vybrané lokace pro pohyb zařízení. Pro konstrukci navrženého modelu je zvolen volně dostupný zdroj dat v podobě webových mapových služeb, ze kterých lze všechny potřebné informace získat.

Součástí práce je také návrh a implementace aplikace, která umožňuje zadávat libovolnou oblast na mapě České republiky. Tento výběr definuje oblast pohybu pro cílové zařízení. Na jeho základě je vytvářen datový model pro cílové zařízení a ostatní klientský software. Zároveň aplikace umožňuje definovat jeden z dostupných režimů, který určuje způsob, jakým se má zařízení pohybovat.

V poslední části práce je vytvořený datový model využitý k vymodelování 3D podoby oblasti, která je modelem popisována. Tato vizualizace reprezentuje skutečný vzhled krajiny spolu se všemi zájmovými objekty (jako jsou například budovy). Lze ji tak použít například k analýze zvolené oblasti před tím, než jsou data odeslána a použita v cílovém zařízení. V závěru práce jsou diskutovány dosažené výsledky a další možný rozvoj aplikace.

1 Autonomní navigační systémy

Autonomní systém je takový systém, který je schopný se pohybovat po nějaký časový úsek v libovolném prostředí reálného světa a to bez jakéhokoliv zásahu z vnějšího zdroje (například ovládání člověkem). Jako jeden z nejzákladnějších a nejdokonalejších autonomních systémů si můžeme uvést živé organismy na naší planetě. Ty zvládají přežívat v různorodých prostředích, která využívají k nalezení a získání potravy a materiálů potřebných k přežití. V převážné většině případů jsou schopné se přizpůsobit změnám prostředí, ve kterém se nacházejí [3].

Lidé se čím dál více snaží o vytvoření čím dál dokonalejších autonomních navigačních systémů (dále jen ANS) a často čerpají inspiraci právě v přírodě. ANS by měl být nejen plně kontrolovatelný člověkem, ale především by se měl umět spolehlivě pohybovat (alespoň v nezbytně nutných mezích, aby nedocházelo ke kolizím s ostatními objekty) v neznámém prostředí, detekovat a vyhýbat se překážkám a to na základě vlastního rozhodnutí [4].

Pro orientaci v prostoru jsou ANS velmi často vybaveny GPS lokátorem, díky kterému dokáží s velkou přesností určit, kde se právě nacházejí. Problémem GPS je možnost ztráty signálu. Pokud by systém spoléhal jen na GPS lokátor a nastala by ztráta signálu, mohlo by dojít k nepředvídatelnému chování, případně ke kolizi. Právě z tohoto důvodu se ANS doplňují dalšími podpůrnými senzory, které dopomáhají k přesnější orientaci v prostoru či detekci objektů pro zamezení kolize. Mezi takové senzory mohou patřit:

- akcelerometr, magnetometr, gyroskop,
- kompas, jedna nebo více kamer pro snímání okolí,
- laserový dálkoměr a další [5].

Pro potřeby této práce bereme v potaz dva druhy ANS - bezpilotní autonomní zařízení typu dron (UAV – Unmanned Aerial Vehicle) a pozemní zařízení (UGV – Unmanned Ground Vehicle). Zejména dron využívá výše zmiňovaných metod pro orientaci v prostoru. Zároveň je potřeba oběma typům zařízení poskytnout data popisující oblast, ve které se mají pohybovat.

2 Webové mapové služby

Webová mapová služba (dále jen WMS) je standard definovaný organizací Open Geospatial Consortium [6]. Tento standard je založen na architektuře klient-server. Primárním účelem služby je poskytování obrazových dat vygenerovaných z databáze geografického informačního systému, avšak umožňuje získat i doplňující data vztahující se ke zvolenému bodu určeného pomocí vhodných souřadnic. Tato funkcionalita je však volitelná a záleží na konkrétní implementaci WMS [7].

Každá služba musí implementovat dva povinné dotazy (GetCapabilites, GetMap) a jeden volitelný (GetFeatureInfo). Popis jednotlivých dotazů obsahuje Tabulka 1.

Tabulka 1: Typy dotazů na WMS [7]

NÁZEV DOTAZU	POPIS
GetCapabilities (povinné)	Vypíše metadata o dané službě (např. verze, maximální šířka a výška pro GetMap dotaz atd.).
GetMap (povinné)	Vrátí obrázek dotazované mapy. Oblast musí být specifikována šířkou a výškou.
GetFeatureInfo (volitelné)	Slouží pro získání více informací o bodu definovaném v těle dotazu.

Pro Českou republiku existuje několik poskytovatelů WMS. Jedním z nich je například Česká geologická služba. Ta obsahuje několik desítek tematicky rozdělených WMS (např. geologie, hydrologie, půdy apod.). Každá služba pak slouží pro zobrazení odlišných dat [8].

Jako dalšího poskytovatele takovýchto služeb si můžeme uvést Ředitelství silnic a dálnic pro ČR. Na jejich portálu můžeme najít WMS poskytující informace o odpočívkách na dálnicích ČR, aktuální a plánované uzavírky na dálnicích a silnicích v ČR nebo také službu obsahující informace o celé dálniční a silniční síti [9].

Jak si můžeme všimnout, všechny tyto služby jsou cenným zdrojem informací. Pro potřeby této práce byly využity služby dostupné na portálu Českého úřadu zeměměřického a katastrálního (dále jen ČÚZK), který dává zdarma přístup ke zmapovanému území celé České republiky. Poskytované služby jsou přístupné buď za pomocí WMS klienta nebo jako ArcGIS Server, se kterým se komunikuje pomocí REST či SOAP rozhraní [10].

2.1 Geografický informační systém a platforma ArcGIS

Geografický informační systém (dále jen GIS) je pojem, jehož definice se autor od autora liší. Nejčastěji se však můžeme setkat s definicí, kterou vyslovil Burrough [11]: *Geografický informační systém je souborem prostředků pro sběr, ukládání, vyhledávání, transformování a znázorňování dat z reálného světa s ohledem na speciální účely jeho použití.* Takovýto systém lze pak uplatnit v mnoha oblastech. Například v logistice lze pomocí GIS optimalizovat trasy a tím šetřit výdaje firem na provozní náklady. GIS je užitečný také při vytváření evakuačních plánů v oblastech ohrožených hurikány nebo jinými živly [12].

ArcGIS [13] je platforma pro geografickou analýzu a mapování. Samotný ArcGIS je distribuován v mnoha podobách:

- ArcGIS Desktop – desktopová verze umožňující například vizualizace, analýzy map, plánování tras atd.
- ArcGIS Enterprise – verze pro společnosti, obsahuje vlastní server a možnost sdílet data uvnitř společnosti.
- ArcGIS for Developers – verze pro vývojáře, ArcGIS poskytuje kompletní SDK pro vývoj aplikací v několika programovacích jazycích (například Java, .NET, Python a další) a mnoho dalších distribucí [14].

Převážná většina funkcionalit ArcGISu je publikována skrze webové služby. To je také důvodem, proč existuje takové množství rozdílných distribucí. Zároveň lze pomocí

ArcGIS REST API vyvíjet vlastní webové služby. Jedním z takových případů jsou i REST služby od ČÚZK, které byly využity k realizaci této diplomové práce.

2.2 REST služby poskytované ČÚZK

Jak již bylo zmíněno v kapitole 2, ČÚZK má zmapované území celé České republiky a všechna tato data bezplatně poskytuje prostřednictvím svých REST služeb. Důležitými jsou pro nás analýzy výškopisu pro požadovanou oblast. Pro tento účel jsou k dispozici dvě služby: Digitální model povrchu České republiky 1. generace (DMP 1G) a Digitální model reliéfu České republiky 5. generace (DMR 5G). ČÚZK také poskytuje mapy formou digitálních fotografií skrze službu ortofoto. Každá z těchto služeb využívá souřadnicový systém S-JTSK v Křovákově zobrazení, což je zobrazení používané pro Českou a Slovenskou republiku. Pro práci s nimi jsou k dispozici různé operace. Každá má odlišné použití i odlišný způsob práce s ní.

Komunikace s jednotlivými službami probíhá pomocí POST metody. Při dotazu na službu je nutné zadat validní URL vybrané operace, ve které se specifikuje oblast, pro kterou chceme získat data (výškopisy, digitální fotografii apod.). Oblast může být specifikována více způsoby – sadou bodů, obdélníkem, jedním bodem, polygonem nebo lomenou čarou. Souřadnice každého z bodů pak musí být zadána ve správném formátu pro S-JTSK souřadnice. V odpovědi, kterou služba zašle, se nachází data pro námi vybranou oblast v JSON formátu [15].

2.2.1 DMP 1G

DMP 1G [16] představuje zobrazení celého území České republiky zahrnující stavby a rostlinný porost. Data byla pořizována metodou leteckého laserového skenování. Pro přesně vymezené objekty (budovy) je úplná střední chyba výšky 0,4 m, pro neohraničené objekty (lesy, stromy a jiné rostlinné porosty) pak 0,7 m. Tento model vznikl v letech 2009 až 2013.

Operace dostupné v této službě (viz Tabulka 2) můžeme použít například pro získání vzorků výškopisných dat pro specifikovanou oblast, vzorku pro jeden bod nebo třeba ke spočítání zastoupení výškových bodů formou histogramu.

Tabulka 2: Operace nad službami DMP 1G a DMR 5G [15]

NÁZEV OPEARCE	FUNKCIONALITA
Export Image	Generuje obrázek vybrané oblasti.
Measure	Měří výšku, vzdálenost, směr, plochu a obvod vybrané oblasti.
Compute Histograms	Spočítá zastoupení výškových bodů ve vybrané oblasti.
Compute Statistics Histograms	Obdobné jako Compute Histograms. Navíc analyzuje oblast a jako odpověď odešle minimální a maximální výšku ve zvolené oblasti.
Get Samples	Vytvoří vzorky pro zvolenou oblast. Lze zadat vzdálenost a celkový počet požadovaných vzorků.
Identify	Identifikuje bod nebo střed vybrané oblasti a vrátí informace o tomto bodu.

2.2.2 DMR 5G

DMR 5G [17] je na rozdíl od služby DMP 1G zobrazení území přirozeného nebo lidskou činností neupraveného zemského povrchu. Model byl vytvořen pomocí leteckého laserového skenování s úplnou střední chybou výšky 0,18 m v odkrytém terénu a 0,3 m v zalesněném terénu. Tato data byla pořízena v letech 2009 až 2013. Podporované operace pro tuto službu jsou stejné jako pro DMP 1G (viz Tabulka 2).

2.2.3 Ortofoto

Služba ortofoto [18] poskytuje ortofotomapu České republiky. Fotografie jsou poskytované formou mapových dlaždic především z důvodu optimalizace rychlosti. Výsledné fotografie jsou poskytované v mnoha formátech (PNG, JPG, SVG atd.) a volitelné

velikosti, která je omezena na maximální velikost 4096 pixelů pro šířku i výšku fotografie. Hlavní operací pro tuto službu je *GetMap*, která se využívá pro získání digitální fotografie zvolené oblasti. Výběr oblasti se provádí zadáním dvou bodů, které dohromady určují obdélníkový výřez. Vzdálenost těchto bodů pak udává měřítko získané fotografie – pokud body utvářejí malý výřez, měřítko bude také malé a výsledek detailnější, pokud se naopak vybere velká oblast, měřítko bude větší a detaily se sníží.

2.3 Projekt OpenStreetMap

Projekt OpenStreetMap (dále jen OSM) je GIS, do jehož databáze přispívají miliony lidí po celém světě. Každý, kdo se rozhodne, může data vytvářet, využívat a upravovat pod podmínkou uvedení licence (Open Database License) a zdroje dat. K vytváření záznamů využívají dobrovolníci GPS lokátory, pomocí kterých ukládají do databáze informace o poloze, typu (budovy, stromy, silnice a mnoho dalších objektů reálného světa) a doplňující metadata o jednotlivých entitách [19]. Poloha každé entity je udána jedním nebo více uzly – bod definovaný souřadnicemi v souřadnicovém systému WGS84 (latituda a longituda) a identifikátorem uzlu [20]. Možnost libovolného přispívání do databáze s sebou přináší rizika ve formě vzniku chybných dat. Může se tak například stát, že má budova nepřesně určený půdorys či špatně zadaná metadata.

Pokud chceme pracovat s OSM za účelem získání dat, existuje REST webová služba. Pomocí ní lze provádět dotazy na OSM databázi. Poskytovaných služeb existuje celá škála. Lze se dotazovat na jednotlivé entity na mapě pomocí jejich identifikátoru. Především pak lze službu využít k získání všech entit zastoupených ve vybrané oblasti ve formátu XML [21].

