VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

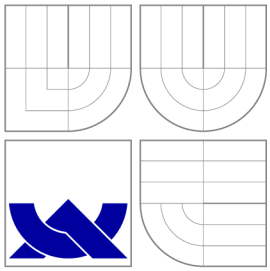
NAVIGACE PRO IOS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

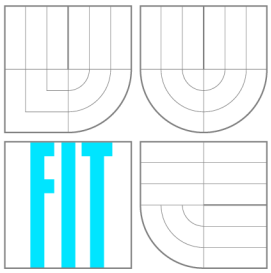
AUTOR PRÁCE
AUTHOR

MARTIN SLADEČEK

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NAVIGACE PRO IOS

NAVIGATION FOR IOS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN SLADEČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JIŘÍ ŠEVCOVIC

BRNO 2013

Abstrakt

Cílem práce je návrh a implementace softwaru mobilní navigace pro platformu iOS, respektive aplikace pro zařízení iPhone a iPad. Jednoduchá navigace umožňuje uživateli zjistit aktuální polohu, zobrazit mapu ve 2D i 3D módu a navigovat k zadanému cíli. Cílová zařízení disponují GPS modulem k zjištění aktuální polohy. Mapové podklady jsou získávány z projektu OpenStreetMap a uchování geografických dat zajišťuje databázový systém SpatiaLite.

Abstract

The aim of this thesis is a design and implementation of software mobile navigation for platform iOS, or more precisely an application for devices iPhone and iPad. Simple navigation enables the user to discover the current position, display map both in 2D and 3D mode and navigate to given destination. Target devices handle with GPS module to discover the current position. Maps sources are gained from the project OpenStreetMap and database system SpatiaLite provides preservation of geographical dates.

Klíčová slova

Navigace, iOS, iPhone, iPad, OpenStreetMap, SpatiaLite, GIS, GPS, geografický souřadný systém, WGS84, Open GL ES

Keywords

Navigation, iOS, iPhone, iPad, OpenStreetMap, SpatiaLite, GIS, GPS, geographic coordinate system, WGS84, Open GL ES

Citace

Martin Sladeček: Navigace pro iOS, bakalářská práce, Brno, FIT VUT v Brně, 2013

Navigace pro iOS

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Ševcovice. Uvedl jsem veškeré literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Sladeček
15. května 2013

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu Ing. Jiřímu Ševcovicovi za rady a podněty při vytváření práce. Dále bych chtěl poděkovat svým přátelům za nápady a rady k vylepšení. V neposlední řadě pak své rodině za skvělé podmínky k práci a shovívavost.

Obsah

Kapitola 1

Úvod

Navigace je proces určení polohy a nalezení cesty podle požadovaných kritérií. Jedná se o jeden z nejstarších oborů v lidské historii, který úzce souvisí s migrací lidí. Za první způsoby navigace můžeme považovat osobní dorozumívání či orientaci pomocí styčných bodů a map. Pravděpodobně největšího rozmachu geografie a kartografie bylo dosaženo během obchodování a objevování nových zemí pomocí mořeplavby. Díky pokroku v astronomii (určování polohy na základě konstelace slunce, měsíce a hvězd) či objevení kompasu a sextantu dochází k zpřesnění určování polohy. V moderní době, s příchodem elektrotechniky, vznikají radionavigační zařízení, která dokáží určit polohu automaticky na základě vzdáleností od radio-majáků (NDB). Později pak pozemní (LORAN) či družicové navigační systémy (GPS, GLONASS, Galileo) založené na podobném principu. I když se prostředky s příchodem moderní techniky změnily, cíle navigace zůstávají stejné.

V dnešní době je pojmem navigace nejčastěji myšlen elektronický přístroj, který určuje aktuální polohu automaticky (co nejpřesněji v závislosti na hardwarové výbavě a zvolené technologii), a obsahuje software, který dokáže nalézt cestu a vést uživatele pomocí instrukcí z bodu A do bodu B (tzv. turn by turn navigace). Existuje řada druhů navigací, často specificky zaměřených, které se liší funkčností, přesností určování polohy či využitými technologiemi. Navigace je možné rozdělit na online a offline. První jmenované potřebují ke svému chodu internetové připojení, stahování mapových podkladů i vyhledávání trasy probíhá na vzdáleném serveru, kdežto offline navigace veškerá potřebná data obsahují, vykreslování i vyhledávání tras probíhá v rámci zařízení. Nevýhodou offline navigací je bezesporu jejich datová náročnost a nutnost provádět pravidelné aktualizace.

