VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ APLIKACE ZOBRAZUJÍCÍ MODEL TERÉNU V REÁLNÉM ČASE

MOBILE APPLICATION RENDERING DIGITAL ELEVATION MODELS IN REAL TIME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **RADIM HEJL**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. MARTIN ČADÍK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

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

Zadání bakalářské práce

Student: **Hejl Radim**
Program: Informační technologie
Název: **Mobilní aplikace zobrazující model terénu v reálném čase**
Mobile Application Rendering Digital Elevation Models in Real Time
Kategorie: Počítačová grafika

Zadání:

1. Seznamte se s problematikou mobilních aplikací pro zobrazení digitálních modelů terénu, se zpracováním obrazu a grafikou na mobilních zařízeních.
2. Proveďte rešerši existujících poskytovatelů digitálních modelů terénu a metadat a vhodný zdroj vyberte.
3. Navrhněte a implementujte systém pro zobrazení digitálního modelu terénu v reálném čase s využitím kamery zařízení a digitálního kompasu, příp. dalších senzorů.
4. S mobilní aplikací experimentujte, posuďte její vlastnosti uživatelskou studií, příp. dalšími syntetickými experimenty. Diskutujte možnosti budoucího vývoje.
5. Dosažené výsledky prezentujte formou videa a plakátu, příp. článku.

Literatura:

- Dle pokynů vedoucího.
- http://cadik.posvete.cz/locate/

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

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

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

Abstrakt

Tato bakalářská práce se zabývá vykreslováním terénu na mobilním zařízení v reálném čase. Zaměřuje se na aktuální vývoj aplikací v operačním systému Android, zejména vykreslování grafiky pomocí OpenGL ES. Na základě těchto informací a aktuálního stavu již existujících řešení je navrhnuta, implementována a testována výsledná aplikace, která vykresluje okolní terén a za pomoci senzorů zařízení v něm naviguje.

Abstract

This bachelor thesis deals with terrain rendering on mobile device in realtime. It focuses on current developement of applications in operating system Android, especially rendering of graphics using OpenGL ES. Based on this information and actual state of existing solutions, there is designed, implemented and tested resulting application, that renders surrounding terrain and navigating in it using sensors of device.

Klíčová slova

Android, OpenGL ES, mobilní aplikace, Java, Kotlin, vykreslování terénu, 3D grafika, OSG Earth

Keywords

Android, OpenGL ES, mobile application, Java, Kotlin, terrain rendering, 3D graphics, OSG Earth

Citace

HEJL, Radim. *Mobilní aplikace zobrazující model terénu v reálném čase*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Martin Čadík, Ph.D.

Mobilní aplikace zobrazující model terénu v reálném čase

Prohlášení

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

.....
Radim Hejl
30. července 2020

Poděkování

Chtěl bych poděkovat panu Ing. Martinovi Čadíkovi, Ph.D. za odbornou pomoc a cenné připomínky při psaní této práce.

Obsah

1 Úvod **2**

2 Digitální modely terénu **3**
2.1 Digitální model terénu 3
2.2 Zdroje dat 5
2.3 Existující aplikace 7

3 Vývoj aplikací pro operační systém Android **10**
3.1 Operační systém Android 10
3.2 OpenGL ES 14
3.3 OSG Earth 15

4 Návrh aplikace **17**
4.1 Případy užití 17
4.2 Uživatelské rozhraní 17
4.3 Funkce aplikace 17
4.4 Nativní knihovna 20

5 Implementace **23**
5.1 Instalace OSG Earth 23
5.2 Implementace aplikace 23
5.3 Implementace v nativní knihovně 27

6 Testování a výsledky **29**
6.1 Informace o aplikaci 29
6.2 Testování hardwarové náročnosti 29
6.3 Uživatelské testování 30
6.4 Výsledná aplikace 31

7 Závěr **36**

Literatura **37**

Kapitola 1

Úvod

Chytré mobilní telefony v poslední době nabývají na popularitě. V roce 2019 je používalo 70 % Čechů ve věku 16 a více let, což znamenalo meziroční nárůst o necelých 7 %. Zásluhu na tom má zejména cenová dostupnost mobilních telefonů nižší třídy, které už jsou ovšem svým výkonem schopny provádět většinu běžných uživatelských operací [11].

S rychlým vzestupem výkonu mobilních zařízení přišla možnost vyvíjet množství pokročilých aplikací. Jedním odvětvím z nich je vykreslování 3D grafiky.

Tato bakalářská práce se zabývá vývojem aplikace pro operační systém Android, která vykresluje okolní terén s využitím GPS a za pomocí pohybových senzorů zařízení v něm naviguje.

Přestože se na trhu objevují podobné aplikace, většina z nich není opensource a není tedy možnost je upravovat nebo využívat při vývoji jiných aplikací. Jelikož je tato aplikace vyvíjena s pomocí volně dostupné knihovny OpenSceneGraph Earth, jsou tyto problémy vyřešeny a může být využito budoucích aktualizací knihovny pro rozšiřování a optimalizaci funkčnosti aplikace.

Druhá kapitola této práce obsahuje informace o způsobech reprezentace a vizualizace modelů terénu a způsobech získávání a ukládání výškových map používaných pro tyto účely. Dále tato kapitola obsahuje výčet již existujících aplikací řešících podobnou problematiku. Ve třetí kapitole se nachází základní popis vývoje aplikací na systému Android, včetně API pro vykreslování grafiky OpenGL ES a také popis nativní knihovny OSG Earth a knihoven, které využívá.

Čtvrtá kapitola obsahuje návrh aplikace, tedy případů užití, uživatelského prostředí a popis funkčnosti jednotlivých tříd v systému Android i v nativní knihovně. V páté kapitole jsou popsány implementační detaily aplikace. Šestá kapitola obsahuje informace o vyvíjené aplikaci, její testování a vyhodnocení výsledků. V závěrečné kapitole je popsán souhrn dosažených výsledků a návrh na potenciální rozšíření.

Kapitola 2

Digitální modely terénu

Lidstvo se během celé své historie pokoušelo zaznamenat okolní terén. Už ve starověku byly využívány mapy znázorňující výškový model terénu pomocí tvarů a barev. Zlom nastal přibližně v 18. století, kdy se k reprezentaci výšky začaly používat vrstevnice. Další pokrok v zobrazení přišel zároveň s vyvinutím fotografie. Zkombinováním páru překrývajících se leteckých snímků bylo možné zjistit informace o výšce jednotlivých bodů [19].

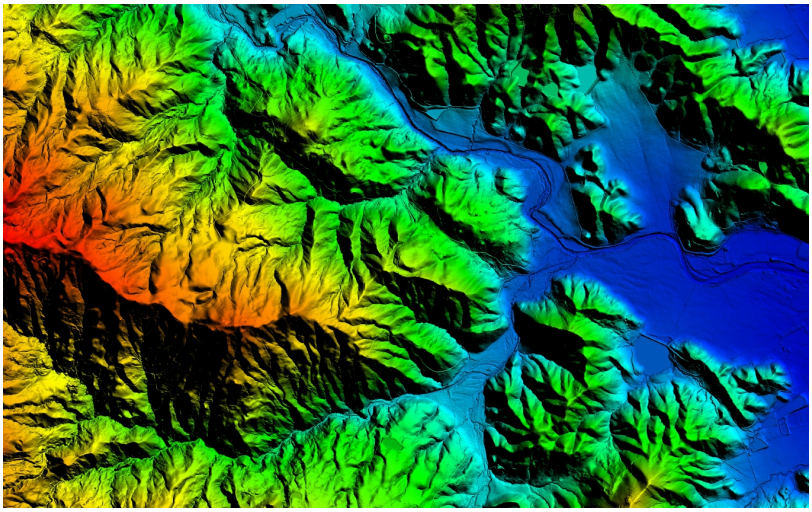
Počítačové zpracování modelu terénu se řeší už od 50. let 20. století. Kromě hodnot výšky používá i další způsoby popisující topografii povrchu jako sklon a kostru. Zpočátku se výzkum zabýval zejména přesností modelu. Byly srovnávány malé vymezené oblasti a hledaly se v nich rozdíly s již existujícími zdroji. Většina dat byla stále získávána pomocí leteckých fotografií [20].

Revoluci ve vytváření modelů terénu přineslo získávání výškových dat pomocí senzorů na vesmírných družicích. Na obrázku 2.1 je ukázka vizualizace digitálního modelu terénu.

2.1 Digitální model terénu

Digitální model terénu [18] je trojrozměrná reprezentace povrchu terénu. Vzhledem ke způsobu získávání dat nejsou modely reprezentující velké oblasti (kontinent, stát) schopny zobrazit terén ukrytý pod nejvyšší vrstvou terénu (např. jeskyně). Využívá se zejména pro modelování částí planet, popř. měsíců a asteroidů. Dělí se na:

- Model povrchu, který zobrazuje vrchní plochy všech objektů, tedy mimo terén i střechy budov, stromy atd. Využívá se v aplikacích pro vizualizaci a modelování krajiny a měst.
- Model reliéfu, zobrazující pouze reliéf zemského povrchu. Používá se k modelování záplav a odtoků, využívání půdy a planetárnímu výzkumu.
- Výškový model, který pracuje s nadmořskými výškami bodů, bez jiných informací o terénu.



Obrázek 2.1: Ukázka vizualizace digitálního modelu terénu [15].

Reprezentace DMT

Vstupní data modelu ve formátu vrstevnice nebo výškové mapy netvoří kompletní reprezentaci povrchu (hluchá místa ve výškových mapách, interval vrstevnic). Pro vytvoření kompletních dat pro tvorbu modelu je zapotřebí provést interpolaci vstupních dat. Pro tyto účely se používají metody rasterizace a TIN (Triangulated irregular network) [18].

Rastr

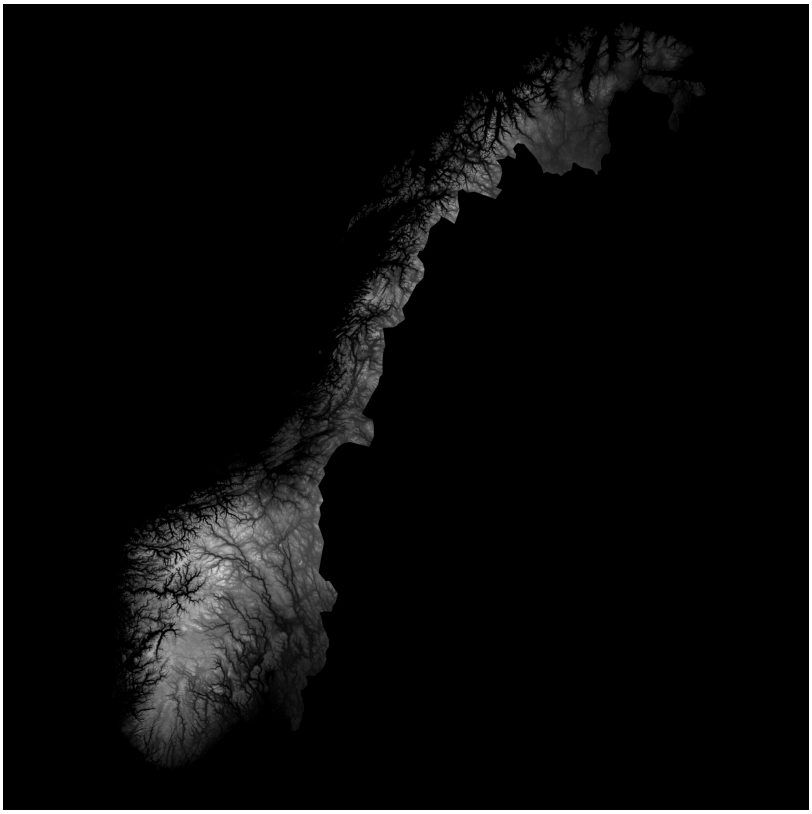
Digitální model terénu reprezentován rastrovým obrázkem, který obsahuje mřížku pravidelně distribuovaných čtverců různých odstínů šedé barvy. Odstín označuje výšku v prostoru, kdy černá značí nejmenší a bílá největší hodnotu, které může nabývat. Tato reprezentace se nazývá výšková mapa. Standardní 8-bitový RGB obrázek může nabývat pouze 256 odstínů šedi, což může způsobovat nepřesné vyobrazení zejména plochého terénu, kde vzniká schodovitá nerovnost. Z tohoto důvodu se pro větší přesnost vyobrazení mohou použít i odstíny jiných barev, čímž se zvýší počet hodnot, tedy výšek, kterých může mapa nabývat. Ukázka výškové mapy znázorněné rastrovým obrázkem je na obrázku 2.2 [18].

TIN

Digitální model terénu tvořený pomocí sítě nepravidelných trojúhelníků. Tato síť je vytvořena na základě Delaunayovy triangulace. Výhodou oproti používání rastru v mapování a analýze je, že body TIN jsou rozloženy variabilně na základě algoritmu, který určuje, které z nich jsou nejpotřebnější k vytvoření přesné reprezentace terénu. Vstup dat je proto flexibilní a není tedy potřeba takové množství bodů, jako v rastrovém DMT s pravidelně distribuovanými body. TIN ale není vhodný pro analýzu sklonu a úhlu povrchu, protože tyto vlastnosti mohou být nepřesné [18].

WGS84

WGS84 (World Geodetic System 1984) je světově uznávaný standard pro použití v kartografii, geodézii a satelitní navigaci (včetně systému GPS). Byl vytvořen ministerstvem obrany



Obrázek 2.2: Ukázka výškové mapy Norska reprezentované rastrovým obrázkem [16].