2.4 Shrnutí

Jelikož se nepodařilo najít službu, která by byla dostupná a sama o sobě by poskytovala všechna potřebná data (výškové profily, půdorysy budov, informace o entitách atd.),

je potřeba využít obou výše zmiňovaných služeb. Data budou použita k naplnění datového modelu, jehož charakteristika a struktura je popisována dále v této práci.

Je tedy nutné vhodnými postupy data zkombinovat tak, aby splňovala požadavky datového modelu a aby mohla být využita v cílovém zařízení a dalším klientském softwaru. Drobnou nevýhodou tohoto přístupu je fakt, že každá služba využívá jiného souřadnicového systému, a proto je potřeba souřadnice převádět (z S-JTSK do WGS84 a naopak). Pokud však aplikujeme vhodné metody pro přepočet, měli bychom mezi nimi docílit přesných převodů.

Data dostupná z každé z těchto služeb jsou dostatečně přesná. Proto mohou být využita pro implementaci a především v cílovém ANS. Ten by na základě dat měl být schopný správně vyhodnocovat pohyb ve vybrané oblasti. Samozřejmostí pro ANS je implementace doplňujících metod, jako jsou kamery, gyroskop, akcelerometr aj., pomocí kterých se dokáže v případě potřeby orientovat ve svém prostředí.

3 Java 3D

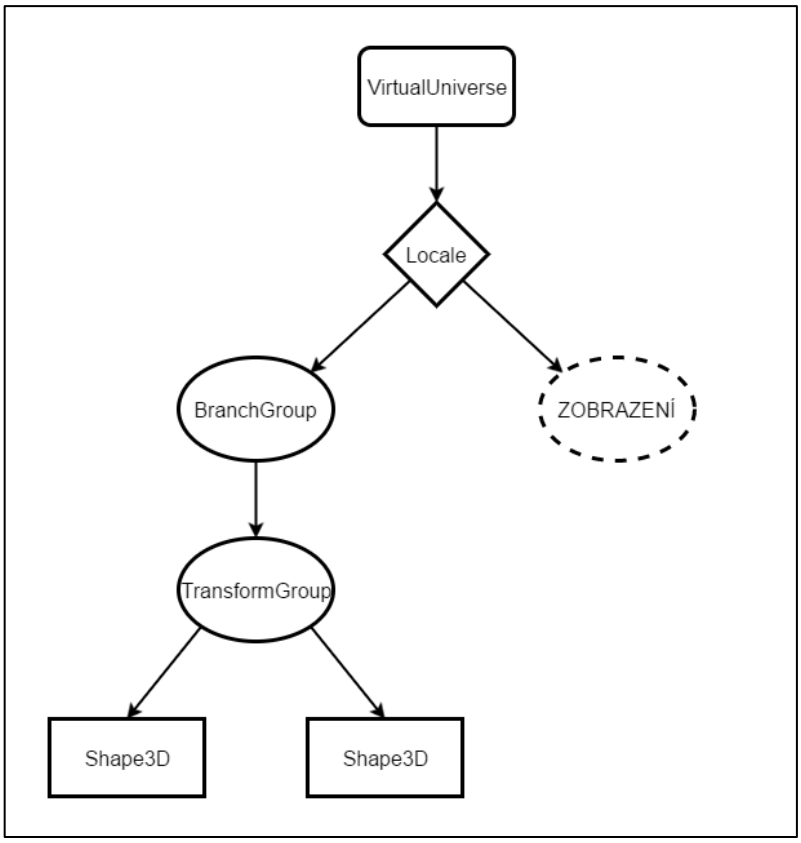
Java 3D je API pro vytváření interaktivních 3D aplikací v jazyce Java. Do verze 1.6 používá programovací rozhraní OpenGL nebo DirectX pro vykreslování objektů. Od verze 1.6 a výše pak využívá JOGL (Java OpenGL – knihovna umožňující volat funkce OpenGL při programování v jazyce Java). Java 3D je nejen nadstavbou nad funkcemi OpenGL, ale je plnohodnotným, rychlým a dobře řešeným nástrojem například pro vytváření profesionálních 3D her nebo pro tvorbu vizualizací vědeckých výpočtů.

Jednou z hlavních výhod je možnost psát aplikace čistě v jazyce Java. Java 3D je navíc plně objektově orientovaná. To je velkou výhodou v porovnání s programováním v čistém OpenGL s využitím jazyka C, který pracuje přímo s grafickým procesorem na té nejnižší úrovni a objektový přístup zde prakticky neexistuje [22].

3.1 Struktura Java 3D aplikace

Všechny objekty, které mají být ve scéně vykresleny, se v Java 3D uchovávají ve stromové struktuře – tzv. graf scény (viz Obrázek 1). Tato struktura je orientovaným acyklickým grafem. Jejím kořenem je objekt *VirtualUniverse* (ve zjednodušené podobě pak *SimpleUniverse*), který odkazuje (ať už přímo nebo nepřímo) na všechny ostatní objekty ve scéně. Přímým potomkem z pravidla bývá jeden nebo více objektů *Locale*. Ty udávají počátek souřadné soustavy pro všechny své potomky.

Objekty, které jsou potomkem *Locale* už mohou být poměrně rozmanité – objekty definující vykreslované prvky, skupiny, transformační skupiny a mnoho jiných [23].



Obrázek 1: Ukázka grafu scény v Java3D [23]

3.1.1 Typy objektů ve scéně

Všechny objekty jsou odvozeny od třídy *Node* (uzel). Podle typu a účelu použití se uzly dělí na:

1) **Řídící uzly** – do této skupiny patří uzly jako jsou *Locale*, *BranchGroup*, *TransformGroup* a několik dalších. Jejich hlavním cílem je seskupovat potomky (další řídící uzly nebo geometrické uzly).

2) **Geometrické uzly** – objekty viditelné (vykreslené) ve scéně (*Shape3D*, *Background* atd.).

3) **Uzly ovlivňující chování** – uzly jako jsou *Behavior*, *Light*, *Sound* a další určují výsledné chování aplikace. Obvykle nejsou přímo vázány na geometrické objekty ve scéně, ale jistým způsobem je mohou ovlivňovat (např. osvětlovat).

Důležitými jsou objekty odvozené od třídy *Group* (spadají mezi „Řídící uzly“). Ty jsou určeny pro seskupování ostatních objektů. Pomocí třídy *TransformGroup*, která sem také spadá, můžeme objekty škálovat, posouvat a otáčet. Pokud nějakou z těchto operací pro skupinu specifikujeme, promítne se na všechny její potomky.

Při vykreslování objektů (ať už se jedná o pouhý kvádr, kouli nebo nějaký složitější objekt, jako je například lidská tvář) využíváme objekt *Shape3D*. Ten v sobě nese dvě základní informace. První z nich specifikuje umístění jednotlivých vrcholů v prostoru, které tvoří výsledný objekt. Druhá pak definuje to, jak objekt bude vypadat – materiál, barva, otexturování objektu a další vlastnosti určující vzhled [24].

3.2 Silné a slabé stránky Java 3D aplikací

Mezi nejsilnější stránky [25] patří organizace objektů do grafu scény (viz kapitola 3.1), který dává programátorovi možnost lépe organizovat, spravovat a upravovat celou scénu. Díky tomu se zjednodušuje celý proces vytváření 3D aplikací. Java 3D byla vyvíjena především s důrazem na výkon, který je zlepšován optimalizacemi na dvou úrovních:

- na nejvyšší úrovni se optimalizuje práce s grafem scény,
- naopak na té nejnižší se optimalizují procesy vykonávané přímo nad OpenGL (případně DirectX).

Pokud bychom chtěli s pomocí Javy 3D tvořit aplikace, kde se klade maximální důraz na opravdu rychlé vykreslování, je nepochybně lepší volbou zvolit jazyk C a programovat v čistém OpenGL. I přes to, že Java 3D využívá mnoho optimalizací pro zlepšení výkonu, stále je tu fakt, že nepracujeme přímo s grafickou kartou (jak je tomu u OpenGL) a je výkonově slabší. Pomineme-li tento fakt, je Java 3D vhodnou volbou při tvorbě většiny grafických aplikací.

4 Návrh řešení

Aplikaci jsem rozdělil na dva od sebe oddělené celky podle jejich funkcionality. Prvním je aplikace pro práci s mapou, která umožňuje specifikaci oblasti, ve které by se mělo cílové zařízení pohybovat. Druhá část slouží k vytvoření 3D vizualizace oblasti, která byla v předchozím kroku vybrána. Vizualizační aplikace je plně závislá na aplikaci pro výběr oblasti, jelikož vizualizace se provádí na základě dat, která jsou v prvním kroku získána.

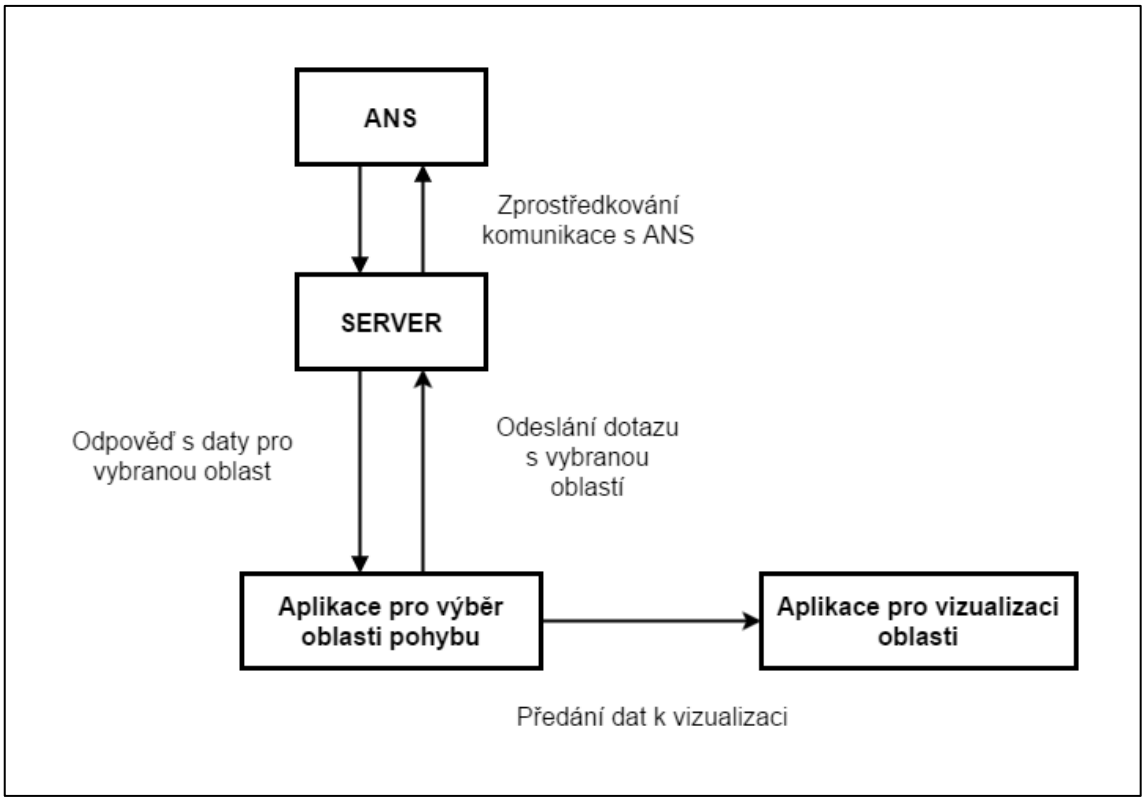
Aplikace využívá architektury klient-server. Klientskou část představuje řešení popisované v této práci. Serverová část je implementována na vzdáleném serveru. Využívá aplikační server Tomcat a pro práci s ním poskytuje REST API. Server zajišťuje zpracování dat ze služeb ArcGIS a OpenStreetMap, které je poměrně časově a výkonově náročné.

Server navrhl a implementoval Bc. Jan Kozánek [26], se kterým jsem v průběhu implementace spolupracoval. Jelikož se naše práce v určitých bodech shodovaly, napsal jsem pro jeho server metodu, která zajišťuje práci s ArcGIS REST Serverem a získává data z tohoto serveru v požadovaném formátu. Tato metoda byla pro správnou funkčnost obou řešení klíčová. Její implementace je popsána v kapitole 5.1.