S vývojem hardwaru dochází čím dál častěji k integraci potřebných komponent navigace do chytrých telefonů i jiných mobilních zařízení. V dnešní době je již většina těchto přístrojů vybavena vestavěným GPS modulem. Rozšíření této technologie zapříčinilo vznik celé řady geografických aplikací. Existují velmi sofistikované offline auto-navigace zaměřující se na autodopravu, navigace pro pěší turistiku či cykloturistiku.

Cílem této bakalářské práce je návrh a implementace softwaru mobilní navigace pro platformu iOS, respektive aplikace pro zařízení iPhone a iPad. Jedná se o univerzální aplikaci, kterou lze využít na obou typech zařízení. Přístroje obsahují modul GPS, který zajišťuje zjišťování aktuální polohy. Mapové podklady jsou získány z projektu OpenStreetMap a uchování dat zajišťuje databázový systém SpatiaLite. Vzhled a funkčnost aplikace se snaží o maximální uživatelskou přívětivost a jednoduchost. Navigace umožňuje uživateli mimo jiné zjistit aktuální polohu, zobrazit mapu ve 2D i 3D módu a navigovat k zadanému cíli za pomoci směrových instrukcí. Grafické uživatelské rozhraní je testováno na reprezentativním vzorku potenciálních uživatelů a porovnáváno s dalšími podobnými aplikacemi.

Následující kapitola 2 se týká teoretických znalostí potřebných k vytvoření softwaru mobilní navigace. Je zde analyzována cílová platforma iOS, včetně popisu logických vrstev. Důraz je kladen na části, které lze využít při vývoji navigace. Další podkapitoly jsou věnovány modulu GPS (včetně teoretických znalostí z oblasti geografie), získávání geografických dat z projektu OpenStreetMap i jejich ukládání do databázového systému SpatiaLite. Následuje kapitola 3 popisující návrh aplikace. Je zde popsán návrh uživatelského rozhraní, struktura databáze či návrh diagramu tříd. Kapitola 4 popisuje nejzajímavější pasáže z implementace aplikace. Je zde přiblížen princip získávání aktuální polohy či vysvětlen princip vykreslování mapových podkladů. Kapitola 5 stanovuje metodiku testování a analyzuje výsledky. Závěrečná kapitola 6 pak shrnuje výsledky této práce a navrhuje případná rozšíření.

Kapitola 2

Analýza problému

Kapitola se zabývá analýzou nezbytných teoretických znalostí, které jsou důležité k vývoji softwaru mobilní navigace. První podkapitola se věnuje analýze cílové platformy iOS. Popisuje operační systém iOS, jeho strukturu a nástroje usnadňující vývoj. Zaměřuje se převážně na oblasti využitelné při vývoji navigace. Podkapitola 2.2 popisuje Core Location, součást operačního systému iOS, která umožňuje mimo jiné zjišťování aktuální polohy. Nejdůležitější součástí vrstvy Core Location je modul GPS, proto je zde popsán princip funkčnosti a potřebné znalosti z oblasti geografie a kartografie.

Nedílnou součástí každého GIS[1] je ukládání geografických dat. Pro tento účel je použita knihovna SpatiaLite, nadstavba databázového systému SQLite, která je popsána v podkapitole 2.3. K získání podkladových map byl využit projekt OpenStreetMap, který se zaměřuje na vytváření, editaci a uchovávání geografických dat z celého světa. Podkapitola 2.5 se věnuje analýze konkurenčních řešení.

2.1 Platforma iOS

iOS je operační systém od společnosti Apple určený pro mobilní zařízení [26]. Jedná se o systém UNIXového typu, který je odlehčenou variantou desktopového systému Mac OS X. Zpočátku byl vyvíjen pouze pro mobilní telefon iPhone, později však našel uplatnění i v dalších zařízeních společnosti jako je iPad, iPod Touch nebo nejnovější Apple TV [27].

Vývoj systému iOS je uzavřený a řízený společností Apple. Tento systém nelze licencovat na zařízení dalších společností. Systém je napojen na celý ekosystém společnosti (iCloud, iTunes, Game Center atd.). Do zařízení lze aplikace nahrávat pouze přes elektronický obchod AppStore. Vývojáři musí vlastnit placený vývojářský účet, popřípadě být zařazeni do studentského programu.

Primárním vývojovým jazykem je Objective-C [9]. Jedná se o objektově orientovaný programovací jazyk (OOP) implementovaný jako rozšíření jazyka ANSI C (díky tomu lze využít i jazyky C/C++). Objektové prvky jsou převzaty z jazyka Smalltalk. Existuje řada rozdílů od jiných představitelů OOP, především v syntaxi (pojmenovávání metod, volání metod), dále pak například datový typ id (reference na jakýkoliv objekt), přetěžování metod, virtuální metody či rozdílná interpretace dědičnosti atd. Pěkné shrnutí rozdílů oproti jazyku C++ je zpracováno v dokumentu [3]. Zdrojové soubory mají příponu `.m`, popřípadě `.mm` (pokud obsahují C++ kód), hlavičkové soubory pak vždy příponu `.h`.