USA v roce 1984 a mimo něj je využíván také Mezinárodní organizací pro civilní letectví nebo NATO. Jedná se o trojrozměrný systém souřadnic pro určování zeměpisné šířky, délky a nadmořské výšky. V současné době se jedná o nejlepší systém pro mapování a navigaci pro planetu Zemi. Protože je Země kulatá a nebylo by na ní určování pozice jednoduché, používá tento systém její projekci na rovinu. Souřadnice WGS84 vycházejí ze zeměpisných souřadnic, šířka tedy nabývá hodnot 0° až 90° na sever od rovníku a 0° až -90° na jih od rovníku a délka hodnot 0° až 180° na východ od výchozího poledníku a 0° až -180° na západ od výchozího poledníku. Výchozím poledníkem je nultý poledník nacházející se 102m východně od Královské observatoře v Greenwichi. Jedná se tedy o kartézskou soustavu souřadnic s nulovým bodem v místě, kde se rovník protíná s nultým poledníkem [4].

2.2 Zdroje dat

SRTM

SRTM [17] (The Shuttle Radar Topography Mission) je mezinárodní výzkumný program zastřešený americkou kosmickou agenturou NASA. Jeho cílem je vytvoření digitálního výškového modelu v nejvyšším rozlišení.

Program byl spuštěn v roce 2000 a během sedmi let získal informace o asi 80 % zemského povrchu, konkrétně mezi rovnoběžkami 60° s.š. a 56° j.š. Samotné pořizování fotografií bylo prováděno 11 dní, během kterých byla nasnímána požadovaná oblast (90 % území bylo skenováno dvakrát a 50 % třikrát). Data byla vydána ve dvou verzích. První obsahovala

přímý výsledek získaný z radarových senzorů raketoplánu Endeavour, druhá zahrnuje řadu následných zpracovatelských operací zaměřených na editaci a opravu problematických oblastí (vodní plochy, pobřežní oblasti, lokální diskontinuity, horská území s velkými sklony, ...).

Pro území USA byly data vzorkována na čtvercovou mřížku o délce strany čtverce 1 úhlová vteřina (asi 30 metrů na rovníku). Pro zbytek území byla data vzorkována s délkou strany čtverce 3 úhlové vteřiny. Všechna tato data jsou přístupná pro veřejnost ve formátu .hgt. Data jsou rozdělena do mřížek, kde každá znázorňuje plochu jednoho stupně šířky a jednoho stupně délky. Jedna mřížka obsahuje 3601×3601 hodnot pro USA nebo 1201×1201 pro zbytek světa. Mřížky jsou nazývány podle pozice jejich jihozápadního rohu (např. N50E012 obsahuje data mezi 50° s.š. 12° z.d. a 51° s.š. 13° z.d.).

ASTER GDEM

Ve spolupráci japonského Ministerstva ekonomie, obchodu a průmyslu (METI) a americké kosmické agentury (NASA) vznikl globální výškový model GDEM [21] (Global Digital Elevation Model). Pro vytvoření modelu byla získávána data z japonského kosmického senzoru ASTER (The Advanced Spaceborne Thermal Emission and Reflection Radiometer), který je součástí satelitu Terra. Ten byl vypuštěn do vesmíru americkou kosmickou agenturou NASA v roce 1999. ASTER je schopný pořizovat snímky Země v 14-ti spektrálních pásmech v rozlišení až 15 metrů. Používá se pro vytváření map zobrazujících teplotu, emisivitu, odrazivost a nadmořskou výšku povrchu Země. Všechna data pořízená senzorem ASTER byla bezplatně zpřístupněna veřejnosti v roce 2016.

Globální výškový model senzoru ASTER byl vydán ve třech verzích. První verze obsahuje výšková data zemského povrchu mezi rovnoběžkami 83° s.š. a 83° j.š. Pokrývá 99 % zemského povrchu s přesností 1 úhlové vteřiny (oproti SRTM na celém území). Data byla vytvořena automatickým zpracováním 1.5 milionu obrázků pořízených během jednoho roku. Stejně jako SRTM DEM (Digital Elevation Model) pro USA obsahuje data s velikostí mřížky 3601×3601 hodnot. Jako první výškový model obsahuje podrobné informace z oblasti zemských pólů. Druhá a třetí verze jsou považovány za vylepšení první verze. Obsahují opravy anomálií v mapách první verze a rozeznání vodních ploch už od rozlohy 1km^2.

Formát .hgt

Formát .hgt [12] se používá k uchovávání dat získaných z radarů pro měření nadmořských výšek povrchu Země. Data jsou uložena v posloupnosti šestnáctibitových integerů a jejich hodnota reprezentuje nadmořskou výšku v metrech. Pro výškové mapy s přesností 3 úhlových vteřin má tedy mapa 1442401 integerů reprezentujících 1201×1201 mřížku a pro mapy s přesností 1 úhlové vteřiny 12967201 integerů reprezentujících 3601×3601 mřížku. V případě, že jsou některá pole z důvodu nepřesnosti měření neznámá, obsahují hodnotu -32768. Posloupnost je řazena v row major pořadí, tedy po řádcích od západu k východu a po sloupcích od severu k jihu. Krajní řádky a sloupce se překrývají s přilehlými výškovými mapami. Endianita (pořadí bajtů) je vzhledem ke stáří pořízených dat ve formátu big-endian, nejvíce významný bit je tedy v paměti uložen na nejnižší adrese. Jelikož většina moderních zařízení používá formát little-endian, je zapotřebí tato data převést.

Formát .tif

Formát .tif (Tagged Image File) se používá k ukládání rastrové grafiky. Byl navržen pro účely skenování a stala se z něj alternativa k formátu .jpeg, na rozdíl od něhož umožňuje bezeztrátovou kompresi. Je to tzv. kontejnerový formát, tedy formát, který dokáže nést různá obrazová data komprimovaná různými kodeky. Je to flexibilní formát, který dokáže kombinovat různé obrázky uvnitř jednoho souboru, dokáže používat různé barevné hloubky nebo různé způsoby komprese, všechny tyto informace, stejně tak jako obsah souboru, organizaci dat a jejich velikost, jsou popsány pomocí tagů v hlavičce souboru.

Formát GeoTIFF

Formát GeoTIFF umožňuje zahrnout informace obsahující reálné zeměpisné souřadnice do jednotlivých pixelů TIFF souborů. Mimo to v něm může být zahrnutý i způsob projekce mapy. Výhodou formátu GeoTIFF ve srovnáním s formátem .hgt je jeho komprese a tedy menší velikost stejně rozsáhlé mapy. Díky tomu je možné k němu rychleji přistupovat, což může být stěžejní v aplikacích, které vyžadují velký počet snímků za sekundu. Nevýhodou je snížení kvality.

API OpenTopography

OpenTopography [5] je zdarma dostupné online API[1], které umožňuje získat topografická data celého světa ve vysokém rozlišení. Spravuje je Centrum superpočítačů San Diego, které je součástí Kalifornské univerzity. Je díky němu možné získat topografická data pro aplikace pracující na Google Earth, ale také rasterová data obsahující výškové mapy ve formátech GeoTIFF, AAIGrid a HFA v přesnostech SRTM1 i SRTM3. Hlavní výhodou OpenTopography je možnost určení rozsahu požadované výškové mapy za pomocí souřadnic WGS84.

API Overpass

Overpass [9] je volně dostupné API, které slouží k získávání vybraných částí map z OpenStreetMap (projekt pro vytvoření volně dostupných a upravitelných map světa a zejména geologických dat, které obsahují). Overpass API funguje jako webová databáze, které uživatel pošle dotaz, ve kterém specifikuje která data, v jaké oblasti a v jakém formátu požaduje a databáze mu vrátí požadovaný soubor dat. Mezi body zájmu, o kterých Overpass umožňuje získávat informace patří například veřejné instituce, budovy, cesty, vrcholy hor a kopců, povodí řek, města a vesnice nebo zastávky veřejné dopravy. O těchto bodech zájmu Overpass uchovává informace o zeměpisné poloze a další atributy podle jejich typu (např. název a velikost populace u měst a vesnic nebo nadmořskou výšku vrcholů kopců a hor). Tyto data Overpass umožňuje odesílat ve formátech .JSON a .XML.

2.3 Existující aplikace

Existuje mnoho webových i mobilních aplikací, které vytváří 3D model terénu za pomocí geologických dat zadávaných manuálně nebo pomocí senzorů v mobilním zařízení. Tyto senzory dokáží určit aktuální pozici a směr pohledu. Po spojení těchto informací s databází terénu jsou aplikace schopné vytvořit model který si uživatel může prohlédnout. Většina z

[1] Rozhraní, které určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu.

těchto aplikací umožňuje také offline režim, pro který je ovšem potřeba stáhnout databázi terénu do zařízení. Hlavní nevýhodou níže uvedených aplikací je, že nemají volně přístupný kód a není tedy možné na jejich základech vyvíjet nové aplikace nebo vylepšení.

PeakVisor

Webová i mobilní aplikace, která pro zadanou polohu na mapě vytvoří model okolního terénu. Zobrazuje souřadnice aktuální pozice, její nadmořskou výšku, směr pohledu, zajímavé vrcholky v blízké vzdálenosti a topografickou mapu okolí s vrstevnicemi. Samotný model ukazuje názvy a výšku vrcholů, profily hor jsou v modelu zvýrazněny. Po rozkliknutí konkrétního hory se zobrazí její informace, jako prominence (převýšení mezi vrcholem a nejvyšším sedlem hory), vzdušná vzdálenost od aktuální pozice, informace z Wikipedie a prohlídku 3D modelu této hory. Existuje také v placené mobilní verzi, kde aplikace využívá fotoaparát a GPS k zobrazování informací o vrcholech v reálném čase. Obsahuje 3D mapy a umožňuje stahování databází a tedy využívání aplikace offline. Za zmínku také stojí možnost přidání popisků do fotky manuálním nalezením pozice pořízení nebo pomocí geografických dat v ní obsažených. Ukázka webové aplikace je na obrázku 2.3.

PeakFinder

Webová i placená mobilní aplikace vytvářející jednoduchý, na hardware nenáročný model terénu. Stejně jako aplikace PeakVisor zobrazuje informace o současné poloze jako souřadnice, nadmořskou výšku, zeměpisný směr pohledu, ale i časy, kdy vychází a zapadá slunce a měsíc. Model obsahuje zvýrazněné obrysy hor a názvy jejich vrcholů. Po rozkliknutí vrcholu se zobrazí jeho výška, zeměpisná pozice, vzdálenost od aktuálního bodu, stát, ve kterém se nachází a tlačítko, kterým je možné přejít na tento vrchol. Umožňuje také přibližování na určitou část modelu, zvýšení kamery až o 1000 metrů a zobrazení sluneční a měsíční oběžné dráhy. Ukázka webové aplikace je na obrázku 2.3.

PeakIdentify

Mobilní aplikace, která umožňuje identifikovat vrcholy hor ze starých fotografií, ale i živého přenosu obrazu z fotoaparátu. Používá k tomu automatickou detekci profilu hory, nebo jeho ruční nákres. Aplikace poté najde shodu s tímto profilem v databázi a je schopna podle ní určit lokaci, zeměpisný směr a natočení kamery. Aplikace obsahuje také editor, ve kterém je možné upravit popisky, pokud jsou nesprávné, například kvůli mrakům v pozadí a určit pozici slunce nebo měsíce v zadaný den a čas.

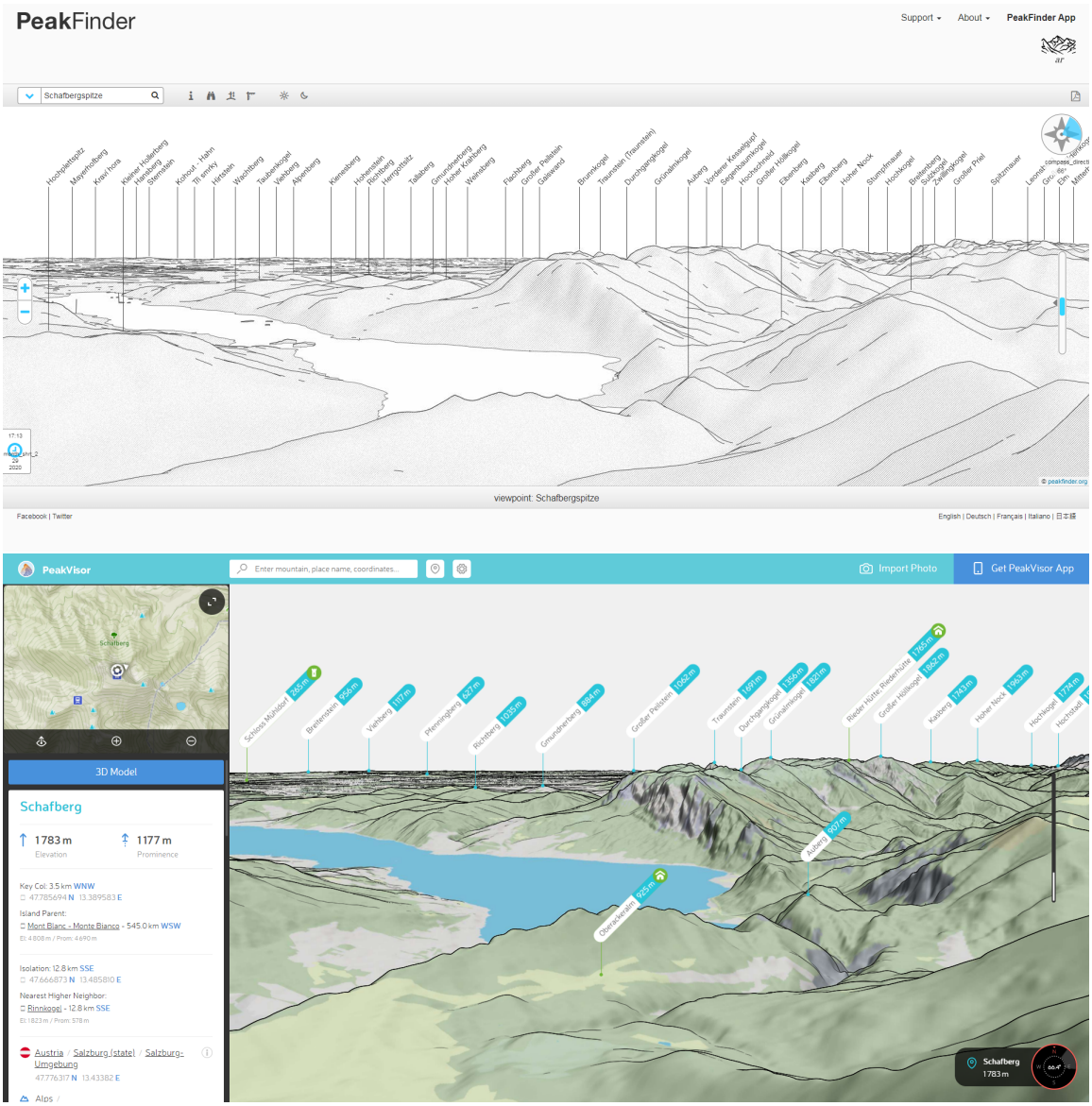
PeakLens

Jednoduchá mobilní aplikace s příjemným uživatelským rozhraním k zobrazování názvů vrcholků terénu. Kromě toho ukazuje také zeměpisný směr, na který fotoaparát míří. Umožňuje pořizovat fotografie s vloženými popisky vrcholů. Ve srovnání s konkurencí obsahuje menší databázi míst, která je ale rozdělena na jednotlivé státy a je možné její částí stáhnout offline.