4.1 Komunikační schéma

Jak již bylo zmíněno výše, aplikace využívá architektury klient-server. Veškeré zpracování dat probíhá na straně serveru. Ten navíc slouží jako hlavní uzel, který propojuje veškerý klientský software a zařízení mezi sebou. V případě mé aplikace, která určuje oblast pohybu pro koncové zařízení, tak nekomunikuji přímo se zařízením, ale odešlu dotaz na server. Dotaz se zpracuje a server zašle odpověď zpět do mé aplikace a zároveň stejná data zašle do zařízení. Tato odpověď obsahuje data, která popisují mnou vybranou oblast. Každá aplikace, která je v tomto celku implementována, musí přes tento server komunikovat. Současnou strukturu komunikace znázorňuje Obrázek 2.

Serverová komponenta má navíc za úkol data předzpracovávat a cachovat. Tato funkcionalita vychází z předpokladu, že se cílová oblast mění jen zřídka. Pomocí cachování lze docílit značného zrychlení při zpracování dotazů na server. Více o implementaci a návrhu serveru se můžete dočíst v [26].



Obrázek 2: Komunikační schéma

4.2 Datový model

Návrhu datového modelu předcházela analýza dostupných služeb. Na ně bylo kladeno několik požadavků. Hlavní byl kladen především na maximální možnou přesnost a úplnost získaných dat. Dalším pak byla dostupnost těchto služeb. Existuje několik služeb, které by data v námi požadovaném formátu poskytovaly. Avšak ty jsou určeny buď k vojenským účelům (k běžnému použití nedostupné) nebo jsou komerční (pro nás příliš nákladné). Z tohoto důvodu jsme se uchýlili k využití volně dostupných služeb

(viz kapitola 2.4) za cenu toho, že je nutné data pro výsledný datový model různými způsoby zpracovávat (například kvůli použití rozdílných souřadnicových systémů) kombinovat.

4.2.1 Konstrukce datového modelu

Aby mohl být datový model zkonstruován, musí se vhodně zkombinovat dostupná data. Výsledný datový model by měl obsahovat následující informace:

- výškový profil reliéfu krajiny,
- entity vyskytující se v oblasti (budovy, silnice, stromy apod.),
- výšky jednotlivých entit,
- satelitní snímek dané oblasti.

Všechny údaje lze tedy získat buď z ArcGIS nebo OSM služeb. Jelikož máme dva různé typy cílových zařízení – pozemní a letecké prostředky, je nutné využívat odlišný datový model pro každý z nich. Ten vychází primárně z toho, jak se jednotlivá zařízení mohou pohybovat. V této práci budeme využívat datový model pro drony, ale pro úplnost si uvedeme charakteristiku modelu jak pro drony, tak i pro pozemní zařízení.

Drony mohou realizovat pohyb po všech třech osách. Osy X a Y určují aktuální polohu zařízení. Hodnota na ose Z pak udává výšku, ve které se nachází. Z toho se odvíjí struktura výsledného datového modelu. Ten v sobě nese informace o oblasti v podobě sítě bodů. Každý bod má definované právě souřadnice X, Y a Z. Pomocí datového modelu bychom měli být schopni rozlišit tři typy oblastí – průchozí, neprůchozí a zakázané.

Neprůchozí lokací chápeme takovou, kterou nelze dronem přeletět či podletět. Jedná se tedy o fyzickou překážku, u které hrozí riziko kolize. Zakázanými jsou takové, na které je kladeno dočasné nebo trvalé omezení pohybu pro drony (například z legislativních důvodů). Mezi ty trvale zakázané zóny spadají například prostory v okolí jaderných elektráren nebo muničních skladů. Dočasné omezení může být na oblast kladeno

například z důvodu konání nějaké události. Poslední, průchozí, jsou pak všechny ostatní zóny, ve kterých se lze volně pohybovat.

V případě datového modelu pro pozemní zařízení můžeme zjednodušeně říci, že se jedná o stejný datový model, který využíváme u dronů. Je zde pouze pár rozdílů. První je v tom, že pozemní prostředky nevyužívají pohyb po ose Z, ale jen po osách X a Y. Hodnota Z je v datovém modelu obsažena, ale pouze poskytuje doplňující informaci o výškovém charakteru oblasti.

Dalším rozdílem je chápání průchozích zón. Mezi ně spadají především silnice, stezky, zpevněný terén a jiné entity, po kterých by se zařízení mohlo pohybovat. Neprůchozími jsou pak každé takové, které lze považovat za výškovou překážku (plot, budova, strom atd.). Zakázané zóny se ve své podstatě nemění.

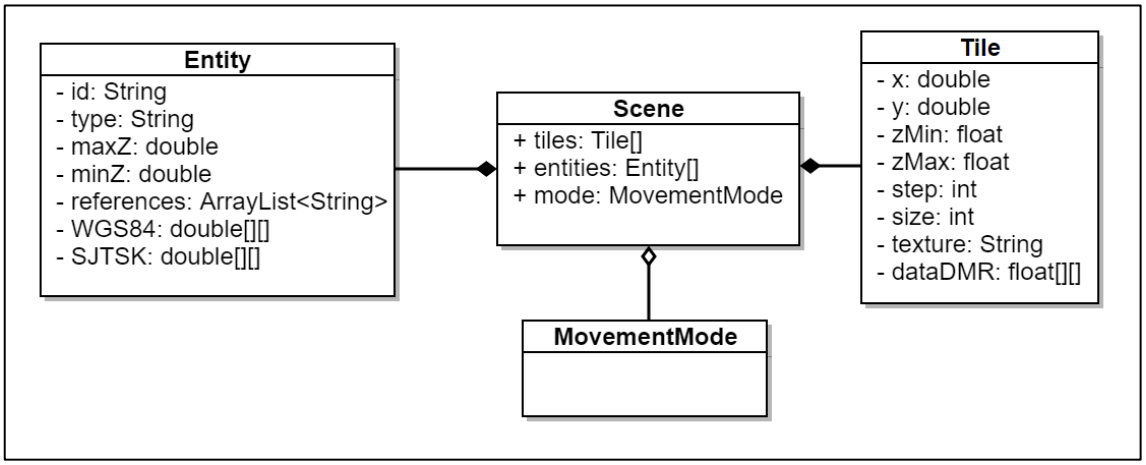
Výsledná konstrukce obou zmiňovaných datových modelů se rozpadá do sedmi pevně stanovených kroků:

1) Výběr oblasti a režimu pohybu pomocí desktopové aplikace. Pro tuto oblast jsou pak zpracovávána data do datového modelu.
2) Získání dat popisujících výškovou charakteristiku reliéfu.
3) Získání informací o všech entitách, které do vybrané oblasti zasahují.
4) Pro každou entitu zjistit její maximální a minimální Z souřadnici.
5) Získání satelitního snímku pro oblast v požadovaném rozlišení.
6) Sestavení datového modelu a naplnění daty.
7) Export modelu v binární nebo serializované podobě do cílového zařízení a do klientských softwarů, které data potřebují.

4.2.2 Objekty datového modelu

Celý datový model (viz Obrázek 3) je rozdělen do několika objektů (tříd). Aby cílové klientské zařízení nebo aplikace mohla tento datový model využívat, musí jej ve své implementaci reflektovat (implementovat shodné třídy jako jsou na serveru). Tzn. každý

takový klient u sebe musí uchovávat to, jak jednotlivé třídy vypadají, aby s nimi mohl dále pracovat.



Obrázek 3: Class diagram datového modelu

Třída Tile

Vybraná oblast je rozdělena na dlaždice (objekt *Tile*) o stejné velikosti, která je nastavena na 128×128 S-JTSK jednotek. Každá dlaždice je jednoznačně identifikována souřadnicí levého horního bodu v souřadnicovém systému S-JTSK. Konstantní velikost každé z nich je využita z důvodu cachování do databáze a předzpracování na straně serveru.

Každá dlaždice v sobě nese také informaci o tom, jaká je její minimální a maximální hodnota Z souřadnice reliéfu. Hlavní položkou tohoto objektu je již zmiňovaná síť výškových bodů (data DMR). Ta je uchovávána v dvourozměrném poli, ve kterém první položka odpovídá souřadnici dané dlaždice. Poslední položka v poli odpovídá hodnotě souřadnicím dlaždice posunutým o 128 na obou osách (na ose X přičítáme 128, na ose Y pak odečítáme 128).

Dlaždice v sobě také obsahuje satelitní snímek odpovídající dané oblasti na mapě. Doplňujícími informacemi pro danou dlaždici jsou její velikost a krok mezi vzorky výškového profilu. Pro konstrukci dlaždic dané lokace jsou využívány služby z ArcGIS

serveru ČÚZK. Konkrétně se jedná o službu DMR, která se využívá pro konstrukci sítě výškových bodů a ke zjištění minimální a maximální hodnoty Z souřadnice.

Druhou použitou službou je ortofoto, ze které se stahuje satelitní snímek v požadované velikosti pro danou dlaždici ve formátu JPG. Ten je později pomocí kódování Base64 převeden do textové podoby. Ten je vhodný jak pro uložení do databáze na straně serveru, tak i pro umožnění následné serializace a odesílání po síti klientským zařízením.

Třída Entity

Druhým objektem datového modelu je entita (třída *Entity*). Ta představuje objekty reálného světa (budovy, stromy atd.), které se ve vybrané oblasti vyskytují. Entita je určena množinou bodů. Ty jsou zde uloženy jak v souřadnicovém systému S-JTSK, tak i ve WGS84. Tím se zabrání potřebě převádět mezi souřadnicovými systémy až na cílovém zařízení a sníží se tak zátěž na ně kladená.

Každý bod je určen jen souřadnicemi X a Y. Hodnota souřadnice Z se pro každou entitu ukládá pouze jako minimální a maximální hodnota. Každá entita má určitý typ – budova, silnice, plot a mnoho dalších. Ten mimo jiné určuje to, jak je entita definována. Například budova je určena množinou bodů, kde první a poslední bod je stejný. Tato množina tedy definuje půdorys dané budovy. Silnice a cesty jsou určeny také body, ale ty dohromady tvoří soustavu úseček podle toho, kudy silnice vede.

Pro konstrukci entity se využívá několika dříve zmiňovaných služeb:

1) Body tvořící entitu jsou získány z OSM. Zároveň jsou převedeny z WGS84 do S-JTSK. Definice pomocí obou souřadnicových systémů jsou pak v entitě uloženy pro pozdější použití.

2) Maximální Z souřadnice (výška entity) je získána ze služby DMP 1G pomocí metody *Compute Statistics Histograms*. Metoda přebírá jako parametr definici oblasti, kterou poskládáme z bodů získaných z OSM (body musí být v S-JTSK formátu).

3) Pro zjištění minimální Z souřadnice (minimální výška reliéfu v dané oblasti) se využívá služba DMR 5G. Opět se za použití metody *Compute Statistics Histograms* dotazujeme na oblast, která je definována body z OSM.

Třída Scene

Posledním objektem je třída *Scene*. Jako taková nám představuje vybranou oblast pohybu. Spolu s režimem pohybu (jejich typy a funkce jsou popsány níže), který je pro aktuální výběr zvolen, slučuje dlaždice a entity do pole. Tento objekt je určen především pro vytvoření výstupu ve formátu JSON, který je vytvořen pomocí deserializace všech objektů a následně odeslán cílovým klientským zařízením.

4.3 Režimy pohybu ANS

Cílové zařízení potřebuje mít ke své činnosti definovaný způsob, jakým se bude ve vybrané oblasti pohybovat. K tomu slouží režimy pohybu. Jsou dva způsoby, jak lze specifikovat tento režim. Prvním je zadání příkazu do již inicializovaného zařízení pomocí dálkového ovládání. Druhý způsob pak využívá klientský software, popisovaný v této práci. Ve všech režimech zařízení potřebuje data popisující vybranou oblast pohybu, kvůli správnému vyhodnocování svého okolí a zamezení případným kolizím. V současné době lze zadat tři různé režimy.

4.3.1 Režim náhodného pohybu

Z hlediska požadavků kladených na definici tohoto režimu se jedná o nejjednodušší režim. Zařízení potřebuje ke svému pohybu pouze data vztahující se k definované oblasti pohybu, aby mohlo plnohodnotně rozhodovat o tom, kam se může či nemůže přemístit. Nespoléhá přitom pouze na datový model, ale i na nainstalované senzory, které jsou nezbytnou součástí pro validní rozpoznávání svého okolí. Výpočet náhodné trasy pohybu pak probíhá již v samotném zařízení a není potřeba žádného vnějšího zásahu.