[1] Geografický informační systém

2.1.1 Struktura iOS

Na obrázku 2.1 je znázorněno rozdělení architektury systémů Mac OS X a iOS na logické vrstvy [24]. Nejnižší vrstva `Core OS` zajišťuje základní chod systému. Obsahuje OS X jádro (upravená verze UNIXového jádra), prostředky pro správu paměti a výkonu, BSD sokety, certifikáty a souborový systém. Vývojář nepracuje s těmito prostředky přímo, jsou mu zprostředkovány pomocí vyšších vrstev. Souborový systém není rovněž vývojáři přístupný, aplikace může pracovat jen s přiděleným prostorem, tzv. sandbox, ve kterém lze ukládat uživatelská data.

Obrázek 2.1: Architektura systému Mac OS X a iOS. Převzato z [24]

Vrstva `Core services` zprostředkuje služby nejnižší vrstvy jako je přístup k souborům, práce se sítí, uživatelské nastavení, knihovny pro zpracování XML dokumentů, databázová vrstva `SQLite` či `Core Location`. Poslední jmenované položky jsou zvláště důležité pro tuto práci. Databáze `SQLite`, respektive její nástavba SpatiaLite, je využita k ukládání veškerých perzistentních dat v aplikaci a podrobně je popsána v kapitole 2.3. Služba `Core Location` zprostředkuje komunikaci mezi aplikací a hardwarem nutným k zjišťování aktuální polohy. Více o problematice v kapitole 2.2.

`Media` vrstva obsahuje prvky využívané k podpoře multimediálního obsahu. Knihovnu Quartz lze využít k tvorbě vektorové 2D grafiky či renderování obrázků. OpenGL ES [18] zajišťuje hardwarově akcelerované vykreslování 2D/3D objektů [17]. Knihovny Quartz a OpenGL ES jsou využity k vykreslování mapových podkladů, podrobněji v kapitole 4.3. Core Animation umožňuje pokročilou práci s animacemi. Práci s audio/video obsahem zajišťují knihovny Media Player framework, AV Foundation či Core Audio framework a Core Media. Jsou podporovány nejrozšířenější formáty obrázků (JPG, PNG, TIFF), videí (kodeky H.264 a MPEG-4) i zvuků (AAC, ALAC, IMA4).

Jak je z obrázku 2.1 patrné, desktopový sytém Mac OS X a mobilní systém iOS sdílí některé části systému. V případě mobilního systému samozřejmě částečně odlehčené. Největší rozdíl mezi desktopovým a mobilním systémem je v nejvyšší vrstvě. Vrstva `Cocoa Touch` vychází z Mac OS X Cocoa API a obsahuje nejdůležitější framoworky. Součástí jsou knihovny pro vzhled a ovládání aplikace v dotykovém mobilním prostředí. Podporuje uživatelská multidotyková gesta, zajišťuje lokalizaci, spravuje notifikace atd.

2.1.2 iOS SDK

iOS SDK [10] je sada nástrojů od společnosti Apple usnadňující vývoj aplikací pro iOS. Součástí je mimo jiné vývojové prostředí Xcode, iOS simulátor, Interface Builder a Instruments. Tyto nástroje jsou spustitelné pouze na počítačích s operačním systémem Mac OS X.

Xcode je vývojové prostředí, které funguje jako propracovaný editor kódu. Mimo jiné obsahuje integrovaný verzovací systém, debugger, správce projektů či offline dokumentaci. Slouží jako prostředek k práci s ostatními nástroji popsanými níže.

iOS Simulátor slouží k testování aplikací. Tento nástroj emuluje zařízení iPhone či iPad a dovoluje vývojáři testovat aplikace přímo v počítači. Zvládá základní ovládací gesta, natočení displeje a lze uměle simulovat nečekané situace. Nejedná se však o plnohodnotnou kopii a má určité limity. Reálná zařízení jsou postavena na ARM architektuře, kdežto simulátor překládá aplikace do instrukční sady x86. Nelze simulovat více-dotyková gesta a některé prvky zde logicky zcela chybí, např. telefonování, zasílání SMS zpráv, fotoaparát, mikrofon atd.

Interface Builder je nástroj, který dovoluje metodou drag & drop připravit uživatelské rozhraní a napojit komponenty na funkční logiku (nastavit outlety a akce).