GeoFlyer

Mobilní aplikace umožňující mimo jiné volný přelet 3D modelu vytvořeného z mapy. Zobrazuje zajímavosti v okolí a jejich popis z Wikipedie. Pro lepší vizualizaci modelu používá

vrstevnice. Zobrazuje geografické informace o zvoleném bodu. Aplikace také umožňuje naplánovat si trasu mezi dvěma body a vypsat o ní informace, jako převýšení v průběhu zdolané vzdálenosti.



Obrázek 2.3: Ukázka webových aplikací - shora: PeakFinder, PeakVisor

Kapitola 3

Vývoj aplikací pro operační systém Android

V následující kapitole je popsán operační systém Android, jakým způsobem na něm probíhá vývoj aplikací, jaké vývojové prvky se k tomu používají a jaké jsou využívány nástroje. Dále kapitola obsahuje informace o API k vykreslování grafiky OpenGL ES a knihovnách použitých při vývoji aplikace.

3.1 Operační systém Android

Operační systém Android [1] je založený na modifikované verzi Linuxového jádra ve spojení s dalším open source softwarem. Je navržený zejména pro mobilní zařízení s dotykovou obrazovkou, jako jsou mobilní telefony a tablety. V poslední době se ovšem používá i v chytrých televizích a nositelných chytrých zařízeních. Je rozvíjen uskupením vývojářů známých jako Open Handset Alliance, kteří se zabývají vytvářením otevřených standardů pro mobilní zařízení. Výrobci zařízení tedy mohou systém Android při dodržení stanovených podmínek upravovat (například MIUI od společnosti Xiaomi). Hlavním přispěvatelem a podporovatelem je společnost Google, která Android koupila od společnosti Android Inc. První zařízení s OS Android bylo odhaleno v roce 2008. Aktuální verze Androidu je 10.0. byla představena v září 2019.

Aktuálně operační systém Android používá cca 75 % mobilních zařízení. Jeho hlavním konkurentem je operační systém iOS společnosti Apple. Společně jsou tyto dva produkty používány skoro na 99 % mobilních zařízení na celém světě [13].

Programovací jazyky

Původně operační systém Android podporoval vývoj aplikací pouze v jazyce Java. Postupem času ovšem přibývala podpora pro další programovací jazyky. Hlavní programovací jazyky použitelné pro vývoj na Androidu jsou [10]:

Java

Java je objektově orientovaný programovací jazyk. Jde o jeden z nejpoužívanějších programovacích jazyků na světě. Běží nad JVM (Java Virtual Machine). Díky své přenositelnosti je používán pro vývoj aplikací na mnoha platformách.

Kotlin

Programovací jazyk Kotlin byl představen v roce 2017 a o dva roky později nahradil Javu jako preferovaný jazyk pro vývoj Androidových aplikací. Jedná se o je staticky typovaný programovací jazyk běžící, stejně jako Java, nad JVM. I když není syntakticky kompatibilní s Javou, je navržen pro interoperabilitu s jejími knihovnami.

C/C++

S využitím Java NDK (Native Development Kit) je možné vyvíjet aplikace také v jazycích C a C++. Části kódu v těchto jazycích jsou vykonávány přímo procesorem. Jejich používání je tedy výhodné zejména při operacích, které by v jazycích využívajících JVM mohlo být náročné na výkon.

C#

C# je vysokoúrovňový objektově orientovaný programovací jazyk, který se při vývoji Androidových aplikací hodí zejména pro práci s platformou Xamarin a herním enginem Unity 3D.

Vývojové prvky aplikací na OS Android

Activity

Aktivita je akce, kterou může uživatel provést. Typicky je to jedna obrazovka uživatelského rozhraní aplikace a podobá se tedy oknům v operačních systémech stolních počítačů, ale není to pravidlem, mohou být například vloženy do jiných oken. Znázornění životního cyklu aktivity je na obrázku 3.1.

Intent

Intent je mechanismus pro přepínání aktivit a pro přenos dat mezi nimi. Jelikož jedna je jedna aktivita často reprezentována jednou obrazovkou, intent se používá i pro přepínaní mezi nimi. Existují dva typy:

- Implicitní, při jehož použití není specifikovaná cílová aktivita, ale pouze operace, která má být provedena. Operační systém nebo uživatel poté vybere aplikaci, která může intent vykonat.
- Explicitní, při kterém je přímo použita konkrétní komponenta v rámci aplikace (např. aktivita), která intent vykoná.

Layout

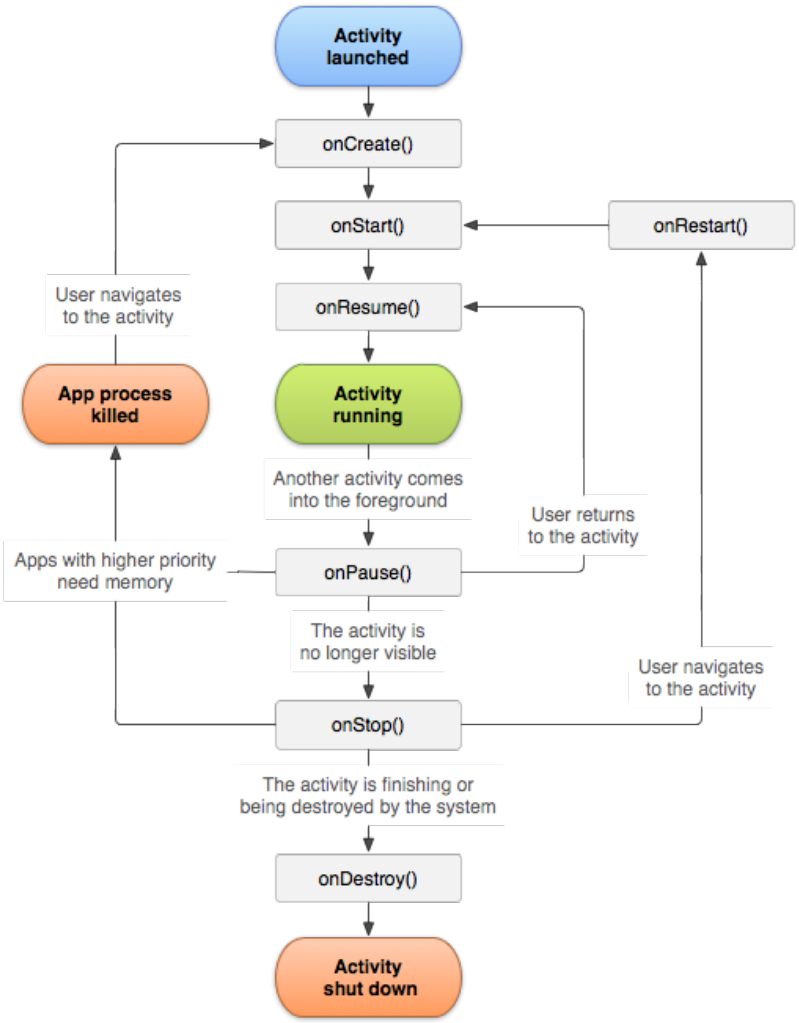
Layout popisuje vzhled obrazovky, tedy uživatelského rozhraní. Je popsaný ve značkovacím jazyce XML. Určuje jak jednotlivé elementy vypadají, jakou mají pozici a vlastnosti.

Android Manifest

Povinný soubor všech aplikací na Android ve formátu .XML, který se musí nacházet v kořenovém adresáři. Obsahuje informace o aplikaci, jako třeba seznam komponent, používaných knihoven a senzorů nebo oprávnění, které aplikace požaduje ke správnému chodu.

Resources

Složka, obsahující dodatečné soubory a neměnný obsah, který aplikace používá. Jsou to například definice rozložení, bitmapy, definice barev nebo fonty. Složka zdrojů obsahuje také podsložku raw, ve které jsou uloženy libovolné soubory v jejich surové formě [3].



Obrázek 3.1: Znázornění životního cyklu aktivity [14].

Android Studio

Android Studio [2] je vývojové prostředí založené na IntelliJ IDEA. Bylo představeno společností Google v roce 2013. Je zdarma k dispozici pro uživatele na platformách Windows, Mac OS X a Linux. Mezi jeho hlavní vlastnosti patří:

- Vizuální editor rozložení umožňující vytvoření nákresu aplikace pouhým přetažení jednotlivých prvků uživatelského rozhraní a automatické vygenerování XML souboru. Je možné nastavit dynamické hodnoty rozložení prvků, aby aplikace fungovala správně na různých zařízeních. Editor také umožňuje zobrazit, jak bude návrh vypadat na zařízeních s různými velikostmi obrazovky a na různých verzích systému Android.

- APK analyzátor, který poskytuje okamžitý náhled o kompozici APK souboru[1] po dokončení sestavení projektu. Díky tomu ulehčuje debugging DEX souborů[2], využívání zdrojů zařízení a výslednou velikost aplikace.
- Flexibilní systém sestavení založený na systému Gradle, díky kterému je možné přizpůsobit sestavení aplikace tak, aby bylo z jednoho projektu vygenerováno více variant aplikace pro různá zařízení.
- Inteligentní editor kódu založený na IntelliJ IDEA.
- Emulátor systému Android běžící v rámci Android Studia, který umožňuje virtuálně testovat aplikace na různých zařízeních s různými verzemi systému Android bez nutnosti tyto zařízení vlastnit. Emulátor umožňuje simulovat skoro všechny vlastnosti fyzického zařízení, jako různé rychlosti připojení k síti, rotaci zařízení, určení jeho pozice a simulaci dalších hardwarových senzorů. Testování aplikace s použitím emulátoru je jednodušší a rychlejší, než testování na fyzickém zařízení, protože odpadá potřeba přenášet data pomocí USB kabelu.
- Android Profiler, monitorovací zařízení, které v reálném čase zobrazuje data o využívání CPU, paměti, síťovém provozu a baterii jak na fyzickém, tak i na zařízení běžícím pomocí emulátoru.

Android NDK

Součástí Android Studia verze 2.2 a výše je také Android NDK (Native Developement Kit), který umožňuje používat knihovny napsané v jazyce C a C++. NDK je vhodné používat zejména při výkonnostně náročných operacích nebo při potřebě použít knihovny, které nebyly vyvíjeny konkrétně pro systém Android. Nárůst výkonu se projevuje tím, že kód v jazyce C nebo C++ je zkompilován do binárního kódu, zatímco kód v jazyce Java nebo Kotlin předtím musí být převeden do virtuálního stroje JVM (Java Virtual Machine).

CMake

K překladu nativního kódu v NDK je používán CMake, který je součástí Android Studia. Je to open source multiplatformní soubor nástrojů k sestavení programu. Používá se k vytvoření hierarchie adresářů a přípravu zdrojových souborů, závisejících na vícero knihovnách, pro použití s konkrétními překladači, vázanými na operační systém, například program make na Unixových systémech, Xcode na systémech firmy Apple nebo Microsoft Visual Studio na systému Windows.

Java Native Interface

Java Native Interface (JNI) je rozhraní umožňující propojit kód běžící na virtuálním stroji Javy s nativními programy a knihovnami napsanými v jiných jazycích, jako jsou například C, C++ nebo assembler. Využívat nativní kód namísto jazyka Java se hodí v případě, kdy je potřeba pracovat s platformou nižší úrovně, což JVM neumožňuje. Takové případy mohou nastat, pokud je zapotřebí přímo komunikovat s hardwarovými zařízeními nebo pokud je

[1] Instalační soubor aplikací na OS Android vytvořený po kompilaci programu, který obsahuje všechny části aplikace zabalené do jednoho kontejnerového souboru.

[2] Soubory obsahující bytový kód vytvořený kompilací kódu v Java Virtual Machine

potřeba využít starší knihovnu napsanou v nativním kódu a nebo z ní přistupovat k aplikaci běžící v jazyce Java. JNI tedy umožňuje používat hardware, který nemá rozhraní v Javě, jako jsou například čtečky čárových kódů nebo platebních karet. Mezi neposlední výhodu patří, že ód zkompilovaný v nativním prostředí se provádí rychleji než v JVM. To je výhodné u náročných výpočetních operací nebo vykreslování grafiky. Mezi nevýhody patří ztráta nezávislosti na platformě, což patří mezi jeden z hlavních benefitů jazyka Java. V tomto případě je nutné znovu překládat nativní knihovny pro každé hostitelské prostředí. Další nevýhodou jsou nevýhody pramenící z nativních jazyků jako je uvolňování nepotřebných zdrojů z paměti nebo typová bezpečnost.