4.3.2 Režim pohybu po předem stanovené trase

S volbou tohoto režimu se aktivuje pohyb zařízení po trase, kterou definujeme na straně klientského softwaru. Vstupem je tedy datový model společně s množinou bodů, které definují vybranou trasu. Zařízení svým pohybem kopíruje specifikovanou trasu. Datový model a nainstalované senzory využívá k rozpoznávání entit, u kterých by mohla nastat případná kolize (vysoké budovy, stromy, ploty atd.).

4.3.3 Režim pohybu do cílového bodu

Tento režim můžeme přirovnat k režimu náhodného pohybu popsaném v 4.3.1. Rozdíl je zde v tom, že zařízení dostane zadaný bod, do kterého se má dostat. Veškerý výpočet trasy bude zařízení provádět samo, dle charakteristiky datového modelu pro vybranou oblast pohybu.

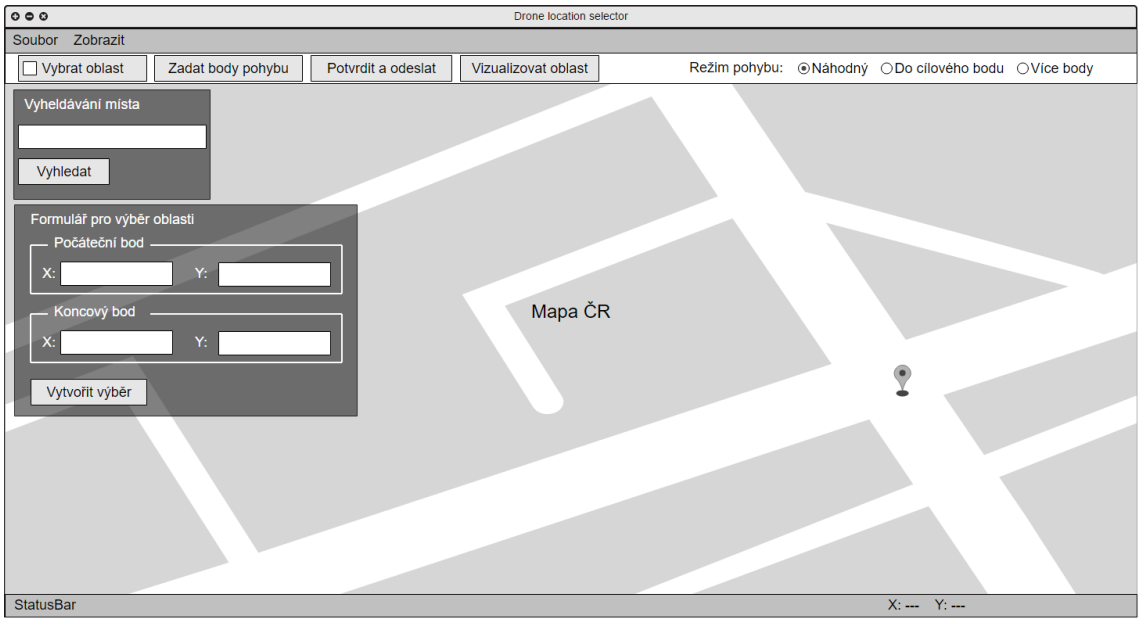
4.4 Aplikace pro výběr oblasti pohybu

Tato aplikace představuje první část navrhovaného řešení. Zastává úlohu klientského softwaru, který komunikuje se serverovou komponentou.

Hlavním úkolem aplikace je umožnit výběr oblasti na mapě České republiky. Ve vybrané oblasti se pak bude cílové zařízení pohybovat. Pro implementaci jsem využil ArcGIS Runtime SDK for Java 10.2.4, která je poskytována společností Esri jako jedna z knihoven platformy ArcGIS for Developers. Díky těmto knihovnám lze vytvářet aplikace v jazyce Java, které využívají mapy, provádět nad nimi analýzy dat, vyhledávat v nich a provádět nespočet dalších operací. Pro vytvoření GUI jsem využil knihovny Swing. Pomocí ní lze pracovat právě s komponentami s ArcGIS SDK a vkládat je do GUI aplikace.

4.4.1 Návrh grafického rozhraní

Nejdříve byl navržen drátěný model (viz Obrázek 4), podle kterého se následná implementace odvíjela. Díky modelu se také stanovily požadavky na jednotlivé funkce, které by aplikace měla mít implementované.



Obrázek 4: Drátěný model aplikace pro výběr oblasti pohybu

Na obrázku výše můžeme vidět, že GUI obsahuje kromě mapy také sadu tlačítek a textových polí, které realizují požadované funkce. Pro kreslení objektů v mapě jsou k dispozici dva nástroje. Prvním je nástroj pro výběr oblasti pohybu aktivovaný tlačítkem „Vybrat oblast“. Tím lze do mapy zakreslit obdélníkový výběr představující oblast pohybu pro cílové zařízení.

Po výběru oblasti je možnost specifikovat body pohybu. Ty však lze uvést pouze u režimů „Do cílového body“ a „Více body“, aktivovanými pomocí tlačítek typu *RadioButton*. Každý z bodů definující trasu musí být uvnitř vybrané oblasti. Proto je bezpodmínečně nutné mít nejdříve určenou oblast pohybu (toto chování je ošetřeno povolením/zakázáním možnosti zadávat body pohybu tlačítkem „Zadat body pohybu“).

Po validním výběru a zadání trasy pro výše zmíněné režimy lze odeslat dotaz na server. Tam se data zpracují a aplikace obdrží odpověď obsahující všechna data vztahující se k vybrané oblasti. Zároveň se bezprostředně po obdržení odpovědi začnou zpracovávat přijatá data pro 3D vizualizaci. Po zpracování je pak vyvoláno nové okno s vykreslenou 3D podobou oblasti.

Aplikace také umožňuje vyhledávat libovolná místa na území ČR (města, ulice, hory atd.). Když je vyhledávání úspěšné je mapa automaticky zaostřena na vyhledaný výsledek. Další funkce, která je k dispozici, je určena ke specifikaci oblasti pohybu zadáním počátečního a koncového bodu, které budou oblast určovat. Body musí být v souřadnicovém systému S-JTSK a oblast je automaticky dopočítávána tak, aby spadala do mřížkového systému.

V aplikaci lze také provádět pouhou vizualizaci oblasti bez posílání příkazu a dat do cílového zařízení. Podmínkou zde je však přítomnost validního JSON souboru, který obsahuje data popisující nějakou oblast na mapě (soubor musí mít stejný formát jako je formát odpovědi ze serveru).

4.4.2 Zobrazení a nastavení mapy

Zobrazení mapy je realizováno vytvořením objektu *JMap*. Mapa může mít mnoho typů zobrazení (satelitní, topografická, mapa oceánů a další), záleží jen na tom, jaký zvolíme WMS, ze kterého mapové podklady chceme využívat. Použitý WMS nám zároveň určuje, jaký souřadnicový systém bude objekt mapy interně využívat.

Jelikož se soustředíme na území České republiky, využijeme dříve zmiňovaný WMS od ČÚZK. Tyto služby mají nastaven souřadnicový systém S-JTSK, tudíž naše aplikace bude interně tento systém také využívat. Ten se také jevil jako jediný vhodný, jelikož veškerá komunikace a ukládání objektů na straně serveru využívá souřadnicového systému S-JTSK. Díky tomu se zbavíme na klientské straně nutnosti přepočtu souřadnic z jednoho souřadnicového systému do druhého (z WGS84 do S-JTSK a naopak). To bude mít vliv nejen na rychlost, ale zároveň tím docílíme maximální přesnosti, kterou bychom při převodech souřadnic ztráceli.

4.4.3 Zvýšení efektivnosti cachování

Veškeré dotazy jsou na serveru ukládány do databáze z důvodu pozdějšího použití. Tím se docílí zvýšení rychlosti při odpovědích na dotazy klientů, jelikož není potřeba dříve dotazované oblasti znovu zpracovávat. Aby bylo cachování co nejefektivnější, je nutné zavést systém, který by zajistil efektivní ukládání a zpracování jednotlivých dlaždic pro zvolenou oblast.

Po vzájemné dohodě s Bc. Janem Kozánkem, který pracoval na serverové komponentě, jsme dospěli k řešení, kdy využijeme pro oblast České republiky mřížkový systém. Ten rozdělí celé území na pomyslné dlaždice s konstantní velikostí. Při realizaci výběru oblasti pohybu se bude úvodní dlaždice výběru „přichytávat“ k nejbližší vyhovující dlaždici, která do mřížky zapadá. Všechny ostatní dlaždice, které budou na tu úvodní navazovat, budou již automaticky ve stanovené mřížce. Díky tomuto systému pak budou do databáze ukládány pouze unikátní záznamy pro každou dlaždici a nestane se, že by se některé dlaždici překrývaly. Odpadne tím také nutnost kontroly, zda se přijatá vybraná oblast nepřekrývá s již uloženou dlaždicí jen v nějaké její části. Samotné implementaci a detailnějšímu popisu se věnuji v kapitole 6.1.

4.5 Aplikace pro vizualizaci vybrané oblasti

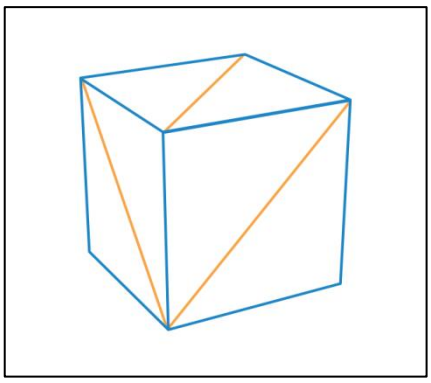
Na základě dat získaných ze serveru po realizaci výběru oblasti zájmu lze danou oblast vizualizovat. K tomu slouží tato druhá část navrhovaného řešení. Výsledkem vizualizace je 3D scéna. Ta přímo reflektuje oblast, která byla vybrána na mapě ČR. Pro realizaci vykreslení všech potřebných objektů jsem využil Java 3D API probírané v kapitole 3.

Vizualizaci jsem navrhl tak, aby byla pokud možno co nejvíce interaktivní – s celou scénou lze libovolně otáčet, přibližovat, zobrazovat a skrývat vybrané objekty. U všech těchto operací jsem naplno využil sílu toho, jak je Java 3D implementována. Právě díky grafu scény je možné se scénou manipulovat pomocí poskytovaných nástrojů, které buď využijeme tak, jak jsou nebo je lze libovolně modifikovat, aby vyhovovaly našim potřebám.

4.5.1 Postup vykreslení scény

Vytvoření jakéhokoliv objektu (pokud se nejedná o geometrická primitiva jako je koule, kvádr, válec a kužel, která má Java 3D již implementována) předchází zpracování a příprava dat. Je potřeba si zvolit vhodnou datovou strukturu pro reprezentaci objektu, který má být vykreslen. Od toho se pak odvíjí postup, jakým je objekt konstruován. V zásadě se jedná o určení polohy každého vrcholu v rámci scény. Tyto vrcholy pak tvoří výsledný objekt, který může být následně vložen do scény a vykreslen.

Samozřejmě i zmiňovaná geometrická primitiva využívají tohoto přístupu, kdy se do vhodně zvolené datová struktury specifikují jednotlivé vrcholy tvořící tvar. Jelikož se však jedná o jednoznačně určené objekty, u kterých se mění pouze jejich velikost, úroveň detailů nebo poloměr, tak je celý tento proces již implementován uvnitř Java 3D API. Jako příklad si zde můžeme uvést reprezentaci obyčejné krychle (viz Obrázek 5). Pokud si vykreslíme drátěnou reprezentaci tohoto objektu, uvidíme, jak je ve skuteč-nosti strukturován. Můžeme vidět, že Java 3D pro vykreslení jednotlivých stěn nevyužívá čtverců, nýbrž trojúhelníků, které vznikly rozkladem čtverce (tzv. triangulace).



Obrázek 5: Reprezentace krychle v Java 3D

Pokud chceme vykreslit složitější objekty (například terén, lidskou tvář aj.) využívá se převážně tohoto přístupu, kdy se objekt rozdělí na mnoho menších trojúhelníků. Ty pak mohou vytvářet dojem zaoblení objektu.