Instruments je sada nástrojů usnadňující ladění aplikace. Dovolují vývojáři sledovat chod své aplikace a analyzovat její výpočetní nároky. Mimo jiné obsahuje nástroje pro diagnostiku spotřeby energie, sledování internetové konektivity, analýzu grafického výkonu (OpenGL ES) či správu paměti.

2.2 Core Location

`Core Location` je část iOS frameworku, která zajišťuje komunikaci mezi aplikací a hardwarem nutným k zjištění aktuální polohy [10]. Umožňuje vývojáři pohodlný přístup ke geolokačním datům (zeměpisná délka, zeměpisná šířka a nadmořská výška) s přesností závisející na zvolené úrovni [1]. Pokud zařízení obsahuje vestavěný magnetometr, zpřístupní i údaje o aktuálním nasměrování přístroje.

Zjištění aktuální polohy je velmi náročná operace a spotřebuje značnou část energie. Existuje i řada geolokačních aplikací, které nepotřebují znát v každý okamžik polohu s přesností na metry. Řešením je využití více technik k zjištění aktuální polohy. `Core Location` pracuje se třemi principy [1]:

- Nejméně přesná úroveň využívá k zjištění polohy mobilní vysílače. Dokáže za velmi krátkou dobu určit polohu s přesností 2-3 km v závislosti na hustotě mobilní sítě.
- Mnohem více přesnější metoda využívá k zjištění polohy WiFi sítě. Dokáže určit pozici s přesností do 100 m v závislosti na počtu antén v dosahu.
- Nejpřesnější úroveň využívá GPS modul, který dokáže určit polohu s přesností do 10 m. Pochopení principů GPS je pro tuto práci nesmírně důležité, téma podrobněji analyzuji níže v podkapitole 2.2.1.

Programátor je od výše zmíněné problematiky zcela oproštěn. Parametrem si zvolí požadovanou přesnost určování polohy a zařízení se snaží aproximací všech uvedených metod dodat nejvhodnější hodnoty (viz kapitola 4.2). Tato abstrakce umožňuje tvořit aplikace nezávislé na konkrétním geolokačním modulu a dovoluje výrobcům zařízení instalovat nový hardware (v budoucnu například navigační systém Galileo), který bude zpětně kompatibilní se staršími aplikacemi.

2.2.1 GPS systém

GPS[2] je družicový polohový systém jehož hlavní úlohou je určování polohy na Zemi [25]. Systém dokáže určit polohu s přesností do 10 m. Původně vojenský systém provozovaný Ministerstvem obrany Spojených státu amerických byl v 80. letech uvolněn veřejnosti a ihned začala vznikat první zařízení využívající tuto technologii. GPS systém je tvořen třemi logicky oddělenými segmenty:

- Kosmický segment tvoří družice na šesti kruhových oběžných drahách se sklonem 55° k rovině rovníku. Družice jsou vzdáleny přibližně 20 200 km od povrchu Země a pohybují se rychlostí 11 300 km/h. Během dne družice uskuteční 2 oběhy kolem Země. Každá ze 6 drah má 5 pozic k umístění družic, přičemž 1 z nich je vždy náhradní. Aby byl systém využitelný kdekoliv na Zemi, je potřeba 24 funkčních družic.
- Řídící segment se stará o provoz satelitů. Hlavním úkolem je sledovat dráhy družic a stav jejich atomových hodin. Monitorovací stanice jsou umístěny po celém světě.
- Uživatelský segment se skládá z pasivních GPS přijímačů (v našem případě iPhone či iPad), které jsou schopny přijímat a dekódovat signál z družic.

Princip určování polohy je založen na trilateraci. Satelity musí velmi přesně určit svou aktuální polohu (vypočítatelnou ze své trajektorie a aktuálního času). Pasivní přijímač zjišťuje aktuální polohu na základě vzdáleností od okolních družic. Pokud známe vzdálenosti od minimálně tří družic (větší počet zvyšuje přesnost a rychlost), jsme schopni zjistit souřadnice v 3D pravoúhlém souřadném systému, které můžeme interpretovat uživateli (nejčastěji pomocí standardu WGS84, viz podkapitola 2.2.3).