Spojení pomocí JNI v případě, kdy je nativní kód volán z aplikace v jazyce Java, probíhá vytvořením nativní metody, která je volaná stejným způsobem jako všechny ostatní metody. Ve skutečnosti je ale prostřednictvím JNI zavolána metoda jejíž implementace je napsána v nativním kódu a uložena v nativní knihovně.

Knihovna EasyPermissions

EasyPermissons je externí knihovna od společnosti Google, která zabaluje a zjednodušuje funkce pro ověřování a získávání oprávnění pro aplikace na systému Android.

3.2 OpenGL ES

OpenGL ES [6] je multiplatformní API vyvíjené společností Khronos Group pro vykreslování pokročilé 2D a 3D grafiky na vestavěných a mobilních zařízeních, jako jsou například konzole, mobilní telefony, spotřebiče a automobily. Je složený z vybraných částí OpenGL pro stolní počítače, které jsou vhodné pro nízkoenergetická zařízení s omezeným výkonem. Poskytuje flexibilní a výkonné rozhraní mezi softwarem a hardwarem pro akceleraci grafiky.

Přehled verzí

OpenGL ES 1.0

První verze API OpenGL ES z roku 2003, vychází z OpenGL 1.3 pro stolní počítače. Mnoho funkcionality OpenGL 1.3 bylo odebráno, aby výkon mobilních zařízení té doby byl dostatečný pro operace s knihovnou.

OpenGL ES 2.0

Verze z roku 2007, vychází z OpenGL 2.0. Není zpětně kompatibilní s verzí 1.0 a 1.1. Téměř všechny vlastnosti vykreslování, které byly v předchozích verzích specifikovány pomocí API s fixní množinou funkcí, byly nahrazeny tzv. shadery, vytvářenými programátorem. Tato verze je stále často používána, protože funguje na velkém podílu aktuálně používaných mobilních zařízení, konkrétně na Androidu verze 2.2 a výše.

OpenGL ES 3.0

Vydána v roce 2012, je zpětně kompatibilní s verzí 2.0. OpenGL 4.3 poskytuje plnou kompatibilitu s OpenGL ES 3.0. Podporuje novou verzi GLSL ES shader jazyka s plnou podporou celočíselných a desetinných operací v 32bitové přesnosti. Funguje na zařízeních s Androidem verze 4.3 a výše.

OpenGL ES 3.1

Verze 3.1 byla vydána v roce 2014. Vychází z OpenGL 4.4 a je zpětně kompatibilní s OpenGL ES verze 2.0 a 3.0. Je funkční na zařízení s Androidem verze 5.0 a výše. Přidává například možnost vykreslování více cílů současně nebo jednotnou vyrovnávací paměť.

OpenGL ES 3.2

Aktuální verze vydaná v roce 2015. Přidává funkcionalitu založenou na Android Extension Pack for OpenGL ES 3.1, díky čemuž se funkcionalita mobilního API velmi přiblížila OpenGL pro stolní počítače [6].

OpenGL ES na systému Android

Pro práci s OpenGL ES na systému Android se používají dvě základní třídy, které umožňují vykreslování grafiky a manipulaci s ní.

Třída GLSurfaceView

Třída GLSurfaceView představuje komponentu, na kterou je možné zobrazovat grafiku z knihovny OpenGL ES. K samostatnému vykreslování se ale používá třída GLSurfaceView.Renderer. Třída zjednodušuje životní cyklus standardní aktivity, může být pouze vytvořena a zničena nebo pozastavena a obnovena v případě, kdy je aktivována jiná aktivita a je zapotřebí správně uvolňovat zdroje. Třída umožňuje jak statické zobrazování grafiky, tak zobrazování za běhu. Pro zobrazování používá dedikované vlákno, aby bylo odděleno od vlákna uživatelského rozhraní a nedocházelo k omezení výkonu.

Třída GLSurfaceView.Renderer

Třída GLSurfaceView.Renderer se používá pro samotné vykreslování grafiky pomocí OpenGL ES resp. jednotlivých snímků. Typicky je zavolána třídou GLSurfaceView.

3.3 OSG Earth

OSG Earth [8] (OpenSceneGraph Earth) je geoprostorové SDK[3] a engine pro vykreslování terénu pro aplikace využívající OpenSceneGraph. Dalšími potřebnými knihovnami ke spuštění OSG Earth jsou GDAL a LibCURL.

Hlavní vlastnosti OSG Earth jsou:

- Umožňuje jednoduše vykreslovat terén a texturu celého světa přímo ze zdrojů dat a to buď offline nebo dynamicky za běhu.
- Caching částí map pro optimalizaci výkonu.
- Podporuje kombinování více zdrojů snímkových i výškových dat během běhu aplikace a kombinování snímků do vrstev zobrazení.
- Měnit průhlednost jednotlivých vrstev zobrazení.

[3]Software development kit, tedy sada vývojových nástrojů umožňující vytváření aplikací pro určité frameworky, operační systém nebo jiné platformy.

- Překrývání terénu daty z geografických informačních systémů.
- Umísťování 3D modelů do modelu terénu.

OpenSceneGraph

OpenSceneGraph [7] (dále jen OSG) je multiplatformní open source sada nástrojů pro vývoj 3D grafických aplikací například her, simulátorů, virtuální reality nebo vytváření 3D modelů. Je celá napsána v jazyce C++ a díky tomu běží na velkém množství běžně používaných operačních systémů jako Windows, macOS a Linux, ale i na mobilních operačních systémech Android a iOS.

OSG podporuje OpenGL verzí 1.0 až 4.2 a OpenGL ES 1.1 a 2.0, díky čemuž je podporováno velké množství zařízení, včetně starších a méně výkonných typů.

Mezi hlavní výhody OSG patří:

- Výkonnost - OSG podporuje view-frustum culling[4], occlusion culling[5], small feature culling[6], Level Of Detail[7] a další techniky, díky kterým patří OSG mezi nejvýkonnější dostupné grafické nástroje.
- Škálovatelnost - OSG funguje jak na mobilních zařízeních, tak také na výkonných stanicích s více grafickými procesory a to díky podpoře více grafických kontextů a multithreadingu.
- Produktivita - OSG zabaluje většinu funkcionality OpenGL, obsahuje optimalizaci vykreslování a knihovny, díky kterým je možné vyvíjet aplikace, aniž by se musel vývojář zabývat programováním na nízké úrovni.

GDAL

GDAL (Geospatial Data Abstraction Library) je open source multiplatformní knihovna, napsána v jazyce C, pro čtení a zápis rastrových a vektorových dat různých formátů využívaných geografickými informačními systémy.

[4] Odstraňování objektů, které leží kompletně mimo zorné pole kamery.
[5] Odstraňování objektů, které jsou kompletně schovány za ostatními objekty v zorném poli kamery.
[6] Odstraňování objektů, které by byly v zorném poli kamery příliš malé.
[7] Snižování detailů objektu s narůstající vzdáleností od kamery.

Kapitola 4

Návrh aplikace

V této kapitole je popsaný kompletní návrh aplikace. Popisuje případy užití, návrh uživatelského prostředí a návrh samotné aplikace. Je v ní také popsáno využití knihovny OSG Earth a návrh jednotlivých funkcí aplikace. Životní cyklus aplikace je znázorněn na obrázku 4.3.

4.1 Případy užití

Hlavní funkcí vyvíjené aplikace je zobrazování členitosti terénu a názvů měst na obrazovce zařízení. Uživatel může ve vertikálním i horizontálním režimu rotovat kamerou pomocí natáčení zařízení. Pomocí gesta přiblížení dvěma prsty může uživatel přibližovat a oddalovat kameru. Tlačítkem v pravém horním rohu může přepnout na pohled ze zadní kamery a zpět. Tlačítkem exit může ukončit aplikaci.

4.2 Uživatelské rozhraní

Vzhledem k tomu, že se veškeré chování aplikace objevuje jen v jednom okně, je uživatelské rozhraní aplikace jednoduché. Po instalaci a prvním spuštění aplikace vyskočí dialogová okna požadující oprávnění k používání prvků hardware, konkrétně GPS a kamery.

Aplikace poté začne vyhledávat aktuální pozici a stahovat potřebná data. Během tohoto je na obrazovce vykresleno dialogové okno čekání. Po nalezení polohy se spustí samotné okno aplikace, tedy vykreslená členitost terénu v okolí zařízení a názvy okolních měst a vesnic. Toto okno je po celou dobu běhu aplikace v režimu celé obrazovky, a obsahuje ještě ukazatel aktuální výšky nad mořem v levém spodním rohu a dvě tlačítka v levém horním a pravém horním rohu. Tlačítko v levém horním rohu, stejně jako tlačítko zpět slouží k ukončení aplikace. Tlačítko v pravém horním rohu slouží k přepínání mezi modelem terénu a zobrazením přenosu z přední kamery zařízení.

Celá aplikace, zobrazení modelu terénu i přenosu z přední kamery, funguje v obou polohách zařízení, tedy ve vertikálním i horizontálním režimu.

4.3 Funkce aplikace

V následující sekci je popsán návrh aplikace v jazyce Java. Popis je zaměřen na životní cyklus aplikace, používání senzorů, propojení s nativní knihovnou a důvody k vybrání konkrétního typu dat.

Aktivita MainActivity

Aktivita *MainActivity* je spuštěna při startu aplikace. Jejím účelem je získání potřebných oprávnění pro správné fungování aplikace. K tomu je použita knihovna EasyPermissions, která zjednodušuje práci s oprávněními a obsahuje i další užitečné nástroje jako vytváření dialogových oken. Při prvním startu aplikace vyskočí dialogová okna pro získání oprávnění. V případě odmítnutí následuje další dialogové okno, ve kterém je možnost změnit svoji volbu, protože bez povolení oprávnění není aplikace schopna správně fungovat. Pokud se uživatel stejně rozhodně oprávnění neudělit, vyskočí poslední dialogové okno, které ho může přenést do nastavení aplikace, kde může uživatel tato oprávnění nastavit manuálně. V opačném případě je spuštěna aktivita *GPS* a následně aktivita *SRTM*, které jsou aktivní dokud není nalezena přesná lokace a stažena výšková mapa. Poté jsou tyto dvě aktivity ukončeny, aktivita *MainActivity* je obnovena a v případě, že vše proběhlo v pořádku, se spustí aktivita *NativeViewer* a aktivita *MainActivity* je ukončena.

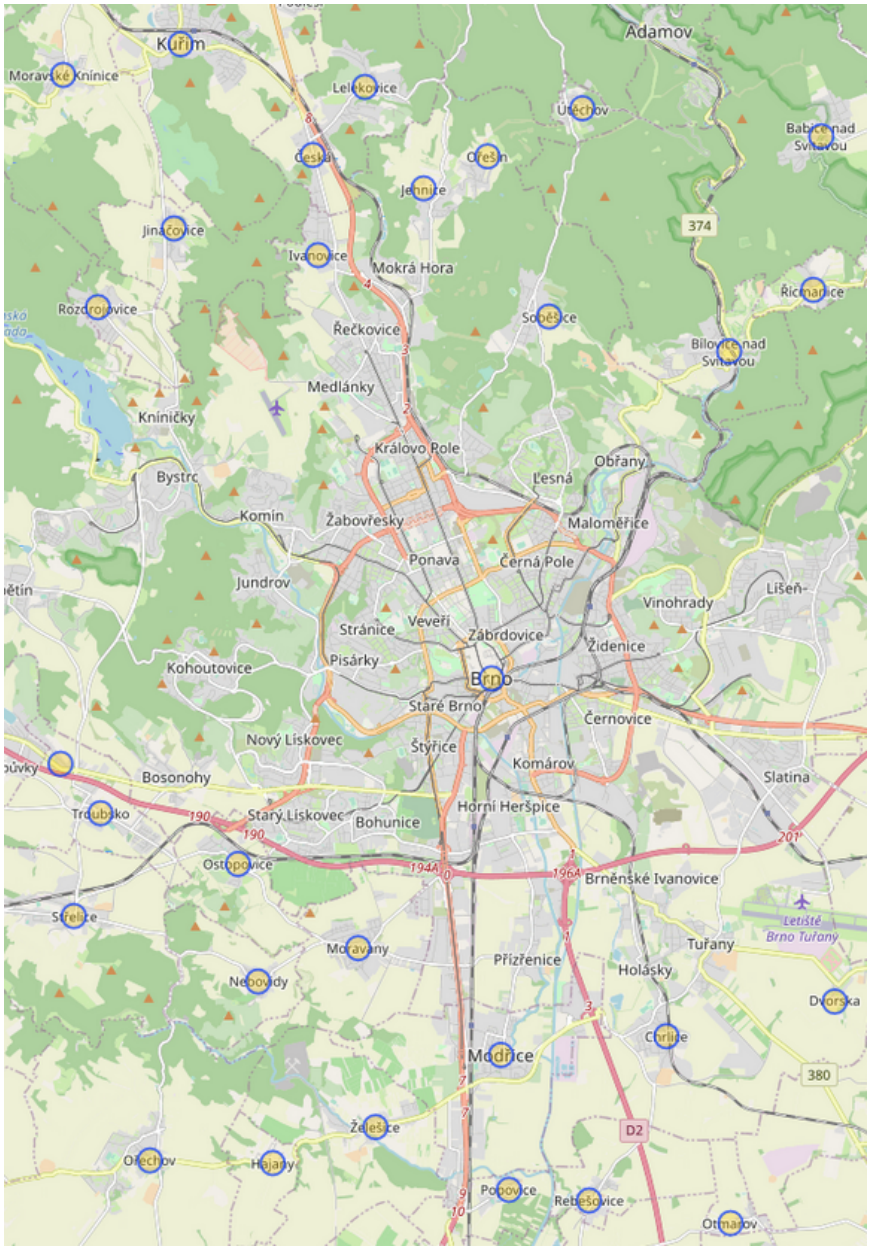
Aktivita GPS

Aktivita *GPS* nejdříve zkontroluje, jestli má aplikace potřebná oprávnění pro používání GPS senzoru v zařízení. Poté vyhledá přesnou pozici zařízení. Pro používání aplikace nestačí přibližná lokace, zjistitelná z internetu nebo mobilních sítí, protože by poté mohla být kamera v aplikaci umístěna až 2000 metrů od správné polohy zařízení. Z tohoto důvodu senzor GPS nemusí být schopen získat data v zejména vysokých budovách. Po připojení k satelitům GPS a získání polohy je spojení i příjem signálu zastaven z důvodu úspory baterie. Získané informace o poloze jsou uloženy ve veřejných proměnných a odeslány do nativní knihovny. S pomocí získaných údajů o lokaci je spuštěn asynchronní úkol, který zajišťuje spojení s Overpass API a získání požadovaných metadat. Požadovanými metadaty jsou informace o všech obcích a městech, které se nacházejí v oblasti desetiny stupně zeměpisné šířky a desetiny stupně zeměpisné délky na obě strany od polohy zařízení. Tato oblast typicky obsahuje 10 - 50 uzlů. Výsledek hledání pro oblast se středem ve městě Brno je na obrázku 4.1. Tyto data jsou následně zpracována a jsou uloženy názvy a zeměpisné polohy pro všechny nalezené vesnice a města. Následuje spuštění aktivity *SRTM*, po jejím zdárném ukončení je obnovena a následně ukončena aktivita *GPS*.