Ve scéně, kterou tato aplikace vytváří, se nachází několik typů objektů. Pro každý z nich je využit jiný způsob vykreslení. Tím se konstrukce scény rozpadá do čtyř následujících kroků:

1) Příprava dat pro konstrukci terénu a následné vykreslení. Tento krok je nezbytně nutné udělat jako první, jelikož si zde určuji minimální a maximální vyvýšení terénu. Vůči těmto hodnotám pak upravuji výšky všech dalších vykreslovaných objektů. Tím zajišťuji zachování poměru – například aby budova, která je ve skutečném světě nižší než strom, nebyla ve vizualizaci vyšší.

2) Konstrukce zájmových entit ve vybrané oblasti (v současné době se jedná pouze o entity typu „budova“).

3) Vytvoření podstavce pod vykresleným terénem. Jeho vytvoření jsem zvolil z důvodu zakrytí oblasti pod vytvořeným terénem a skrytí spodních částí entit, které by pod terén mohly zasahovat. To nastává téměř u každé entity, jelikož její dolní část přímo nekopíruje terén, ale je zde zarovnána na hodnotu její minimální Z souřadnice.

4) Posledním krokem je vytvoření bodů, které se vztahují ke zvolenému režimu pohybu. Tato fáze se vykonává pouze pokud byl zvolen režim pohybu „do cílového bodu“ nebo „po předem stanovené trase“.

5 Serverová komponenta

Serverová část, která byla navržena Bc. Janem Kozánkem, je klíčovým prvkem pro moji aplikaci a také pro všechen budoucí klientský software, jelikož je zodpovědná za správné zpracování dat a jejich následné poskytnutí skrze REST API. Komunikace s ním probíhá pomocí zabezpečeného HTTPS spojení.

Server slouží mé aplikaci k provedení inicializace pohybu cílového ANS. Přímá komunikace by zde nebyla ve své podstatě možná, především z důvodu náročnosti zpracování dat pro datový model. Dále by mohl být také problém s tím, že cílové zařízení bude připojeno k internetu prostřednictvím mobilního operátora. Tudíž se předpokládá, že se bude často měnit jeho IP adresa. Moje aplikace by tedy ve své podstatě nikdy nemohla s jistotou určit, jakou má zařízení v současné době přidělenou IP adresu. Tento stav by bylo možné ošetřit implementací synchronizačních mechanismů, kdy by byly ze strany ANS odeslány informace potřebné ke komunikaci. Bohužel zde narážíme na ten samý problém – nelze s jistotou určit, jakou IP adresu bude mít moje desktopová aplikace, jelikož může být spuštěna na libovolném počítači.

Architektura klient-server všechny tyto problémy odstraňuje. Serverová část je charakteristická právě tím, že její adresa je neměnná (měnit se může, ale jen ve výjimečných případech). Díky tomu mohou být implementovány zmiňované synchronizační mechanismy a lze tedy navázat plnohodnotnou komunikaci.

5.1 Konstrukce dlaždic

Jednou z hlavních činností serveru je naplnění datového modelu představující zájmovou oblast pohybu pro ANS. Ta je z důvodu zpracování a cachování rozdělena na jednu nebo více dlaždic, které tvoří základ modelu. Každá představuje plochu o velikosti 128×128 S-JTSK souřadnic a je určena především k uchování výškopisných dat pro oblast, kterou definuje. Pro jednoznačnou identifikaci každé dlaždice se využívá souřadnice jejího levého horního rohu (X a Y v S-JTSK souřadnicovém systému).

Ke konstrukci jednotlivých dlaždic jsem napsal metodu, která využívá REST webových služeb poskytovaných ArcGIS Serverem od ČÚZK.

Tato metoda umožňuje získávat data pro zvolenou oblast, kterou lze pomocí metody dvěma způsoby. První využívá prostého zadání počtu dlaždic na výšku a šířku, čímž určíme velikost zájmové oblasti. Je také potřeba určit souřadnici levého horního rohu pro požadovanou oblast. Druhý způsob je trochu odlišný. Oblast je vytvářena v okolí vybrané dlaždice do všech směrů. Nutností je určit souřadnici definující prostřední dlaždici a počet dlaždic, které chceme v jejím okolí vybrat. Celkový počet dlaždic pak odpovídá vztahu $(2n+1)^2$, kde n je požadovaný počet dlaždic v okolí. Samotná konstrukce se skládá ze dvou kroků popsaných níže.

5.1.1 Získání výškových profilů

Data popisující výškový charakter vybrané oblasti se získávají ze služby DMR 5G. Pro práci s ní poskytuje několik metod. Pro získání výšek jsem využil metody *Get Samples*. Tato metoda dovoluje určit oblast několika způsoby: pomocí polygonu, obdélníkového výběru nebo zadáním množiny bodů, kde pro každý dostaneme vzorek s výškou terénu. Já jsem využil poslední zmiňovaný způsob. Důvodem byla jeho naprostá přesnost v porovnání se zbývajícími, které z nepochopitelných důvodů mnou vybranou oblast posouvaly a nebylo možné získat přesná data.

Nevýhodou je, že lze zadat pouze tisíc bodů v rámci jednoho dotazu, pro které chceme získat vzorek. To ve výsledku znamená, že na získání výškových dat pro jednu dlaždici je potřeba pět rozdílných dotazů na DMR 5G službu, jelikož je potřeba 4096 vzorků. Vždy je potřeba vytvořit tělo dotazu ve formátu, jaký služba vyžaduje. V něm se určuje způsob, jak je definována oblast (polygon, obdélník nebo množina bodů), jednotlivé body vztahující se ke zvolenému typu a formát dat, v jakém služba DMR 5G vrátí odpověď – JSON nebo HTML. Na příkladu si ukážeme, jak by vypadalo tělo dotazu pro získání dvou vzorků pomocí vybraného způsobu.

```
geometry={"points": [-900128,-1000000],[-900000,-1000128]}
&geometryType=esriGeometryMultipoint
&f=json
```

Zdrojový kód 1: Tělo dotazu pro získání výškových vzorků

Poté, co mám připravený řetězec se všemi body, odešlu jej pomocí POST metody na DMR 5G službu. Použití metody GET zde není možné, jelikož nelze do URL vkládat takto dlouhé řetězce (definice tisíce bodů má přes deset tisíc znaků, což je mnohem více, než maximální délka URL). Obdržená odpověď v JSON formátu je rozparsována a výšková data jsou postupně vkládána do dvourozměrného pole. Zároveň je s průchodem dat zjišťována maximální a minimální hodnota Z souřadnice. Jakmile se provede pět dotazů, je dlaždice úspěšně naplněna všemi potřebnými daty.

5.1.2 Stažení satelitního snímku

Satelitní snímek je v dlaždici obsažen především kvůli využití v klientském softwaru (například při vizualizacích). Může však sloužit také jako podklad pro rozpoznávání obrazu z kamer přímo v ANS. Pro získání satelitního snímku jsem využil služby ortofoto.

Stažení je provedeno až po samotném naplnění pole výškovými daty. Práce s touto službou je velmi podobná jako v předchozím případě. Specifikace oblasti, pro kterou chceme satelitní snímek, se zadává pomocí dvou bodů, které utvářejí obdélníkový výběr. Je nutné také zvolit formát exportovaného obrázku a jeho velikost (já jsem využil formát PNG a velikost 512x512 pixelů). Poté co je obrázek stažen, je zakódován pomocí Base64 do řetězce a uložen do příslušné dlaždice.

6 Realizace aplikace pro výběr oblasti

Implementace této části vychází z návrhu popsaného v kapitole 4.4.1. Hlavní komponentou aplikace je mapa, která pokrývá celé území ČR. Při vytvoření objektu mapy je nastaven souřadnicový systém, který bude mapa a celá aplikace využívat - tedy z výchozího systému WGS84 na S-JTSK. Důvody byly uvedeny v kapitole 4.4.2.

Jelikož jsem pro implementaci využil ArcGIS Java SDK, bylo potřeba aplikaci strukturovat do oddělených vrstev, se kterými tato knihovna pracuje. Základní vrstva slouží ke zobrazování mapy. Pro tuto vrstvu je potřeba nastavit vhodný WMS, který musí využívat stejného souřadnicového systému, jaký je nastaven u objektu mapy – tedy S-JTSK v Křovákově zobrazení.

Jsou implementovány dvě různé varianty zobrazení mapy, mezi kterými lze kdykoliv za běhu přepínat. Jedná se především o ortofoto mapu využívající zobrazení ve formě satelitních snímků (služba je dostupná na adrese http://ags.cuzk.cz/arcgis/rest/services/ortofoto/MapServer). Tato varianta neobsahuje popisky jednotlivých objektů, jako jsou ulice, názvy měst a další, což by mohlo zhoršovat orientaci v mapě. Z tohoto důvodu jsem umožnil zobrazovat topografickou verzi mapy (dostupné z adresy http://ags.cuzk.cz/arcgis/rest/services/zm/MapServer). Pro ulehčení orientace v mapě jsem implementoval také vyhledávání zvolených míst, které je popsáno později v této práci.

Kromě mapové vrstvy obsahuje aplikace také vrstvu pro zobrazování grafických objektů. Mezi ně patří například vybraná oblast pohybu, body pro režim pohybu, případně zakreslení oblastí se zakázaným pohybem cílového zařízení. Tato vrstva je využívána téměř u všech implementovaných operací.

Poslední vrstvou je tzv. kreslící vrstva, která je obsažena v použité knihovně od Esri. Slouží ke kreslení základních grafických objektů – úsečky, lomené čáry, body aj. Tyto objekty jsou po jejich nakreslení vloženy do grafické vrstvy a zobrazeny v aplikaci. Jelikož tato vrstva dokonale nesplňovala mé požadavky na zakreslování výběru oblasti, musel jsem její možnosti rozšířit o nový režim kreslení, který realizuje výběr oblasti pohybu.

6.1 Vytvoření mřížky pro zarovnání dlaždic

Každý objekt mapy má nastavený maximální rozměr. To je maximální hodnota, na kterou lze mapu oddálit. Je určen dvěma S-JTSK souřadnicemi, které tvoří obdélníkovou oblast. Tato hodnota vychází z nastavení použitého pro WMS, které zajišťuje vykreslování map (ať už satelitní nebo topografické zobrazení). Pro mnou používané podkladové služby, které využívá mapová vrstva, je tato výchozí hodnota nastavena následovně:

- levý horní roh oblasti: X = -904 703.61, Y = -934 000.66
- pravý dolní roh oblasti: X = -431 857.61, Y = -1 227 416.66

Oblast tvořená těmito body pokrývá území celé ČR. Tyto body je možné pro objekt mapy přenastavit na novou hodnotu. To se hodí k realizaci mřížkového systému, který jsem navrhl, abych zajistil zarovnávání jednotlivých dlaždic pro oblast výběru. Aby mohl být systém vytvořen, bylo potřeba nalézt nové body definující oblast. Pro ni platí, že musí obsahovat výchozí oblast a zároveň hodnoty souřadnic těchto bodů musí být bezezbytku dělitelné hodnotou 128 (velikost dlaždice). Musí tedy vyhovovat kongruenci: $0 \equiv a \bmod 128$, kde a je hodnota hledané souřadnice. Nově zvolená oblast je vyřešením vztahu definována těmito hodnotami:

- levý horní roh mřížky: X = -904 704, Y = -935 168
- pravý dolní roh mřížky: X = -431 616, Y = -1 227 392

Tyto nové hodnoty již vyhovují výše zmiňovaným požadavkům. Vypočtené body využívám k nalezení vhodné dlaždice, která bude spadat do mřížkového systému. Samotné zarovnání oblasti provádím dvěma výpočty (viz Zdrojový kód 2). Vždy se zarovnává bod, kam uživatel na mapě klikl. Nejdříve je potřeba spočítat hodnotu definující pozici v mřížce. Tuto hodnotu následně vynásobím velikostí dlaždice a přičtu (pro X) nebo odečtu (pro Y) od příslušné souřadnice bodu, který určuje levý horní roh mřížky. Nově vypočtené hodnoty souřadnic jednoznačně určují zarovnanou dlaždici.

```
//zjištění pozice nové dlaždice v mřížce
//startPoint představuje bod, kam uživatel klikl na mapě
double x = Math.round((startPoint.getX()-bounds.getXMin())/128);
double y = Math.round((bounds.getYMax()-startPoint.getY())/128);

//spočítání souřadnic nové dlaždice
double newX = bounds.getXMin()+(x*128);
double newY = bounds.getYMax()-(y*128);
```

Zdrojový kód 2: Zarovnání výběru do mřížkového systému

6.2 Výběr oblasti

Uživatel má možnost vybírat oblast pohybu dvěma způsoby. Primární metodou je výběr pomocí myši. Ten je aktivován po stisku tlačítka „Vybrat oblast“. Pokud uživatel klikne na mapu, je bod kam klikl automaticky zarovnán do výše zmiňované mřížky. Následně lze tažením myši realizovat výběr oblasti. Dalším klikem se výběr potvrdí a oblast je zanesena do mapy.