2.2.2 Geografické souřadné systémy

Souřadný systém je nástroj k vyjádření polohy bodu v nějakém prostoru [8]. V případě geografického souřadného systému prostor planety Země či její části. Systém musí splňovat následující požadavky. Definice polohy je jednoznačná, kvantifikovatelná a musí být definována metrika k měření vzdálenosti mezi objekty. V případě pravoúhlého prostoru lze k měření vzdáleností využít Euklidovskou metriku [8], kdy vzdálenost mezi body $A = (A_1, A_2, ..., A_N)$ a $B = (B_1, B_2, ..., B_N)$ je dána vztahem:

$$d_{(A,B)} = \sqrt{\sum_{i=1}^{N} (A_i - B_i)^2} \tag{2.1}$$

Jelikož Země je nepravidelný objekt a přesné popsání povrchu s následným určením polohy by bylo technicky příliš náročné, existuje abstrakce povrchu v podobě koule nebo častěji elipsoidu. Elipsoid je prostorové těleso dáno následující rovnicí:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} \leq 1 \tag{2.2}$$

Vztažných elipsoidů popisující povrch Země existuje celá řada. Rozlišujeme systémy globální (WGS84, UTM) a lokální (S-JTSK). Globální se snaží definovat celý geografický prostor Země, proto nejsou vhodné pro přesná zaměřování v dané oblasti (katastr nemovitostí). Lokální vznikly velmi specifickou transformací územně platného náhradního elipsoidu

[2] Global Positioning System

na plochu [8]. Každá země či oblast má vlastní standard, který se nejlépe hodí pro konkretní povrch. Tato práce se nadále zabývá pouze standardem WGS84 (viz kapitola 2.2.3). Důvodem je globální využití, používá se v systému GPS i v mapových podkladech OpenStreetMap (viz kapitola 2.4).

2.2.3 Geodetický systém šířka-délka (WGS84)

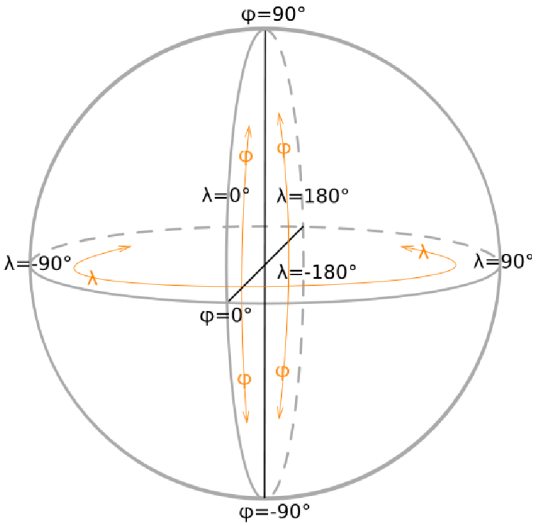
Standard WGS84[3] [11] definuje souřadný systém pro určení polohy bodu na Zemi, známější pod spojením zeměpisná délka/šířka. Vznikl v Americe v roce 1984 a je celosvětově uznávaným standardem. Systém WGS84 má konstanty vztažného elipsoidu stanoveny následovně:

$$a = 6378137\,m$$

$$b = 6356752,3142\,m$$

$$c = 6399593,6258\,m$$

Zeměpisná šířka je úhel, který svírá bod na Zemi (resp. na vztažném elipsoidu) s rovinou rovníku. Zeměpisná délka je úhel, který svírá rovina místního poledníku procházejícího daným bodem a rovinou Greenwichského (nultého) poledníku. Jak lze vidět z obrázku 2.2, zeměpisná šířka má hodnoty v rozsahu $\langle -90, 90 \rangle$, kde na sever od rovníku jsou kladné hodnoty, a zeměpisná délka v intervalu $\langle -180, 180 \rangle$, kde na východ od nultého poledníku jsou kladné hodnoty.



Obrázek 2.2: Země elipsoid. Převzato z [4]

[3] World Geodetic System 1984

2.3 SpatiaLite

SpatiaLite [22] je databázový systém podporující standard SQL92 a OGC-SFS[4] [12]. Jedná se o nadstavbu systému SQLite, kterou díky implementaci standardu OGC-SFS rozšiřuje o možnost ukládání a práce s prostorovými (anglicky spatial) daty. Knihovna je implementována v jazyce C. Jedná se o Open source projekt, o alternativu k databázovému systému PostGIS (rozšiřující relační klient/server databázi PostgreSQL o standard OGC-SFS). Celý databázový systém se skládá ze tří částí:

- **PROJ4** (verze 4.8.0) knihovna [16] pro konverzi mezi geografickými projekcemi.
- **GEOS** (verze 3.3.8) knihovna [6] implementující standard OGC-SFS.
- **libSpatiaLite** (verze 2.4.0) knihovna zajišťuje kompatibilitu mezi systémem SQLite a zmíněnými knihovnami, doplňuje či upravuje specifické funkce pro prostředí SQLite.

2.3.1 GEOS

Knihovna GEOS[5] je napsána v jazyce C++ (poskytuje i C API) a implementuje specifikaci OGC-SFS. Jednou z nejdůležitějších součástí je datový typ Geometry, který zajišťuje ukládání geografických dat. V následující tabulce 2.1 je uveden obecný formát interního uložení dat [20].