Aktivita SRTM

Aktivita *SRTM* vytvoří požadavek na OpenTopography API ke stažení výškové mapy v okolí aktuální lokace zařízení. Tuto mapu poté stáhne, správně pojmenuje a uloží na lokální úložiště. Předtím ale zjistí, jestli už se soubor s tímto jménem v úložišti nenachází, v takovém případě je před zahájením stahování smazán.

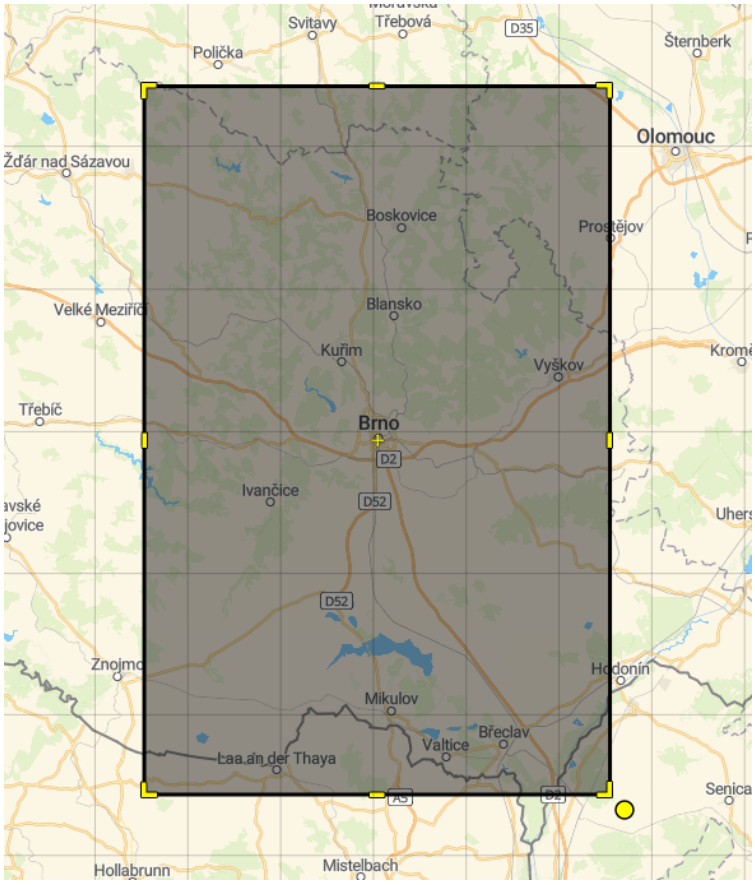
OpenTopography umožňuje získat výškové mapy v rozlišení SRTM1 i SRTM3, bylo tedy nutné vybrat, které z nich bude aplikace stahovat. Vzhledem k úspoře spotřeby dat a rychlosti stažení bylo vybráno rozlišení SRTM3. Bylo také potřeba určit rozsah stahované mapy. Vzhledem k povaze aplikace, která má zobrazovat jen okolní terén z vlastního pohledu, nebylo zapotřebí stahovat příliš velký okruh. Byla proto vybrána oblast o půl stupně zeměpisné šířky a o půl stupně zeměpisné délky na obě strany od polohy zařízení. Tato oblast pro případ umístění zařízení v centru Brna je velká asi $8000 km^2$. Znázornění se nachází na obrázku 4.2. Velikost této mapy je v tomto případě asi 3MB a podobná velikost se dá očekávat pro všechny oblasti na světě s výjimkou oblastí v blízkosti obou pólů a oblastí s velkým procentem vodních ploch ve výšce moře.



Obrázek 4.1: Uzly nalezené v oblasti desetiny stupně zeměpisné šířky a desetiny stupně zeměpisné délky na obě strany od středu ve městě Brno.

Aktivita NativeViewer

Aktivita *NativeViewer* obsahuje vytváření uživatelského prostředí, včetně okna pro vykreslování grafiky v OpenGL ES a spouští vykreslování terénu pomocí třídy *EGLview*. Obsahuje také nastavení pohybových senzorů zařízení a jejich naslouchače. Dále tato aktivita inicializuje pohled zadní kamery ze třídy *CameraPreview* a tlačítko, které proměňuje pohled z kamery a grafické okno. Po vytvoření okna pro vykreslování grafiky tato aktivita posílá nativní knihovně data získaná z pohybových senzorů zařízení.



Obrázek 4.2: Rozsah oblasti půl stupně zeměpisné šířky a půl stupně zeměpisné délky na obě strany od středu ve městě Brno

Třída CameraPreview

Třída *CameraPreview* obsahuje pohled zadní kamery zařízení a rotaci okna v závislosti na vertikální nebo horizontální poloze zařízení. Jelikož aplikace může fungovat ve všech polohách, je pro správné ovládání pohledu nutné v závislosti na aktuální poloze přepočítávat osy zařízení.

Třída EGLview

Třída *EGLview* má na starosti vytváření OpenGL ES pohledu a jeho konfiguraci. Obsahuje také nastavení senzoru doteku a jeho naslouchání. Po správném vytvoření pohledu tato aktivita také odesílá data doteku do nativní knihovny.

4.4 Nativní knihovna

V následující sekci je popsán návrh aplikace v nativní knihovně v jazyce C++. Popis je zaměřen na propojení s aplikací v jazyce Java a využívání funkcí knihovny OSG, OSG Earth a knihoven třetích stran, které využívají.

Třída OsgMainApp

Třída *OsgMainApp* obsahuje všechny funkce, které je možné zavolat z jádra aplikace v jazyce Java a vytváří tedy můstek pro komunikaci. Obsahuje funkce pro inicializaci okna, vykreslení snímku, nastavování zeměpisných souřadnic, nastavení a změnu azimutu, tedy úhlu mezi severem a směrem, kterým zařízení míří a náklonu zařízení. Obsahuje také funkci pro přibližování a oddalování kamery.

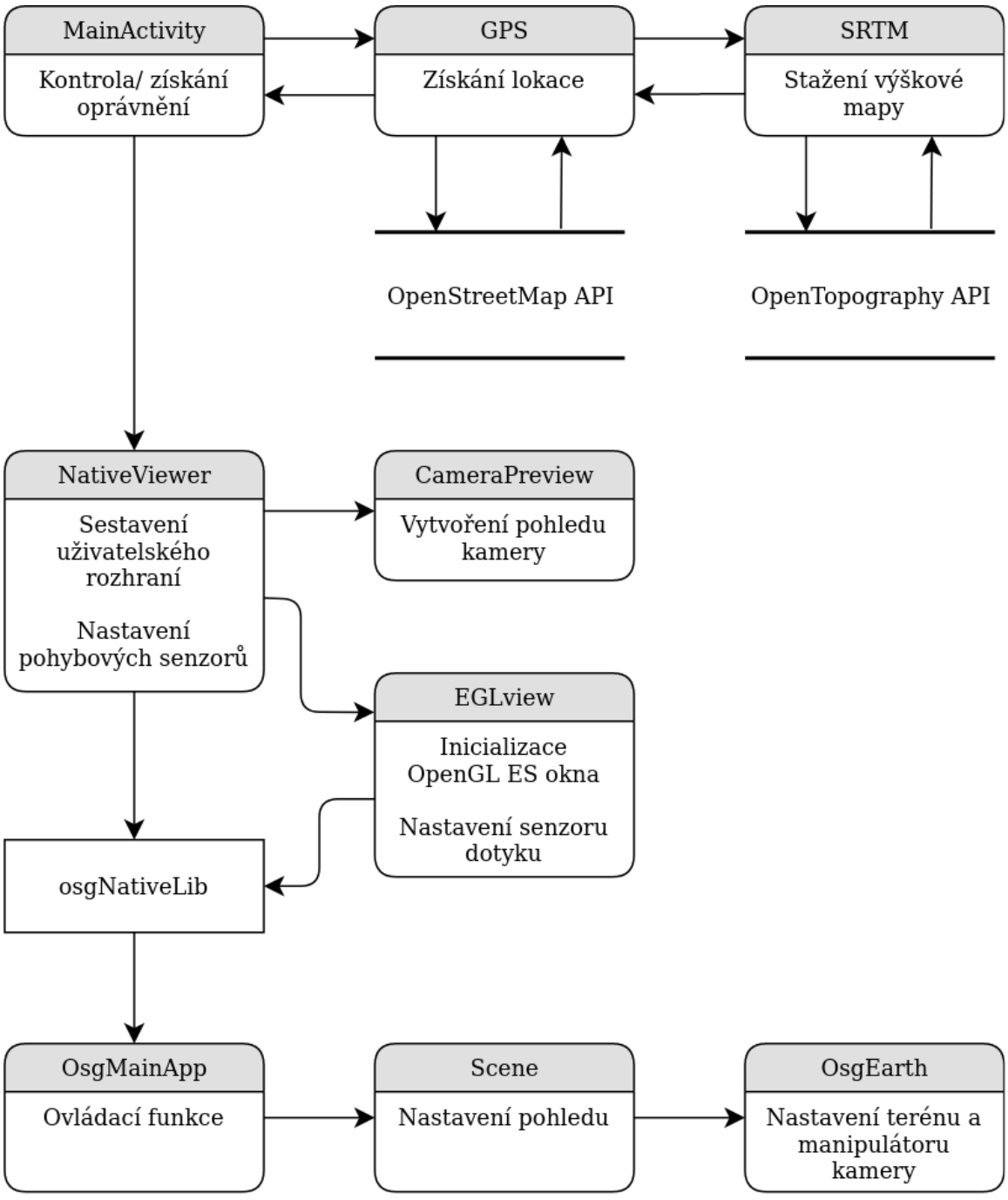
Třída Scene

Třída *Scene* se stará o inicializaci pohledu v OSG Earth, jeho nastavení a nastavení kamery. Tato třída má také na starost vykreslování jednotlivých snímků na obrazovku.

Třída OsgEarth

Třída *OsgEarth* má na starosti nastavení mapy, která bude zobrazována a přístup k datům které k tomu bude potřebovat. Mezi tato data patří obrázek povrchu planety, který je získáván přímo od společnosti Pelican Mapping, která zaštiťuje vývoj knihovny OSG Earth. Druhou částí dat jsou štítky, které zobrazují názvy měst a vesnic na jejich zeměpisných pozicích. Poslední částí dat je výšková mapa, která byla dříve získána v aktivitě *SRTM*. Dále tato třída obsahuje nastavení manipulátoru kamery, mezi které patří nastavení minimálního a maximálního náklonu kamery a také určení bodu na planetě, kde bude kamera umístěna, k tomuto jsou využity data z aktivity *GPS*.

Součástí umístění kamery je také její nadmořská výška určena v metrech nad mořem podle standardu WGS84. Z důvodu nepřesnosti určování výšky nad mořem pomocí senzoru GPS je kamera umístěna 25 metrů nad povrch země. Tímto je zamezeno možnosti umístění kamery pod terén.



Obrázek 4.3: Znázornění životního cyklu aplikace

Kapitola 5

Implementace

V této kapitole je popsána samotná implementace aplikace. Obsahuje popis zajímavých částí aplikace, použité algoritmy a také postup instalace potřebných knihoven.

5.1 Instalace OSG Earth

Ačkoliv je instalace a používání knihovny OSG Earth na operačním systému Android možné, za celou dobu co je vyvíjena nevznikl žádný ucelený postup k jejímu sestavení. K instalaci tedy bylo využito částí dvou skriptů (odkaz ve zdrojovém kódu). První z nich sloužil k instalaci knihovny OpenSceneGraph a všech ostatních knihoven, které využívá. Druhý skript instaloval samotnou knihovnu OSG Earth. K přeložení těchto knihoven a jejich instalaci bylo zapotřebí NDK verze r9d, které už nové verze Android Studia nepodporují, bylo proto nutné je získat z archivu starších verzí.