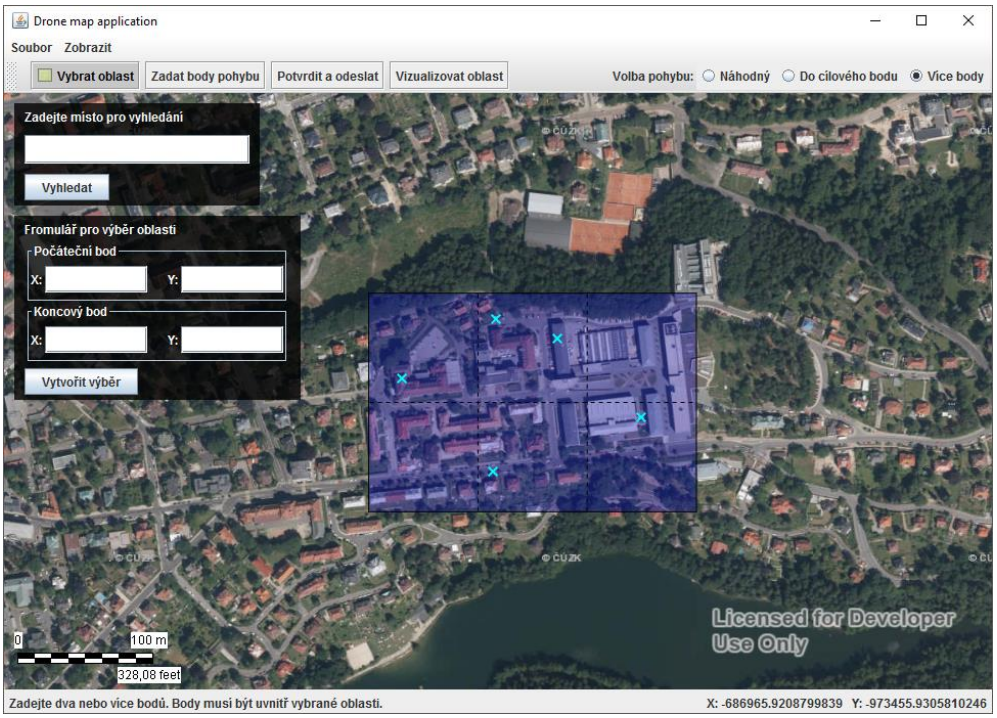
Jako druhý způsob může využít formulář v levé části GUI. Je potřeba zadat počáteční a koncový bod specifikující oblast, která má být vybrána. Zadané souřadnice každého z bodů musí být ve správném formátu platném pro S-JTSK souřadnice. Zároveň také musí být rozdíl mezi hodnotami souřadnice X větší než 64 (tedy polovina velikosti dlaždice). Stejná podmínka platí i pro souřadnice Y. Pokud není splněno výše uvedené, je vyvoláno dialogové okno s chybovou hláškou.

6.2.1 Specifikace trasy pohybu

Zadání bodů udávajících trasu pohybu je úzce spjato s úspěšným výběrem oblasti pohybu. Jestliže tento krok není proveden, nelze tyto body zadávat. Každý ze tří režimů je specifický jak pro chování, které určuje pro cílové zařízení, tak i na to, co je potřeba pro jeho správnou specifikaci.

Režim náhodného pohybu nevyžaduje žádné zadávání bodů určujících trasu. Pro režim pohybu do cílového bodu je nutné specifikovat právě jeden bod, do kterého se má cílové

zařízení přemístit. Posledním režimem je režim po více bodech. Tady je nutné zadat alespoň jeden bod, přes který se má zařízení pohybovat. Ukázku aplikace s vybranou lokací, spolu se zadanými body pro třetí ze zmiňovaných režimů, můžeme vidět na následujícím obrázku. Modrý obdélník určuje oblast pohybu rozdělenou na jednotlivé dlaždice čárkovanou čarou. Tyrkysové křížky znázorňují vybrané body, přes které má zařízení prolétat.



Obrázek 6: Ukázka výběru oblasti a specifikace trasy

6.3 Vyhledávání v mapě ČR

Pro usnadnění orientace na mapě jsem implementoval funkci vyhledávání, které využívá třídy *Locator* z používané knihovny ArcGIS Java SDK. Ta umožňuje asynchronně kontaktovat online vyhledávací službu, která vrací množinu bodů vyhovujících hledanému výrazu. Vyhledávat lze na základě názvu města, ulice, pohoří, hory a dalších. Pokud je oblast úspěšně vyhledána, je na tuto oblast mapa zaostřena. V opačném případě je vyvoláno chybové hlášení.

6.4 Zobrazení oblastí se zakázaným pohybem

Zakázané oblasti jsou jedny ze tří typů, které v této práci bereme v potaz. Jedná se o oblasti, na které je kladeno nějaké omezení. To může například zakazovat pohyb dronů v definované oblasti (vojenské objekty, jaderné elektrárny apod.). Aby byl uživatel obeznámen s těmito omezeními, implementoval jsem do aplikace metodu, která zajišťuje jejich zakreslení do mapy. Prozatím však není stanoven zdroj, ze kterého by se informace o zakázaných zónách daly čerpat. Vykreslení využívá dat uložených v JSON souboru. Je tedy nutné dodržet stanovený formát (viz Zdrojový kód 3) pro správné zakreslení oblastí do mapy. Každá oblast je definována body, které ji vymezují. Pro danou oblast lze také uvést doplňující informace o tomto omezení (pole *info*).

```
{
   "id": 1,
   "points":[[-686471,-973736],
             [-686471,-973737],
             [-686468,-973737],
             [-686471,-973736]]
   "info": "Trvalý zákaz průletu dronů."
}
```

Zdrojový kód 3: Formát JSON souboru definující zakázané oblasti

6.5 Získání dat ze serveru

Posledním krokem před vizualizací vybrané oblasti je odeslání požadavku na server. Pro moji aplikaci je k dispozici jedna metoda, která je v současné době přístupná z adresy: https://drone.fm.tul.cz:18443/DroneComponent/service/getTilesMxN. Dotaz se posílá pomocí POST metody a tělo musí být ve formátu JSON. V něm je uvedena oblast zájmu společně se zvoleným režimem pohybu, která má být předána do cílového zařízení. Po zpracování je odeslána odpověď. Pokud nastane chyba, ať z důvodu nesprávného formátu dotazu nebo ve zpracování dat na serveru, odešle

metoda chybový kód. V opačném případě získám data pro všechny entity v oblasti a dlaždice, které oblast tvoří. Na základě těchto dat pak mohu vytvořit 3D vizualizaci.

6.5.1 Formát odpovědi

Odpověď (viz Zdrojový kód 4) na zvolenou oblast reflektuje využívaný navržený datový model. Pro její následné zpracování na klientské straně využívám objektů *Tile* a *Entity*. Aby mohla být odpověď deserializována, musí se objekty přesně shodovat s tím, jak jsou implementovány na serverové straně. K samotné deserializaci využívám knihovny od Google s názvem *gson*. Ta umožňuje velmi efektivně a rychle pracovat s daty v JSON formátu. S její pomocí tak naplním pole dlaždic a pole entit, které potřebuji k vytvoření vizualizace.

```
{
  "tiles": [{
    "x": -687088,
    "y": -973388,
    "zMin": -973388,
    "zMax": -973388,
    "size": 128,
    "step": 2,
    "texture": "iVBORw0KG..." //satelitní snímek v Base64 kódování
    "dataDMR": [[...]], //pole s výškovými daty
  },
  {...}], //definice dalších dlaždic tvořících zvolenou oblast
  "entities": [{
    "id": "35778990",
    "type": "building",
    "maxZ": 677.69,
    "minZ": 670.12,
    "WGS84": [...], //body definující entitu ve WGS84 formátu
    "SJTSK": [...], //body definující entitu v S-JTSK formátu
  },
  {...}] //definice dalších entit v oblasti
}
```

Zdrojový kód 4: JSON formát odpovědi ze serveru

7 Vykreslení 3D scény

K vytvoření 3D vizualizace jsem využil knihovny Java 3D. Aplikace podporuje dva způsoby pro vykreslení zvolené oblasti. První způsob využívá dat přijatých ze serveru. Tento způsob je také určen pro inicializaci pohybu cílového zařízení. Druhá možnost, jak vizualizovat oblast, je výběr JSON souboru uloženého na disku, kde je moje aplikace spuštěna. Formát tohoto souboru musí být naprosto shodný s formátem odpovědi, kterou přijímáme ze serveru (tedy musí odpovídat stanovenému datovému modelu). Tato druhá metoda nám slouží pouze jako způsob, jak tyto data vykreslovat. Nelze pomocí ní zadávat příkaz k pohybu cílového zařízení, jelikož se zde neodehrává žádná komunikace se serverem.

Vizualizační část aplikace je vykreslena ihned poté, co ze serveru přijde odpověď s daty pro zvolenou oblast nebo po zvolení validního JSON souboru s daty popisujícími nějakou oblast. Každému vykreslení vždy předchází zpracování zadaných dat. Až poté je opravdu vytvořeno nové okno, ve kterém je vidět oblast pohybu ve 3D. Pro manipulaci se scénou je využívána myš. Scénu lze libovolně otáčet, přibližovat a oddalovat. Doplňující funkcí pro scénu je možnost skrývat entity a body definující trasu pohybu pro zvolený režim (jeden nebo více bodů). Tyto funkce jsou přístupné pomocí menu aplikace.

Ve scéně vykresluji čtyři typy objektů – objekt představující terén, podstavec pod celou oblastí zakrývající spodní část terénu, budovy spadající do vybrané oblasti pohybu a pokud byl zvolen režim, který vyžaduje zadat body pro definici trasy, jsou ve scéně vykresleny právě tyto body. Každý objekt je vytvářen jiným způsobem. Postup, jak jsou jednotlivé objekty ve scéně vykreslované, je popsán níže.

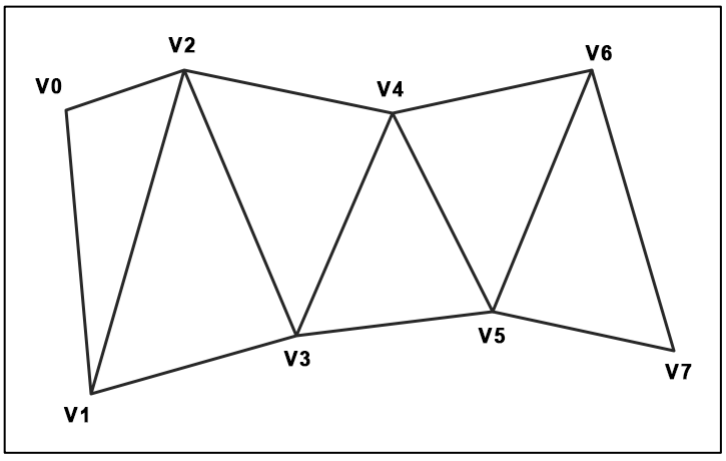
7.1 Příprava dat pro popis terénu

Terén je základním objektem mé scény. Je potřeba jej vytvořit jako první objekt, jelikož na základě vyvýšení reálné oblasti je vytvářeno měřítko pro celou scénu. Pomocí něj jsou pak všechny ostatní objekty vykreslovány a jejich výška je upravována tak,

aby měřítku vyhovovala. Docílíme tak zachování proporcí mezi vyvýšením terénu a výškami zobrazovaných budov.

K vytváření terénu jsem navrhl třídu *TerrainBuilder*, která obstarává kompletní vykreslení objektu představujícího terén spolu s podstavcem a body pro zvolený režim pohybu. Výstupem této třídy jsou v zásadě dva objekty typu *Shape3D*, které lze přidat do grafu scény, aby byly zobrazeny. Jedná se o objekt terénu a podstavce. Pro body pohybu jsem využil rozdílný přístup, jelikož je konstruuji za pomocí geometrických primitiv. Jejich poloha se musí určovat prostřednictvím *TransformGroup*, která umožní s objekty manipulovat a měnit jejich pozici ve scéně.