Offset (byte)	Obsah	Popis
0	START	Začátek datového typu Geometry, vždy 0x00
1	ENDIAN	0x00 Big endian / 0x01 Little endia
2-5	SRID	SRID pro dané souřadnice (32-bitový integer)
6-13	MIN X	Minimální hodnota X (double)
14-21	MIN Y	Minimální hodnota Y (double)
22-29	MAX X	Maximální hodnota X (double)
30-37	MAX Y	Maximální hodnota Y (double)
38	HEADER'S END	Konec hlavičky, vždy 0x7C
39-42	CLASS TYPE	Identifikátor konkrétní třídy
43-...	CLASS SPECIFIC	Délka a obsah záleží na zvolené třídě
LAST	END	Konec datového typu Geometry, vždy 0xFE

Tabulka 2.1: Interní formát ukládání geografických dat

Každý datový typ Geometry obsahuje hlavičku a tělo. Hlavička obsahuje informaci o způsobu uložení (little či big endian), okrajové souřadnice vymezující maximální prostor a důležitý údaj SRID[6] [4], který určuje v jakém geografickém souřadném systému jsou souřadnice uloženy. Hodnoty jsou přesně definované mezinárodní autoritou EPSG[7]. V případě standardu WGS84 (kapitola 2.2.3) hodnota 4326. V těle je uložen typ třídy a vlastní data. V případě potřeby jsou data přepočítána knihovnou PROJ4 do zadaného souřadného systému.

[4] Open Geospatial Consortium - Simple Feature Specification
[5] Geometry Engine - Open Source
[6] Spatial Reference Identifier
[7] European Petroleum Survey Group

Existuje celá řada tříd odvozených od datového typu Geometry [5]. Mezi nejdůležitější patří POINT, LINESTRING a POLYGON. POINT obsahuje souřadnice X a Y uloženého bodu (64-bitový double). LINESTRING obsahuje počet a seznam bodů (třída POINT). Využívá se pro popis lineárních cest (např. silnic). POLYGON obsahuje počet a seznam bodů, přičemž první a poslední bod jsou totožné. Jedná se o uzavřenou oblast a typ se používá k definici prostoru. Databázový systém poskytuje i řadu dalších datových typů a to i v trojrozměrném prostoru, kterým se však tato práce nezabývá.

K zápisu či přenosu geografických dat lze využít i formát Well-Known Text (WKT) nebo Well-Known Binary (WKB). Pomocí formátu WKT lze data zadávat a přenášet jako ASCII řetězec s přesně definovanou syntaxí, např. LINESTRING(100.0 200.0, 201.5 102.5, 1234.56 123.89). Oproti tomu WKB je formát vhodný k výměně dat pomocí proudu bytů. Obsahuje 1 unsigned integer pro určení little či big endian, 4 unsigned integers k definici WKB typu a vlastní data. Např. POINT(1 1) by byl reprezentován sekvencí 21 bytů (každý znázorněn 2 hexadecimálními znaky) 01 01000000 000000000000F03F 000000000000F03F.

Další nedílnou součástí knihovny jsou datové struktury v jazyce C [19] reprezentující Geometry (např. gaiaPointStruct, gaiaLinestringStruct, gaiaRingStruct atd.), funkce pro manipulaci s těmito objekty (gaiaAllocLinestring, gaiaCloneLinestring, gaiaFreeLinestring atd.) či funkce pro vytváření struktur z formátů WKT a WKB (gaiaParseWkt, gaiaOutWkt, gaiaFromWkb, gaiaToWkb atd.).

2.3.2 libSpatiaLite

Jak již bylo zmíněno dříve, libSpatiaLite je knihovna napsaná v jazyce C a zpřístupňuje knihovnu GEOS a PROJ4 pro použití v databázovém systému SQLite, resp. jazyce SQL. Systém SQLite umožňuje definovat vlastní funkce a vytvářet tak rozšíření. Této vlastnosti knihovna libSpatiaLite využívá.

Databázový systém se mimo jiné musí starat o konzistenci ukládaných geografických dat. Metadata jsou uložena v tabulce `spatial_ref_sys` a `geometry_columns`. Tabulka `spatial_ref_sys`, viz tabulka 2.2, obsahuje informace k jednotlivým SRID, resp. obsahuje PROJ4 řetězec potřebný k převodu mezi jednotlivými souřadnými systémy.