Při samotné instalaci došlo k několika problémům. Prvním z nich byla chyba při instalaci jedné z knihoven třetí strany, konkrétně knihovny CURL. Jednalo se o chybu testu na velikosti datových typů. Tato chyba byla zdokumentována v historii verzovacího systému a v následných verzích byla odstraněna. Řešením tohoto problému tedy bylo vyměnit zdrojové soubory knihovny CURL za novější verzi. Dalším z problémů bylo to, že některé z knihoven, které OSG Earth ke svému fungování potřebuje, fungují jen ve 32-bitové verzi, bylo proto nutné vynutit překlad programu ve 32-bitové verzi.

5.2 Implementace aplikace

V následující sekci je popsána implementace aplikace v jazyce Java. Popis je zaměřen na jednotlivé třídy, aktivity a rozložení aplikace, implementaci senzorů, propojení s nativní knihovnou, implementaci náhledu zadní kamery a vytvoření pohledu v OpenGL ES.

Aktivita MainActivity

Aktivita *MainActivity* je spuštěna při startu aplikace. Obsahuje kontrolu a přidělování oprávnění kamery, senzoru GPS a zapisování do interního úložiště. Z tohoto důvodu se tato aktivita chová rozdílně při prvním spuštění aplikace a spouštění následujících.

Aktivita při vytvoření nejdříve načte nativní knihovnu ze souboru OsgNativeLib, který obsahuje všechny nativní funkce, které je možné zavolat z jazyka Java. Poté zavolá funkci *requestLocationCameraStoragePermission*, která pomocí knihovny EasyPermissions zjistí,

jestli už jsou oprávnění aplikaci přidělena. Pokud přidělena nejsou vyvolá tato funkce dialogové okno pro přidělení oprávnění. Pokud uživatel odmítne tato oprávnění udělit, je vyvolána další okno, upozorňující, že aplikace bez nich nemůže správně fungovat. V případě že uživatel odmítne udělit oprávnění a zakřížkuje zapamatování volby, je přenesen do nastavení aplikace, kde je může udělit manuálně. V případě, kdy uživatel udělí požadovaná oprávnění, je spuštěna aktivita *GPS* a nastaven naslouchač *onActivityResult*, který čeká, dokud aktivita *GPS* nevrátí zprávu o svém úspěšném ukončení. Jakmile tato situace nastane, je spuštěna aktivita *NativeViewer* a aktivita *MainActivity* je ukončena.

Aktivita GPS

Aktivita *GPS* se nejdříve zjistí, jestli má potřebná oprávnění k používání senzorů pro získání polohy. V případě že ano, je nastaven naslouchač senzoru GPS. V moment kdy zařízení přijme informace o poloze, je ukončeno naslouchání, které už není zapotřebí, protože aplikace nepotřebuje za běhu získávat nová data. Údaje o poloze a nadmořské výšce jsou uložené do veřejných proměnných. Následně je spuštěn asynchronní úkol CallAPI. Poté je spuštěna aktivita *SRTM*, nastaven naslouchač *onActivityResult* a aktivita *GPS* je ukončena.

Asynchronní úkol CallAPI

Android od verze 6.0 vyžaduje, aby veškeré HTTP požadavky byly prováděny v asynchronním úkolu na pozadí. Získávání metadat z Overpass API je proto prováděno tímto způsobem. Úkol CallAPI nejdříve získá data o poloze z proměnných v aktivitě *GPS* a podle nich sestaví URL[1] požadavku na API. Rozsah prohledávané oblasti je daný tak, aby byly nalezeny jen uzly v oblasti přibližně dvaceti kilometrů od aktuální lokace, aby nebyla obrazovka zahlcena objekty, ale je možné tento rozsah jednoduše změnit přepsáním hodnoty, která je připočtena k proměnným, které obsahují polohu. Změnou celé URL s požadavkem by mohly být získány i jiná data, než jsou názvy měst a vesnic. Jedná se o všechny data, které Overpass API, či případně jiný zdroj poskytuje. Poté je provedeno spojení a získaná data jsou uložena do datového proudu, který je následně převeden na řetězec. Tento řetězec je rozebrán na objekty v formátu JSON[2]. Z těchto objektů jsou v cyklu získávány a následně odesílány do JNI informace o zeměpisné poloze a názvech měst a vesnic. Názvy jsou předtím ještě pomocí funkce *normalize* zbaveny diakritiky. Tím je vyřešen problém předávání řetězců z jazyka Java, které jsou kódovány formátem UTF-16[3], který používá 2 nebo 4 bajty pro reprezentaci jednoho znaku, do jazyka C++, kde jsou řetězce kódovány formátem UTF-8, který používá 1-6 bajtů pro reprezentaci jednoho znaku a přenos speciálních znaků tedy není přímo možný.

Aktivita SRTM

Aktivita *SRTM* v první řadě zjistí, jestli se na interním úložišti nachází soubor obsahující výškovou mapu a pokud ano, smaže jej. To stejné provede i se souborem readymap.earth, který obsahuje zdroje dat pro vykreslování v OSG Earth. Soubor readymap.earth je následně znovu vytvořen se správnou cestou k výškové mapě. Aktivita dále nastaví naslouchač

[1] Řetězec znaků s definovanou strukturou, který slouží k přesné specifikaci umístění zdrojů informací na Internetu.

[2] Datový formát určený pro přenos dat, která mohou být organizována v polích nebo agregována v objektech.

[3] Způsob kódování znaků, tedy přiřazení číselných kódů znakové sadě.

dokončení stahování a zavolá funkci *downloadFile* s parametry aktuální lokace. Tato funkce poté za pomocí předané lokace vytvoří URL na OpenTopography API. Rozsah stahované oblasti je vzhledem k povaze aplikace nastaven na oblast přibližně 35km od aktuální polohy, ale je možné tento rozsah změnit přepsáním hodnoty, která je připočtena k proměnným, které obsahují polohu. Nastavením proměnné *demType* na hodnotu 1 se dá změnit i požadavek na přesnost dat, který v aplikaci používá formát SRTM3. Formát SRTM3 byl vybrán zejména z důvodu úspory dat, protože výškové mapy ve formátu SRTM1 mohou být až 8x větší. Dalším důvodem byla rychlost vykreslování terénu v aplikaci. Aktivita poté stáhne výškovou mapu do interního úložiště a naslouchač dokončení stahování tuto aktivitu ukončí.

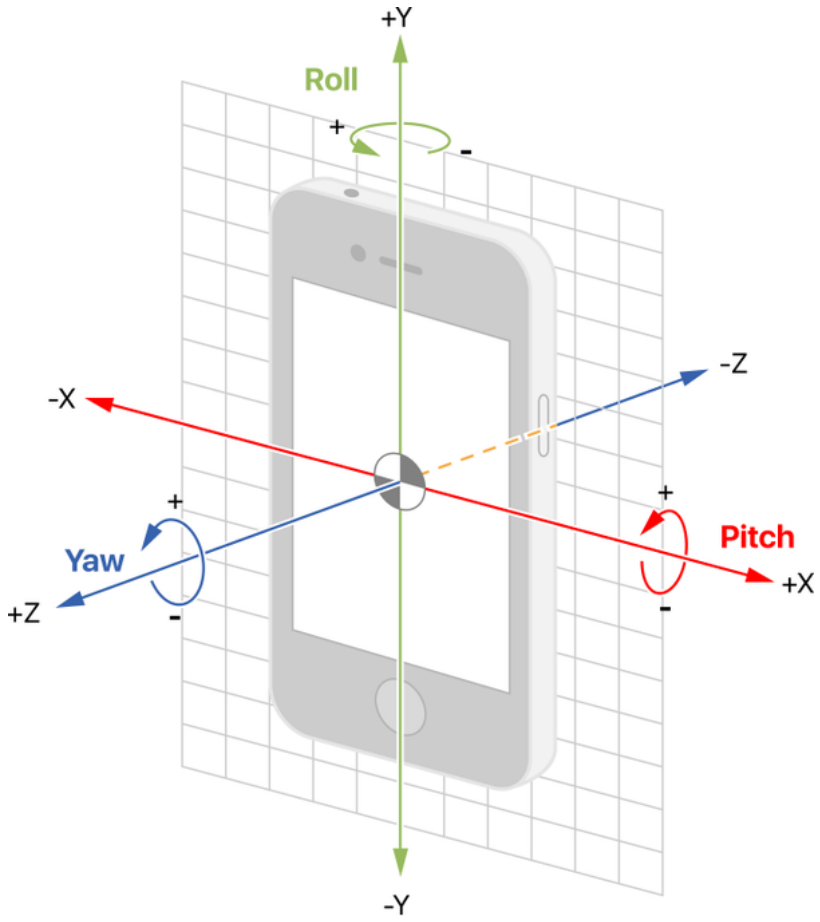
Aktivita NativeViewer

Aktivita *NativeViewer* nejdříve sestaví uživatelské rozhraní a zobrazí v textovém poli v levém dolním rohu obrazovky aktuální nadmořskou výšku zařízení. Poté získá cesty ke složce s daty a složce obsahující soubor readymap.earth a odešle je do JNI. Dále vytvoří novou kameru pomocí funkce *getCameraInstance*, pohled této kamery ze třídy *CameraPreview* a pohled ze třídy *EGLview*, který přidá na obrazovku a nastaví tlačítko pro výměnu tohoto pohledu za pohled ze zadní kamery. Aktivita poté nastaví naslouchače pohybových senzorů, tedy akcelerometru, magnetometru a gyroskopu. Pro určování směru, na který zařízení míří a jeho náklonu je použit akcelerometr a magnetometr. Je k tomu použita funkce *getRotationMatrix*, která vypočítá matici sklonu a rotační matici, která transformuje vektor ze souřadnicového systému zařízení na souřadnicový systém světa. Znázornění os zařízení ve vertikální poloze je zobrazeno na obrázku 5.1. V tomto případě je použita osa Z pro určení azimutu a osa X pro určení náklonu zařízení. Azimut může nabývat hodnot -180° až 180°. Jedná se o úhlový rozdíl mezi osou Z a severním pólem. V případě že zařízení směřuje přímo na sever je tato hodnota 0°. Náklon zařízení může nabývat hodnot -90° až 90°, kdy hodnota -90° značí zařízení ve vertikální poloze jak je znázorněno na obrázku 5.1 a hodnota 0° značí, že zařízení směřuje zadní stranou kolmo dolů k zemi.

V případě, kdy je zařízení orientováno jinak než vertikálně, například horizontálně nebo položené zadní stranou směrem k zemi, je použita funkce *getOrientation*, která zjistí orientaci zařízení a podle ní vymění osy k získání azimutu a náklonu. Informace o náklonu a azimutu jsou poté předány pomocí JNI do knihovny OSG Earth, která ovšem k umísťování kamery používá jiné hodnoty náklonu. Rozdíl mezi nimi je v tabulce 5.1. Je tedy zapotřebí tuto hodnotu upravit v závislosti na orientaci zařízení. Dále je také před odesláním od nativní knihovny zapotřebí přidat k získanému azimutu 90° v případě že je telefon v horizontálním režimu nebo odečíst 90° v případě kdy je v obráceném horizontálním režimu. V nativní knihovně jsou poté tyto data uložena v proměnných, které při inicializaci použije pro počáteční nastavení kamery.

Tabulka 5.1: Srovnání úhlů náklonu v systému Android a knihovně OSG Earth

Android	OSG Earth
-90°	0°
-45°	-45°
0°	-90°



Obrázek 5.1: Znázornění orientace zařízení ve vertikální poloze.

Pro otáčení kamery v OSG Earth je používán senzor gyroskopu, který získává informace o rotaci telefonu podél všech tří os v radiánech za sekundu. OSG Earth ovšem používá rozdílné jednotky náklonu a je zapotřebí získané hodnoty vydělit hodnotou 50. V případě, že jsou proměnné *eglInit*, značící úspěšnou inicializaci OpenGL pohledu a *headingInit*, značící nastavení počátečního natočení a náklonu kamery nataveny na hodnotu *true*, senzor použije data o rotaci z příslušných os podle orientace zařízení a zavolá funkci z JNI *changePitchHeading*, která vede k otáčení a rotaci kamery.

Třída CameraPreview

Třída *CameraPreview* vytvoří pomocí předané kamery pohled, který zobrazuje přenos ze zadní kamery zařízení. Tato třída obsahuje také funkci *setCameraDisplayOrientation*, která získá údaje o orientaci zařízení, která může být vertikální, horizontální, nebo obrácená horizontální a podle toho nastaví otočení náhledu ve stupních ve směru hodinových ručiček. Tím je docíleno toho, že přenos z kamery je ve stejné poloze jako zařízení.

Třída EGLview

Třída *EGLview* má na starost nakonfigurování a vytvoření OpenGL ES pohledu. Ačkoliv OSG Earth umožňuje vykreslování i v OpenGL ES 3.0, vzhledem k přenositelnosti na

co největší počet zařízení byla vybrána verze 2.0. Pro správné fungování knihovny OSG Earth je ovšem zapotřebí vytvořit vlastní kontext, který umožňuje vykreslování ve verzi 1.0. Po nastavení pohledu třída nastaví renderer, který se stará o inicializaci okna v nativní knihovně a následné volání funkce *step* z nativní knihovny, která slouží k vytvoření následujícího snímků, který poté renderer vykreslí. Před vykreslením prvního snímku se je nastavena proměnná *eglInit* na hodnotu true, tím jsou informovány senzory gyroskopu a kompasu, že mohou volat funkce pro ovládání kamery. Tato třída obsahuje také naslouchač dotyku, který v případě provedení gesta přiblížení a oddálení pomocí dvou prstů zavolá nativní funkci *zoom*, která slouží k přiblížení nebo oddálení kamery od jejího ohniska.

5.3 Implementace v nativní knihovně

V následující sekci je popsána implementace aplikace v nativní knihovně v jazyce C++. Popis je zaměřen na propojení s aplikací v jazyce Java a implementaci funkcí knihovny OSG, OSG Earth a knihoven třetích stran, které využívají.