Terén je ve své podstatě soustava po sobě jdoucích vrcholů. Ty musí být vloženy do vhodné datové struktury. Já jsem zvolil tzv. trojúhelníkové pásy, pro jejichž konstrukci je v Javě 3D k dispozici třída *TriangleStripArray*. S její pomocí lze efektivně vytvářet objekty, které využívají tohoto přístupu. Tento pás (viz Obrázek 7) je posloupnost vrcholů, kde první tři tvoří první trojúhelník (vrcholy V0, V1 a V2). Každý další vrchol definuje další trojúhelník, který má s předchozím jednu společnou hranu. Tento přístup je poměrně často používaný pro konstrukci složitějších 3D objektů, jelikož je efektivně využita paměť počítače. Pro všechny pásy, které jsou vloženy do *TriangleStripArray*, musí platit, že jsou vždy stejně dlouhé.



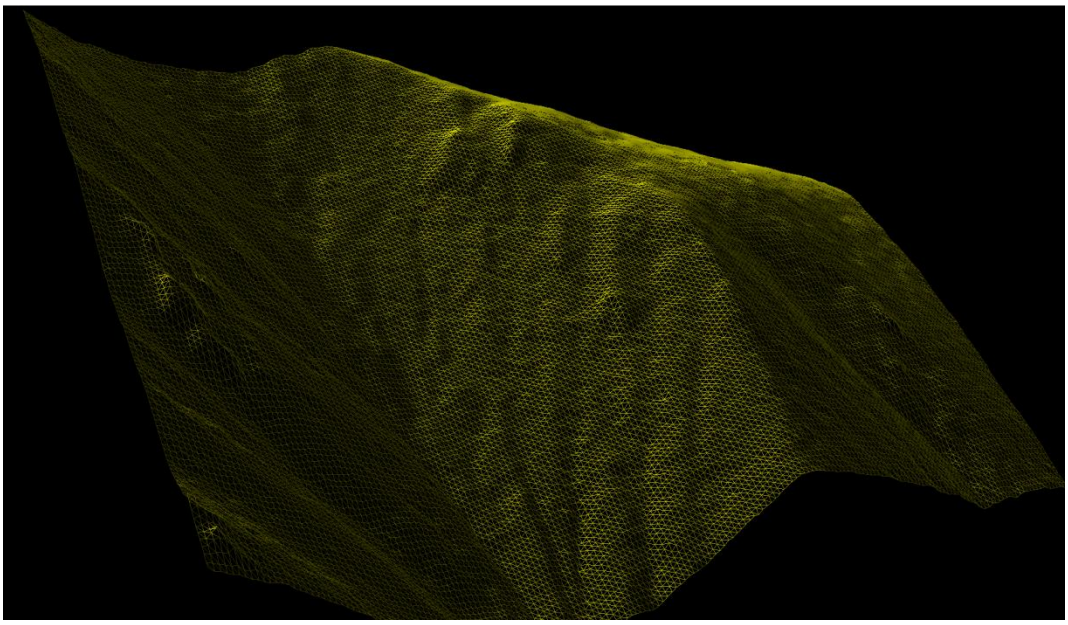
Obrázek 7: Realizace trojúhelníkového pásu

Před samotným vytvořením pásů je potřeba vzít výškopisná data ze všech dlaždic, které tvoří vybranou oblast, a sloučit je do jednoho velkého dvourozměrného pole. Tím zajistím jednodušší procházení všech výškových dat. Každá hodnota v poli pak představuje výšku reliéfu v příslušném bodě. Poté co je tento krok vykonán, můžu přejít ke konstrukci trojúhelníkových pásů. K jejich vytvoření je potřeba projít vzniklé dvourozměrné pole a postupně vytvářet vrcholy definující terén. Hodnota Z souřadnice každého vrcholu odpovídá příslušné hodnotě v poli s výškovými daty. Naprostou nutností je volba vzdálenosti mezi jednotlivými vrcholy. Zvolená hodnota se pak musí využívat pro vytváření všech ostatních objektů, jinak by nebyly na svém místě. Já jsem pro vykreslení zvolil krok 0,025, který se zdál být optimální.

7.1.1 Vykreslení drátěného modelu terénu

Když jsou data připravena, lze přejít k zobrazení základu tvořícího náš terén. Vytvořený objekt *TriangleStripArray* je nyní potřeba vložit do objektu *GeometryInfo*, aby mohly být vygenerovány normálové vektory. Ty určují úhel, který v daném bodě svírá dopadající paprsek s povrchem, a tím umožňují vytvářet plynulé přechody a dojem, že je objekt prostorový. Java 3D má pro automatické vytváření normálových vektorů implementovanou třídu *NormalGenerator*. Ta je velmi užitečná, pokud potřebujeme vytvářet vektory, na něž neklademe nějaké speciální požadavky (to je právě náš případ).

Jakmile jsou hotové normálové vektory, je vše připraveno ke zobrazení terénu ve scéně. Z objektu uloženém v *GeometryInfo* nyní vytvořím *Shape3D*, který nám už reprezentuje finální tvar terénu. Aby bylo možné jej zobrazit, je potřeba mu nastavit jak se bude jevit navenek pomocí třídy *Appearance*. Pro ukázku aplikace trojúhelníkových pásů jsem zde zvolil vykreslení drátěného modelu. Ten ve své podstatě tvoří úplný základ tohoto objektu. Na obrázku níže můžeme vidět vytvořenou vizualizaci. Jednotlivé trojúhelníky utváří dojem vyvýšené krajiny a to jen díky tomu, že každý vrchol, který terén tvoří, je vytvářen s odlišnou výškou, která odpovídá příslušným výškovým datům jednotlivých dlaždic.



Obrázek 8: Vymodelování terénu ve formě drátěného modelu

7.1.2 Aplikování textury na terén

Jelikož máme k dispozici satelitní snímky ze služby ortofoto, lze je využít jako texturu pro vzniklý objekt. Jediné co potřebujeme udělat, je projít všechny dlaždice, vzít satelitní fotografie z každé z nich a sloučit je do jedné fotografie. Tu pak budeme moci využít jako texturu pro náš terén.

Aby byla textura „umístěna“ správně na náš objekt, je potřeba provést mapování. Zde se setkáváme s pojmem texel (z anglického „*texture pixel*“). Stejně jako je u obyčejného obrázku základní jednotkou pixel, u textur se používá jako základní jednotka texel. Textura je pak tvořena polem texelů. Proces mapování textur spočívá v tom, že musíme přiřadit jednotlivé texely na jednotlivé vrcholy mapovaného objektu (v našem případě terénu).

Java 3D disponuje generátorem (třída *TexCoordGeneration*), který udělá mapování textury na objekt za nás. Tomu je potřeba zadat šířku a výšku objektu, který má být otexturován. Jakmile je generátor inicializován, musí být přiřazen do objektu *Appearance* našeho terénu. Díky tomu zajistíme správné namapování textury, kterou

chceme na objekt použít. Tento generátor dosahuje výborných výsledků. O tom se lze přesvědčit na obrázku níže, kde je dříve vytvořený objekt terénu s aplikovanou texturou.

Obrázek 9: Terén s aplikovanou texturou

7.2 Vytvoření a vykreslení budov

Jedním z hlavních úkolů aplikace je vizualizace entit ve zvolené oblasti (pokud se zde nějaké entity nacházejí). V současné době je podporováno vykreslování entit typu budova. Vytvoření každé budovy jsem rozdělil do dvou kroků. Nejdříve je vytvořena střecha, která je zarovnána dle maximální Z souřadnice dané entity. Poté jsou na střechu navázány jednotlivé stěny budovy. Ty jsou ve spodní části zarovnány naopak na minimální Z souřadnici pro vytvářenou budovu.

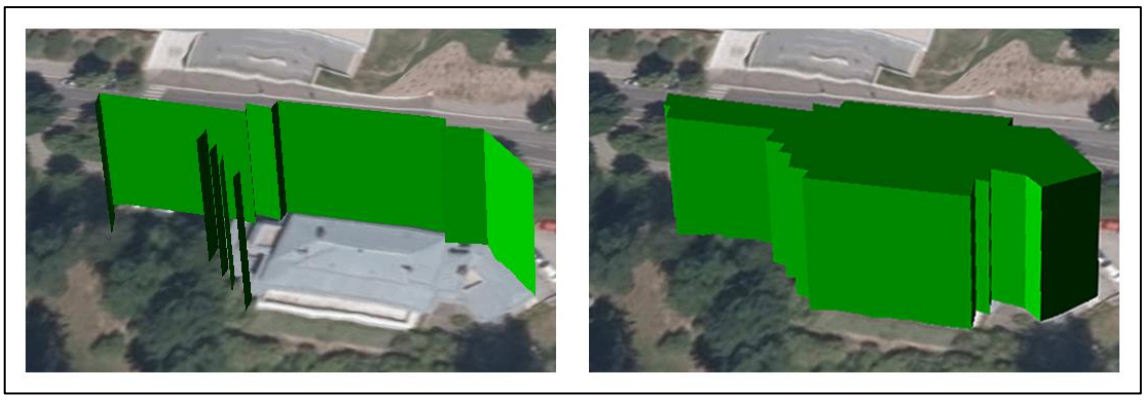
Nejdříve je potřeba opět předzpracovat data. Vytváření budovy tedy začínám tím, že si naplním pole body, které tvar definují. Využívám souřadnice bodů z OSM služby, které přepočítávám vůči dříve vymodelovanému terénu, aby byly budovy na místě, kde mají být (zjednodušeně řečeno musí jejich poloha souhlasit s polohou na textuře terénu). Při vykreslování jsem se setkal s jedním problémem. Tím byl způsob, jakým jsou do

OSM body zadávány. Popisu a následné korekci tohoto problému se věnuji v kapitole 7.2.1.

Pro popis budovy a následné vytvoření normálových vektorů pak opět používám *GeometryInfo* a *NormalGenerator*. Střecha je tvořena polygonem, jehož vrcholy kopírují vrcholy z OSM služby. Každou stěnu pak tvoří obdélník.

7.2.1 Korekce vrcholů z OSM služby

Body, které uživatelé do OSM databáze vkládají, jsou občas zadány po směru hodinových ručiček a občas proti směru hodinových ručiček. Je nutné se rozhodnou, který systém bude aplikace využívat. Pokud se vrcholy objektů vykreslují v rozdílném pořadí, vede to k nesprávnému vykreslování objektů ve scéně, konkrétně se jedná o nesprávnou reakci na osvětlení. Toto chování můžeme vidět na obrázku níže. Vlevo je budova, u které není provedena korekce. Vpravo pak stejná budova po korekci pořadí vrcholů.



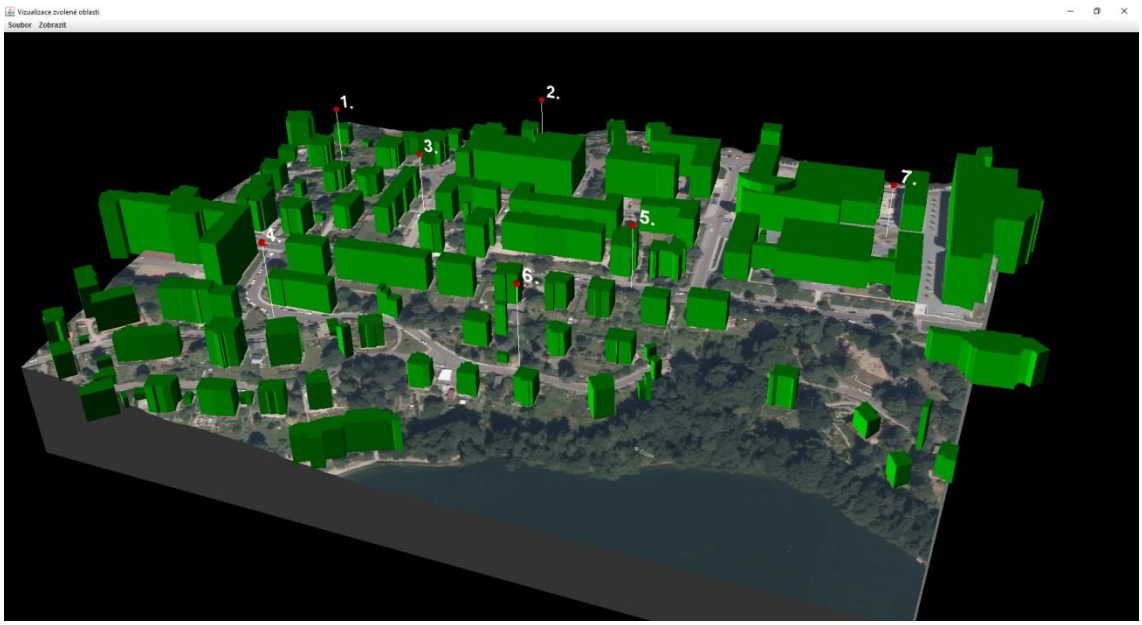
Obrázek 10: Ukázka dopadu uspořádání vrcholů na vykreslení entit

Pro vykreslení budov jsem se držel uspořádání vrcholů ve směru hodinových ručiček. Abych zajistil toto pořadí, je potřeba projít vrcholy definující každou entitu. Při průchodu používám následující vzorec: $(x_2 - x_1) * (y_2 - y_1)$, kde $x_{1,2}$ a $y_{1,2}$ jsou souřadnice dvou po sobě jdoucích vrcholů. Postupně vytvářím sumu všech výsledků. Po projití všemi vrcholy se na základě hodnoty sumy rozhoduji, jaké je uspořádání vrcholů.