SRID	auth_name	auth_srid	ref_sys_name	proj4text
4326	epsg	4326	WGS 84	+proj=longlat +datum=WGS84 +no_defs

Tabulka 2.2: Ukázka metadat z tabulky spatial_ref_sys

V tabulce `geometry_columns`, viz tabulka 2.3, jsou uloženy všechny sloupce datového typu Geometry, ke kterým je třeba ukládat informace o typu, používaném SRID a o povoleném indexování.

f_table_name	f_geometry_column	geometry_type	srid	spatial_index_enabled
roads	geometry	2	4326	1

Tabulka 2.3: Ukázka metadat z tabulky geometry_columns

Indexování rozsáhlých databází je velmi důležité, v případě geografických databází obzvláště. Defaultní indexování používané v databázi SQLite implementované jako BTree (Binary Tree) je nevyhovující pro datový typ Geometry. Pro indexování geografických dat se ukázala nejvhodnější implementace pomocí R*Tree (Rectangle Tree). Každý prvek je reprezentován obdélníkem určeným hraničními body (souřadnice jsou uloženy u každého typu Geometry, viz formát tabulka 2.1). R*Tree je v SQLite (dostupný od verze 3.6.0) reprezentován pomocí virtuálních tabulek. Tabulka `IndexName_node` obsahuje binární data odpovídající stromovému uzlu, tabulka `IndexName_parent` reprezentuje stromovou hierarchii a tabulka `IndexName_rowid` propojuje uzly stromu s čísly řádků původních dat. Vše probíhá transparentně uživateli. Tabulky se aktualizují s každou operací insert, update, delete.

S knihovnou SpatiaLite se výhradně pracuje pomocí jazyka SQL, proto nezbytnou znalostí jsou rozšiřující funkce (matematické, pro práci s Geometry objekty, na měření vzdálenosti bodů v prostoru, na měření délky cest, pro práci s R*Tree atd). Výběr funkcí se syntaxí a popisem je v tabulce 2.4. Kompletní seznam lze nalézt na stránce [21].

SQL funkce	Syntaxe	Popis
Degrees	Degrees(x Double): Double	Převede hodnotu v radiánech na stupně
MbrMinX	MbrMinX(geom Geometry): Double	Získá minimální X souřadnici
GeomFromText	GeomFromText(wkt String [, SRID Integer]): Geometry	Vytvoří Geometry z WKT, lze specifikovat SRID
AsText	AsText(geom Geometry): String	Vytvoří WKT z Geometry
X	X(pt Point): Double	Získá X z Geometry typu Point
StartPoint	StartPoint(c Curve): Point	Získá počáteční bod z křivky
NumPoints	NumPoints(line LineString): Integer	Získá celkový počet bodů v Geometry typu LineString.
Within	Within(geom1 Geometry, geom2 Geometry): Integer	Vrací 1 pokud geom1 je kompletně uvnitř geom2, opak vrací 0
Distance	Distance(geom1 Geometry, geom2 Geometry): Double	Vrací vzdálenost mezi dvěma Geometry objekty
AddGeometryColumn	AddGeometryColumn(table String, column String, srid Integer, geom_type String, dimension String [, not_null Integer]): Integer	Vytváří sloupec column v tabulce table s nastaveným SRID, typem objektu a dimenzí, možno nastavit tabulce not null příznak; vrací 1 pokud úspěch, opak vrací 0

Tabulka 2.4: Výběr SQL funkcí v systému SpatiaLite

V následujícím pseudokódu 2.1 je ukázka reálného SQL dotazu na databázový systém SpatiaLite. Dotaz získává nejbližší silnici z tabulky `Roads` pro zadaný bod (počáteční a koncové body). Využívá R*Tree indexaci pro rychlejší selekci hledané oblasti.

```
SELECT Node_From, Node_To, X(StartPoint(Geometry)), Y(StartPoint
    (Geometry)), X(EndPoint(Geometry)), Y(EndPoint(Geometry)),
    Distance(Geometry,MakePoint(%f, %f)) AS distance
FROM Roads
WHERE id IN
(SELECT pkid AS id FROM idx_roads_geometry WHERE xmin > %f and
    xmax < %f and ymin > %f and ymax < %f)
ORDER BY distance
LIMIT 1
```

Algoritmus 2.1: Ukázka dotazu získávající nejbližší silnici pro zadaný bod

2.3.3 VirtualNetwork modul

Knihovna SpatiaLite je složená z logických modulů. Jedním z nich je VirtualNetwork modul. Zajišťuje podporu vyhledávání cesty přímo v SQL.

Vyhledávání trasy je založeno na teorii grafů. Graf `G = (V,E)` je uspořádaná dvojice, kde `V` je množina vrcholů (uzlů) a `E` je množina hran [28]. Cestu v grafu můžeme chápat jako posloupnost vrcholů a hran (v_0, e_1, v_1,..., e_t, v_t), kde vrcholy v_0,..., v_t jsou navzájem různé vrcholy grafu G a pro každé $i = 1, 2, ..., t$ je $e_i = \{v_{i-1}, v_i\} \in E(G)$.