Třída OsgMainApp

Třída *OsgMainApp* obsahuje všechny funkce, které je možné volat z prostředí v jazyce Java. Funkce, které mají parametry obsahující řetězce, ještě musí být přetypované z řetězců v jazyce Java do řetězců v jazyce C++ nebo C. Tento převod probíhá v třídě *OsgNativeLib*. Funkce *initOsgWindow*, která je volaná ze třídy *EGLview* slouží k vytvoření scény v knihovně OSG Earth. Funkce *draw*, volaná ze stejné třídy, slouží k vykreslení snímku na obrazovku, dále také nuluje proměnné obsahující informace o doteku na obrazovce. Funkce *touchZoomEvent* pomocí údajů o dotecích a funkce manipulátoru kamery *zoom* přibližuje nebo oddaluje kameru od jejího ohniska. Funkce *setCoords* přiřazuje globálním proměnným údaje o poloze a nadmořské výšce získané z aktivity *GPS*. Funkce *setPitchHeading* přiřazuje hodnotu obsahující počáteční náklon a směr zařízení získané ze senzorů akcelerometru a magnetometru globálním proměnným. Funkce *changePitchHeading* pomocí předaných hodnot z gyroskopu a manipulátoru kamery otáčí kamerou kolem jejího ohniska. Funkce *setCitiesData* přiřazuje globálním proměnným, které obsahují pole, hodnoty pozice a názvu měst a vesnic získaných v úkolu *CallAPI*.

Třída Scene

Třída *Scene* slouží k nastavení prohlížeče (viewer), což je základní prvek v OSG Earth. Všechny ostatní prvky, jako je manipulátor kamery, nastavení terénu a barev jeho povrchu, umístění popisků nebo třeba nastavení oblohy, jsou následně navázány na prohlížeč. Třída má také na starosti vykreslování jednotlivých snímků pomocí pohledu *EGLview* na obrazovku.

Třída OsgEarth

Třída *OsgEarth* obsahuje pouze funkci pro inicializaci scény. Tato funkce nejdříve získá cestu k souboru readymap.earth z globální proměnné, která byla nastavena v aktivitě *NativeViewer*. Tento soubor obsahuje cestu k souboru obsahující výškovou mapu SRTM.tif, který byl stažen aktivitou *SRTM* a také URL na server Readymap, který poskytuje snímky pro vykreslení textury terénu. Pomocí souboru readymap.earth je poté vytvořen mapový uzel. K prohlížeči je následně přiřazen *EarthManipulator*, který se používá k práci s kamerou.

Dále se ve funkci nastavuje operátor nápisů, který určuje velikost jejich fontu. Manipulátoru kamery je poté pomocí funkce *setHomeViewpoint* nastavena počáteční pozice, výška, směr a náklon kamery. K tomu jsou využity dříve nastavené globální proměnné. K mapovému uzlu je následně přiřazen objekt *SkyNode*, který nastavuje parametry oblohy. Poté jsou v cyklu na určenou pozici na mapě přidávány nápisy obsahující názvy vesnic a měst, které jsou získány z dříve nastavených globálních proměnných. V poslední fázi funkce je mapový uzel přiřazen k prohlížeči a obsahuje data k vykreslení na obrazovku.

Kapitola 6

Testování a výsledky

V následující kapitole jsou popsány základní informace o aplikaci, testování její hardwarové náročnosti a uživatelské testování. Dále tato kapitola obsahuje ukázky výsledné aplikace.

6.1 Informace o aplikaci

Aplikace byla vyvíjena v programovacích jazycích Java a C/C++. Je spustitelná na zařízeních s SDK verze 16, tedy Android 4.1 a výše. Z důvodu využívání knihoven fungujících jen v 32-bitové verzi, není možné tuto aplikaci umístit na Google Play, protože od 1.9.2019 vyžaduje jak 32-bitovou, tak 64-bitovou verzi. Aplikace využívá knihovnu OpenSceneGraph ve verzi 3.2.1 a knihovnu OSGEarth ve verzi 2.7. Instalační balíček APK má velikost 14,1MB a aplikace po nainstalování má velikost 51,9MB.

6.2 Testování hardwarové náročnosti

Hardwarová náročnost byla testována na mobilním telefonu Xiaomi Redmi 5 Plus. Jedná se o zařízení nižší třídy vydané v únoru 2018. Obsahuje osmijádrový procesor Qualcomm Snapdragon 625 pracující na frekvenci 2GHz, 4GB paměti RAM a baterii o kapacitě 4000mAh. Informace o využívání zdrojů telefonu aplikací byly monitorovány pomocí nástroje Profiler, který je součástí Android Studia.

Vzhledem k povaze aplikace nebylo očekáváno, že výsledky jednotlivých testů budou zásadně odlišné. Bylo proto provedeno 5 testů v různých lokalitách a jejich výsledky byly následně zprůměrovány. Testování bylo prováděno s maximálním jasem obrazovky. Prvním monitorovaným zdrojem bylo využití procesoru znázorněné v tabulce 6.1. Druhým monitorovaným zdrojem bylo využití paměti RAM znázorněné v tabulce 6.2. Spotřeba dat při běhu aplikace je znázorněna v tabulce 6.3 a spotřeba baterie je znázorněná v tabulce 6.4.

Tabulka 6.1: Využití procesoru

	Test 1	Test 2	Test 3	Test 4	Test 5
CPU průměr	11%	11%	11%	9%	10%
CPU minimum	1%	1%	2%	1%	2%
CPU maximum	21%	23%	23%	22%	21%

Využití procesoru bylo ve všech testech podobné, stejně tak jako jeho průběh. Nejvyšší využití nastávalo v bezprostřední chvíli po spuštění aplikace a poté znovu v moment, kdy v aplikaci začala být vykreslována grafika.

Tabulka 6.2: Využití RAM

	Test 1	Test 2	Test 3	Test 4	Test 5
RAM průměr	214 MB	217 MB	211 MB	216 MB	214 MB
RAM minimum	85 MB	84 MB	83 MB	85 MB	84 MB
RAM maximum	289 MB	295 MB	290 MB	293 MB	292 MB

Využití paměti RAM bylo také podobné ve všech testech a průběh kopíroval průběh využití procesoru.

Tabulka 6.3: Spotřeba dat

	Test 1	Test 2	Test 3	Test 4	Test 5
Data spotřeba	27,5 MB	29,3 MB	26,9 MB	27,1 MB	27,9 MB

Spotřeba dat byla podobná ve všech testech, kromě testu 2, během kterého byla často kamera oddálena a vykreslována tedy byla větší část terénu. Z tohoto důvodu aplikace stahovala více dat, které obsahují texturu terénu.

Tabulka 6.4: Spotřeba baterie

	Test 1	Test 2	Test 3	Test 4	Test 5
Baterie spotřeba	5%	6%	5%	5%	5%

Spotřeba baterie byla ve všech testech podobná, tedy 5 % baterie za 10 minut.

Je také nutno podotknout, že časový rámec od doby, kdy se začne vykreslovat terén do doby, kdy je zcela vykreslen a obsahuje texturu v maximálním rozlišení, je průměrně 10 vteřin. Je to ovlivněno rychlostí, jakou je knihovna OSG Earth schopna pracovat s předanými daty a vykreslovat je.

6.3 Uživatelské testování

Během uživatelského testování byla aplikace poskytnuta dvěma uživatelům, kteří po otestování odpovídali na následující otázky:

- Líbí se Vám aplikace?
- V jaké situaci byste ji použil?
- Je spotřeba baterie akceptovatelná?
- Je spotřeba dat akceptovatelná?

- Je ovládání aplikace intuitivní?
- Co byste na aplikaci změnil?

Uživatel Vojtěch

- Aplikace vypadá dobře, ale líbil by se mi větší detail krajiny.
- Dovedu si představit situace, kdy tuto aplikaci použiji místo mapy pro hezčí vizualizaci při pohledu z vrcholů.
- Spotřeba baterie je pravděpodobně standardní pro aplikace tohoto druhu, ve kterých je hodně vykreslování grafiky.
- Spotřeba dat je v rozumných mezích v souladu s tím, že aplikace nemá vysoký detail textur a terénu, takže se jedná o určitý kompromis.
- Aplikaci zvládne ovládat každý, kdo měl někdy zkušenost s ovládáním chytrého telefonu.
- Kromě již zmíněného mě nenapadá nic, co bych změnil.

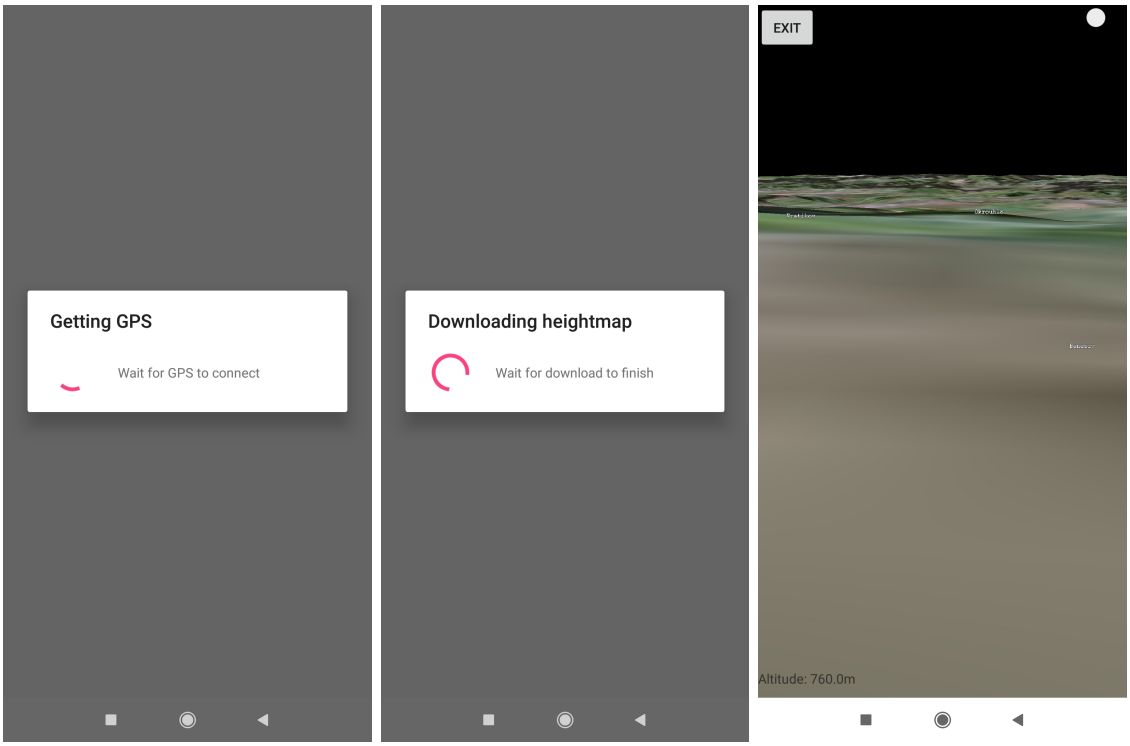
Uživatel Marek

- Aplikace se mi líbí, ale má trošku horší texturu terénu.
- Aplikaci bych použil při cestování, jako náhradu kompasu.
- Spotřeba baterie je vzhledem k povaze aplikace rozumná.
- Spotřeba dat je také rozumná.
- Ovládání aplikace je jednoduché a intuitivní.
- Do aplikace bych přidal možnost stahování offline map a plynulejší přechod při obracení telefonu.