Jestliže je suma kladná, jsou vrcholy ve směru hodinových ručiček a vše je v pořádku. Pokud je suma záporná, jsou vrcholy v proti směru hodinových ručiček. V tomto případě je potřeba převrátit pořadí všech vrcholů v poli.

7.3 Vizualizace bodů definujících trasu pohybu

Zakreslení bodů definujících trasu pohybu je posledním krokem k úplné vizualizaci vybrané oblasti. Pro každou značku jsem zvolil jednoduchý vzhled objektu skládající se ze dvou geometrických primitiv – koule a úsečka. Ke každému bodu navíc také vykresluji text, který určuje pořadí příslušného bodu tak, jak byl zanesen do mapy. Tento způsob mi přišel jako rozumné řešení, jelikož jsou body ve scéně jasně rozpoznatelné. Následující obrázek ukazuje, jak jsou body ve scéně zobrazeny. Zároveň také ukazuje finální vzhled celé scény, která obsahuje všechny implementované objekty.



Obrázek 11: Vymodelovaná scéna se všemi implementovanými typy objektů

8 Závěr

Cílem práce bylo navrhnout způsob na získávání mapových a datových podkladů z projektu OpenStreetMap a služby ArcGIS Server poskytované Českým úřadem zeměměřickým a katastrálním. Tyto služby byly zvoleny především kvůli jejich dostupnosti a bezplatnému použití. Získaná data jsou určena pro zařízení typu dron nebo pozemní vozidlo, která budou tyto informace využívat k rozpoznávání svého okolí a vyhodnocování možností pohybu tak, aby se zamezilo jakýmkoliv kolizím.

Podle typu zařízení byly stanoveny požadavky na data, na jejichž základě byl navržen datový model pro popis vybrané oblasti na území České republiky. Každá oblast je rozdělena na dlaždice o konstantní velikosti (128×128 S-JTSK souřadnic), které obsahují výšková data pro popis reliéfu krajiny pro oblast jimi určenou. Součástí datového modelu je také pole entit (budovy, silnice, cesty), které se ve zvolené oblasti vyskytují. Každá entita je definována sadou bodů, které určují její polohu, a obsahuje informace o její maximální výšce. Veškeré získávání dat a jejich zpracování je z důvodů náročnosti přesunuto na stranu serveru, který je prostřednictvím REST API poskytuje klientskému softwaru.

Za účelem výběru zájmové oblasti, ve které se má cílové zařízení pohybovat, jsem navrhl a implementoval aplikaci v jazyce Java, která využívá knihovny ArcGIS Java SDK 10.2.4. Tato knihovna poskytuje řadu funkcí pro práci s mapami. Aplikace umožňuje uživateli zadávat oblast pohybu kdekoliv na území ČR. Dále je použita pro určení způsobu, jakým se má zařízení pohybovat a případné zadání bodů, podle toho, jaký režim je zvolen. Pro zefektivnění ukládání dat do serverové databáze je výběr oblasti zarovnáván tak, aby spadal do vytvořeného mřížkového systému. Tím se zajistí urychlení zpracování dat na serveru, jelikož se všechny dlaždice ukládají do databáze pro jejich pozdější znovupoužití a odpadá tak potřeba data znovu zpracovávat. Výstupem ze serveru je datová struktura v JSON formátu, která reflektuje navrhovaný datový model.

Jako poslední jsem implementoval aplikaci, která převádí získaná data do 3D podoby. Pro modelování jsem využil knihovnu Java 3D, která pro svůj běh využívá OpenGL API. Důvodem k použití Java 3D byl především její plně objektový přístup, díky

kterému je možné aplikace jednodušeji rozšiřovat a udržovat. Výstupem aplikace je 3D scéna, která plně koresponduje se vzhledem vybrané oblasti v reálném světe. Ve scéně jsou implementovány následující objekty – terén s vytvořenou texturou ze získaných satelitních fotografií, budovy spadající do vybrané oblasti a body, které určují trasu pro vybraný režim pohybu.

Všechny stanovené cíle byly splněny a výsledná aplikace je plně funkční. Jelikož aplikace v současné době umožňuje vykreslovat pouze jeden typ entit (budovy), bylo by do budoucna vhodné rozšířit implementaci o vykreslení dalších entit, jako jsou například stromy, ploty apod. Tyto entity spadají mezi ty, které by mohly představovat riziko kolize pro zařízení. Další prostor pro zlepšení vidím v samotné implementaci cílového zařízení, které provádí řadu výpočtů pro zjištění optimální trasy pohybu. Tyto výpočty samozřejmě zařízení zatěžují a zvyšují výslednou spotřebu energie. Proto by bylo dobré uvažovat nad přesunem náročných výpočtů do desktopové aplikace a do zařízení odesílat již vypočtenou trasu jako součást datového modelu. Zařízení by pak mohlo využít vypočtenou trasu a případně pouze provádět drobnou korekci pohybu na základě dat ze svých senzorů. Toto nové chování by však bylo možné implementovat pouze u režimů, které při výběru oblasti vyžadují specifikovat body definující trasu.

Seznam použité literatury

[1] Budoucnost bezpilotních prostředků. *Ministerstvo obrany* [online]. 2017 [cit. 2017-03-14]. Dostupné z: http://www.army.cz/scripts/detail.php?id=1425

[2] Report: Major growth in global drone sales; delivery drones to remain niche market. *CargoForwarder Global* [online]. 2017 [cit. 2017-03-14]. Dostupné z: https://www.cargoforwarder.eu/2017/02/22/report-major-growth-in-global-drone-sales-delivery-drones-to-remain-niche-market/

[3] BEKEY, George A. *Autonomous Robots: From Biological Inspiration to Implementation and Control.* London: The MIT Press, 2006. ISBN 9780262292474.

[4] KUETHER, Derek J., Benjamin MORRELL, Gregory CHAMITOFF, Michael BISHOP, Daniele MORTARI, Peter GIBBENS a Mauricio D. COEN. *Cohesive Autonomous Navigation System*. In: AIAA Guidance, Navigation, and Control Conference [online]. Reston, Virginia: American Institute of Aeronautics and Astronautics. 2016 [cit. 2017-04-27]. DOI: 10.2514/6.2016-0640. ISBN 978-1-62410-389-6. Dostupné z: http://arc.aiaa.org/doi/10.2514/6.2016-0640

[5] KOPETSCHKE, Igor, Petr KRETSCHMER a Jan NOVÁK. *Data Preprocessing for Autonomous Navigation Systems*. International Conference on Military Technologies, Brno, 2017. ISBN 978-1-5386-1988-9.

[6] Web map service. *Open geospatial consortium* [online]. 2017 [cit. 2017-02-11]. Dostupné z: http://www.opengeospatial.org/standards/wms

[7] OpenGIS Web Map Server Implementation Specification. *Open geospatial consortium* [online]. 2006 [cit. 2017-02-11]. Dostupné z: http://portal.opengeospatial.org/files/?artifact_id=14416

[8] WMS služby. *Česká geologická služba* [online]. 2017 [cit. 2017-02-12]. Dostupné z: http://www.geology.cz/extranet/mapy/mapy-online/wms

[9] Mapové služby. *Ředitelství silnic a dálnic pro ČR* [online]. 2017 [cit. 2017-02-12]. Dostupné z: https://geoportal.rsd.cz/web/Applications/LiveData

[10] Služby Esri ArcGIS Server. *Český úřad zeměměřický a katastrální* [online]. 2017 [cit. 2017-02-12]. Dostupné z: http://geoportal.cuzk.cz/Default.aspx?mode=TextMeta&side=wms.AGS&text=WMS.AGS&head_tab=sekce-03-gp&menu=314

[11] GELETIČ, Jan. *Úvod do ArcGIS 10*. Olomouc: Univerzita Palackého v Olomouci, 2013, ISBN 978-80-244-3390-5.

[12] What is GIS?. *Esri* [online]. 2017 [cit. 2017-02-15]. Dostupné z: http://www.esri.com/what-is-gis

[13] The ArcGIS REST API. *Esri* [online]. 2017 [cit. 2017-02-14]. Dostupné z: http://resources.arcgis.com/en/help/arcgis-rest-api/

[14] Products. *Esri* [online]. 2017 [cit. 2017-02-15]. Dostupné z: http://www.esri.com/products#alpha-list

[15] Image Service. *ArcGIS REST API* [online]. 2016 [cit. 2017-02-16]. Dostupné z: http://ags.cuzk.cz/arcgis/sdk/rest/index.html/02ss00000021000000

[16] Digitální model povrchu České republiky 1. generace (DMP 1G). *Český úřad zeměměřický a katastrální* [online]. 2017 [cit. 2017-02-16]. Dostupné z: http://geoportal.cuzk.cz/Default.aspx?lng=CZ&mode=TextMeta&side=vyskopis&metadataID=CZ-CUZK-DMP1G-V&mapid=8&menu=303

[17] Digitální model reliéfu České republiky 5. generace (DMR 5G). *Český úřad zeměměřický a katastrální* [online]. 2017 [cit. 2017-02-16]. Dostupné z: http://geoportal.cuzk.cz/Default.aspx?lng=CZ&mode=TextMeta&side=vyskopis&metadataID=CZ-CUZK-DMR5G-V&mapid=8&menu=302

[18] Prohlížecí služba Esri ArcGIS Server – Orotofoto ČR. *Český úřad zeměměřický a katastrální* [online]. 2017 [cit. 2017-02-19]. Dostupné z: http://geoportal.cuzk.cz/Default.aspx?menu=3141&mode=TextMeta&side=wms.AGS&metadataID=CZ-CUZK-AGS-ORTOFOTO&metadataXSL=metadata.sluzba

[19] Map Features. *OpenStreetMap Wiki* [online]. 2017 [cit. 2017-02-27]. Dostupné z: http://wiki.openstreetmap.org/wiki/Map_Features

[20] MA, Ding, Mats SANDBERG a Bin JIANG. *Characterizing the Heterogenity of the OpenStreetMap Data and Community*. 2015. ISSN 2220-9964.

[21] API v0.6 – OpenStreetMap Wiki. *OpenStreetMap Wiki* [online]. 2012 [cit. 2017-02-27]. Dostupné z: http://wiki.openstreetmap.org/wiki/API_v0.6

[22] SELMAN, Daniel. *Java 3D programming*. Greenwich: Manning, c2002. ISBN 978-1930110359.

[23] KO, Chi Chung a Chang Dong CHENG. *Interactive web-based virtual reality with Java 3D*. Hershey, PA: Information Science Reference, c2009. ISBN 1599047918.

[24] PALMER, Ian. *Essential Java 3D fast: developing 3D graphics applications in Java*. New York: Springer, 2001. Essential series (Springer-Verlag). ISBN 9781852333942.

[25] DAVISON, Andrew. *Killer game programming in Java*. Sebastopol, CA: O'Reilly Media, c2005. ISBN 978-0-596-00730-0.

[26] KOZÁNEK, Jan. *Serverová komponenta pro aktualizaci modelu autonomního navigačního systému*. Liberec, 2017. Diplomová práce. Technická univerzita v Liberci. Fakulta mechatroniky, informatiky a mezioborových studií. Vedoucí práce Igor Kopetschke.

A Obsah přiloženého CD

- text diplomové práce
  - diplomova_prace_2017_Ondrej_Kubicek.pdf
  - diplomova_prace_2017_Ondrej_Kubicek.docx
  - kopie_zadani_diplomove_prace_2017_Ondrej_Kubicek.pdf
- zdrojové kódy – implementace desktopové aplikace, knihovny potřebné ke spuštění nejsou přiloženy
- návod.txt