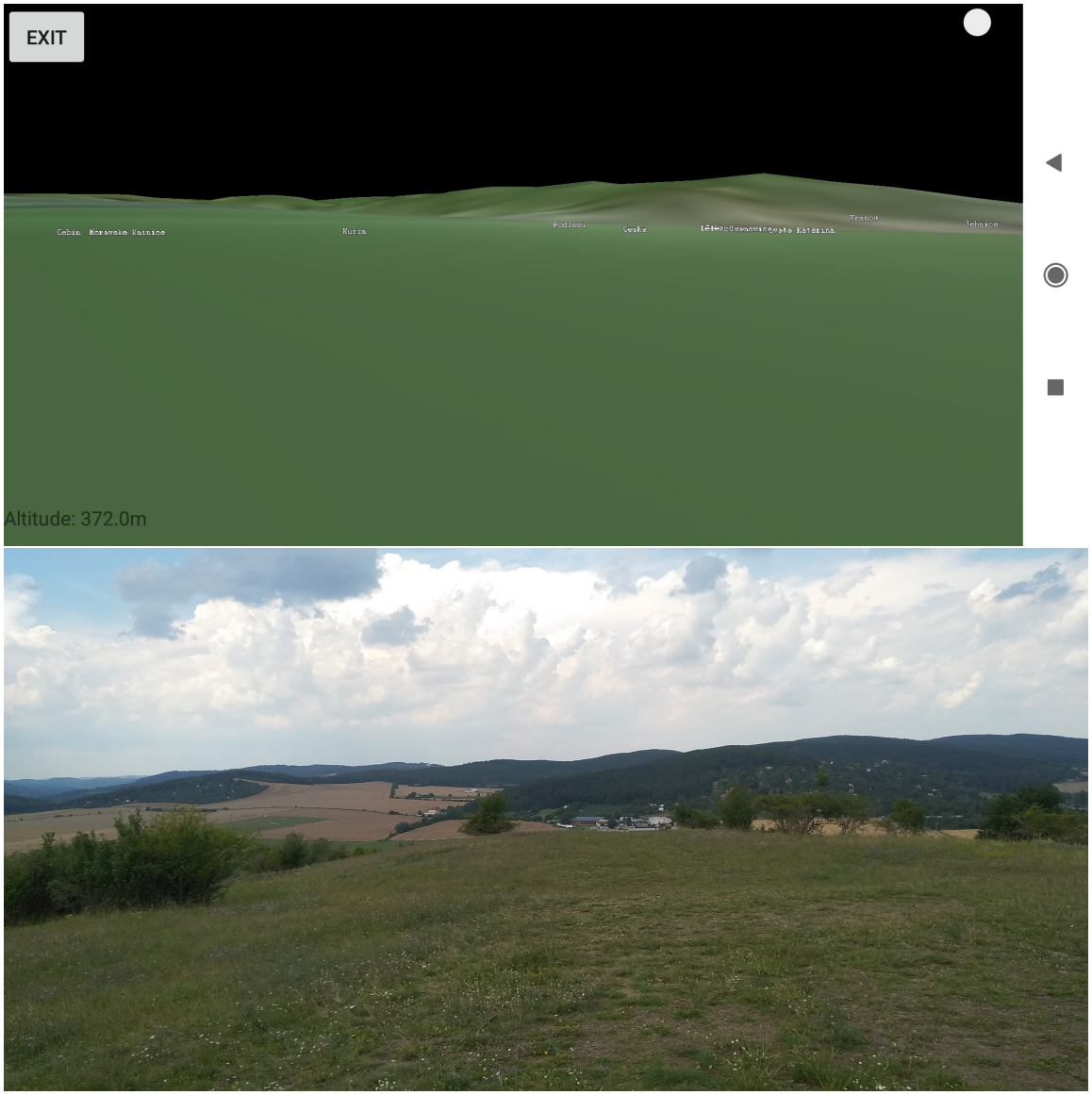
6.4 Výsledná aplikace

Výsledná aplikace je schopná určit polohu zařízení, na základě této polohy vytvořit dva dotazy. První na Overpass API, ze kterého získá názvy měst a vesnic v okolí a druhý na OpenTopography API, ze kterého získá výškovou mapu okolí. Tyto data poté vykreslí na obrazovku. Ovládání aplikace funguje pomocí pohybových senzorů a doteků na obrazovce. Průběh aplikace je zobrazen na obrázku 6.1.

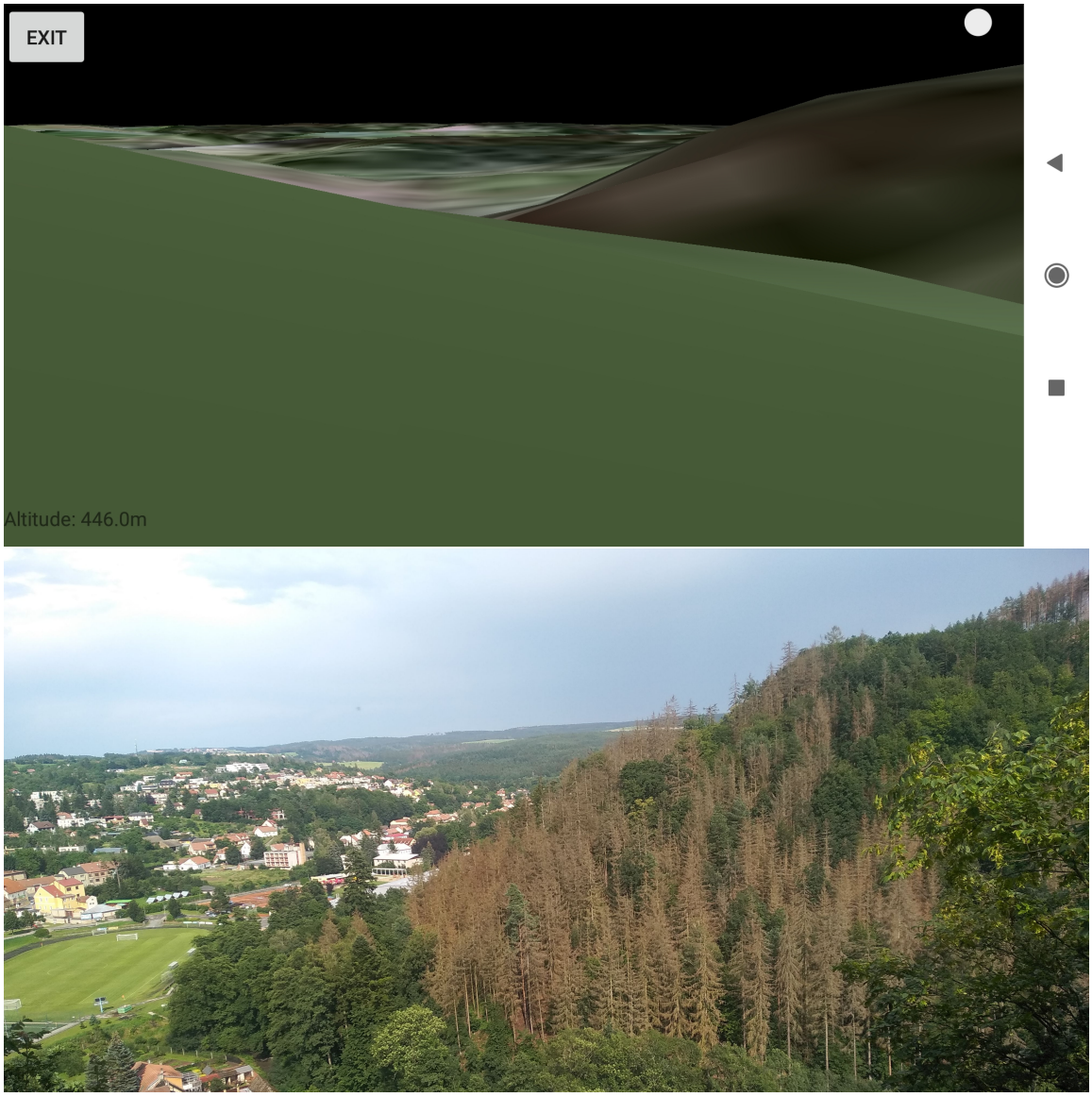
Ukázka a porovnání terénu z Medláneckých kopců je na obrázku 6.2. Ukázka a porovnání terénu z vyhlídky pod Boskovickým hradem je na obrázku 6.3. V případě, kdy uživatel oddálí kameru, aplikace se jeví jako mapa. Toto je znázorněno na obrázku 6.4.



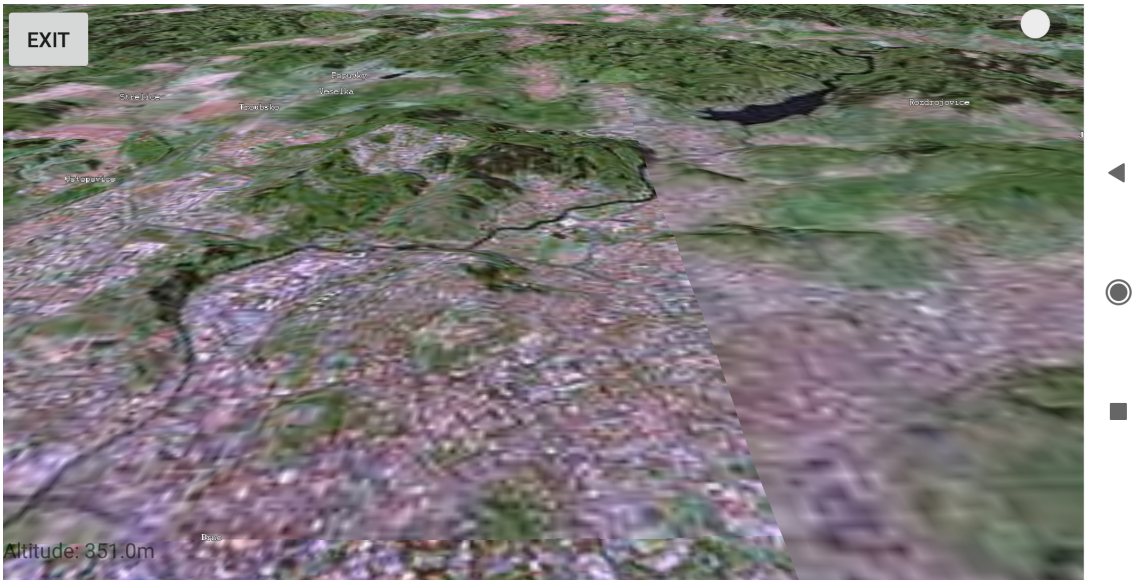
Obrázek 6.1: Průběh aplikace



Obrázek 6.2: Ukázka aplikace vyfocena z Medláneckých kopců.



Obrázek 6.3: Ukázka aplikace vyfocena z vyhlídky pod Boskovickým hradem.



Obrázek 6.4: Ukázka aplikace s oddálenou kamerou.

Kapitola 7

Závěr

Cílem této práce bylo popsat problematiku vývoje mobilních aplikací pro zobrazení digitálních modelů terénu a navrhnout aplikaci, která bude vykreslovat model členitosti terénu a na příslušné pozici zobrazovat názvy obcí a měst. Po dokončení návrhu byla tato aplikace implementována a následně testována z hlediska výkonu i ostatními uživateli. Výsledná aplikace je funkčním zpracováním jejího návrhu. K její implementaci byl použit jazyk Java a jazyk C++ při práci s knihovnou OSG Earth.

Informace pro úspěšný návrh a implementaci aplikace byly získány z konzultací s vedoucím bakalářské práce, z dokumentace pro vývoj aplikací na OS Android, dokumentace knihovny OSG Earth a knih, které rozebírají problematiku digitálních modelů terénu a zdrojů jejich dat.

Při vývoji byl kladen důraz na splnění zadání. Z tohoto důvodu byl přehodnocen původní koncept návrhu, který byl přetvořen do podoby, ze které vznikla výsledná aplikace.

Aplikace splňuje zadání a funguje stabilně. Vývoj v knihovně OSG Earth nabízí velké množství případných vylepšení nebo přidání funkcionality. Mezi ně patří například přidání simulace slunce, které osvětluje terén v závislosti na své reálné pozici. V případě získání kvalitních výškových dat pro určité lokality by mohly být použity pro vykreslení velmi přesného terénu, který může obsahovat i modely budov. Do aplikace by také mohla být přidána nabídka, které geografické informace má aplikace vypsat. Nemuselo by se tedy jednat jen o názvy měst a vesnic, ale například vrcholy kopců a hor nebo názvy řek. Dalším vylepšením by mohla být možnost stahování map a používání aplikace i bez připojení k internetu.

Literatura

[1] *About the platform* [online]. [cit. 2020-01-17]. Dostupné z: https://developer.android.com/about.

[2] *Android studio* [online]. [cit. 2020-01-10]. Dostupné z: https://developer.android.com/studio.

[3] *API Reference* [online]. [cit. 2020-01-11]. Dostupné z: https://developer.android.com/reference.

[4] *NGA: DoD World Geodetic System 1984* [online]. [cit. 2020-04-17]. Dostupné z: https://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html.

[5] *Open Topography* [online]. [cit. 2020-04-17]. Dostupné z: https://opentopography.org/.

[6] *OpenGL ES* [online]. [cit. 2020-01-10]. Dostupné z: https://www.khronos.org/opengles/.

[7] *OpenSceneGraph* [online]. [cit. 2020-04-17]. Dostupné z: http://www.openscenegraph.org/.

[8] *OpenSceneGraph Earth* [online]. [cit. 2020-04-17]. Dostupné z: http://osgearth.org/.

[9] *Overpass API* [online]. [cit. 2020-04-17]. Dostupné z: https://wiki.openstreetmap.org/wiki/Overpass_API.

[10] *Platform Architecture* [online]. [cit. 2020-01-17]. Dostupné z: https://developer.android.com/guide/platform/index.html.

[11] *Používání mobilního telefonu a internetu na mobilním telefonu* [online]. [cit. 2020-01-17]. Dostupné z: https://www.czso.cz/csu/czso/5-komunikace-na-internetu.

[12] *SRTM HGT (Shuttle Radar Topography Mission Height) Reader* [online]. [cit. 2020-01-17]. Dostupné z: https://docs.safe.com/fme/html/FME_Desktop_Documentation/FME_ReadersWriters/srtmhgt/srtmhgt.htm.

[13] *StatCounter* [online]. [cit. 2019-12-31]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[14] *Understand the Activity Lifecycle* [online]. [cit. 2020-01-17]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle.

[15] *Geoscientific Mineral Resources* [online]. 2019 [cit. 2019-12-10]. Dostupné z: https://geoscientific.net/digitalelevation.html.

[16] Eikenes, J. O. *Navimation Research* [online]. [cit. 2019-12-10]. Dostupné z: http://www.navimationresearch.net/tag/heightmap/.

[17] FARR, P. A. R. E. C. e. a. *The Shuttle Radar Topography Mission. Reviews of Geophysics* [online]. 2007 [cit. 2019-12-10]. Dostupné z: http://doi.wiley.com/10.1029/2005RG000183.

[18] HIRT, C. *Digital Terrain Models* [online]. Springer International Publishing, 2014. Dostupné z: https://link.springer.com/referenceworkentry/10.1007%2F978-3-319-02370-0_31-1. ISBN 978-3-319-02370-0.

[19] LI, Z., ZHU, Q. a GOLD, C. *Digital terrain modeling: principles and methodology.* New York: CRC Press, 2005. ISBN 0-4153-2462-9.

[20] Podobnikar, T. *Methods for visual quality assessment of a digital terrain model* [online]. Institut Veolia, únor 2009 [cit. 2019-11-24]. Dostupné z: https://journals.openedition.org/sapiens/738.

[21] TACHIKAWA, M. H. M. K. a. A. I. *Characteristics of ASTER GDEM version 2* [online]. 2011 [cit. 2019-12-10]. Dostupné z: http://ieeexplore.ieee.org/document/6050017/.