Každý bod silnice je reprezentován uzlem a spojen se sousedními pomocí hran. Hrany mohou být orientované v případě jednosměrných ulic. Každá hrana je ohodnocena (délka cesty, čas potřebný k projetí trasy). Jsou zde implementovány 2 algoritmy na vyhledání nejkratší cesty: Dijkstrům algoritmus a A* algoritmus.

Dijkstrův algoritmus publikoval v roce 1959 informatik Edsger Dijkstra. Algoritmus funguje nad nezáporně ohodnoceným orientovaným grafem. Každý sousední uzel je ohodnocen pomocí následující funkce

$$f(u) = f(v) + \{v, u\} \tag{2.3}$$

kde $f(v)$ je vzdálenost (či jiné kritérium) od počátku a $\{v, u\}$ je vzdálenost (či jiné kritérium) od aktuálního uzlu v k sousednímu u. Nejnižší hodnota se vybere za aktuální a dojde opět k prohledání prostoru. Algoritmus je konečný, úplný a optimální.

A* algoritmus je variantou Dijkstrova algoritmu, který přidává heureistický prvek. Každý sousední uzel je ohodnocen pomocí následující funkce

$$f(u) = g(u) + h(u) \tag{2.4}$$

kde $g(u)$ je ohodnocení od počátečního uzlu k sousednímu uzlu a $h(u)$ je heureistická funkce, která odhaduje cenu postupu (např. vzdušná vzdálenost k cíli). Algoritmus je konečný, úplný a optimální.

2.4 OpenStreetMap

Mapové podklady jsou jednou z nejdůležitějších součástí navigace. Bez kvalitních podkladů by sebelepší software nefungovala dobře. Řada nadnárodních firem, zabývajících se vývojem map a GIS, si vytváří vlastní databáze mapových podkladů. Jelikož se jedná o velmi nákladnou záležitost, vznikl před několika lety projekt OpenStreetMap.

OpenStreetMap [14] se zaměřuje na vytváření, editaci a uchovávání geografických dat. Projekt je založen na kolektivní spolupráci a principech Open source. Geografická data jsou volně dostupná pod licencí ODbL[8]. K vytváření dat jsou využívány záznamy z GPS zařízení, popřípadě dochází k digitalizaci mapových podkladů.

Projekt založil v roce 2004 Steve Coast. Později vznikla stejnojmenná nadace, která zaštiťuje celý projekt. Značnému rozšíření pomohla společnost Yahoo, která poskytla letecké snímky jako podklady k získávání dat. S příchodem systému iOS 6 se společnost Apple rozhodla přestat podporovat mapové podklady společnosti Google a vyvinula vlastní řešení založené na projektu OpenStreetMap.

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="CGImap 0.0.2">
  <bounds minlat="54.0889580" minlon="12.2487570" maxlat="
      54.0913900" maxlon="12.2524800"/>
  <node id="1831881213" version="1" changeset="12370172" lat="
      54.0900666" lon="12.2539381" user="lafkor" uid="75625"
      visible="true" timestamp="2012-07-20T09:43:19Z">
    <tag k="name" v="Neu Broderstorf"/>
  </node>
  ...
  <way id="26659127" user="Masch" uid="55988" visible="true"
      version="5" changeset="4142606" timestamp="2010-03-16
      T11:47:08Z">
    <nd ref="292403538"/>
    ...
    <nd ref="261728686"/>
    <tag k="name" v="Pastower Strase"/>
  </way>
  <relation id="56688" user="kmvar" uid="56190" visible="true"
      version="28" changeset="6947637" timestamp="2011-01-12
      T14:23:49Z">
    <member type="node" ref="294942404" role=""/>
    ...
    <member type="way" ref="4579143" role=""/>
    <tag k="name" v="Kustenbus Linie 123"/>
    <tag k="route" v="bus"/>
  </relation>
  ...
</osm>
```

Algoritmus 2.2: Ukázka formátu OSM [15]

[8] Open Database License

2.4.1 Formát OSM

Projekt OpenStreetMap využívá k ukládání vektorových dat vlastní formát - OSM [15], založený na značkovacím jazyku XML. OSM formát je kolekce základních primitiv - uzlů, cest, relací doplněných o atributy. Viz ukázka formátu 2.2

Kořenovým elementem je `<osm>`, který definuje verzi API (v současné době v. 0.6). Element `<bounds>` vymezuje rozsah oblasti, kterou soubor popisuje, tedy minimální/maximální zeměpisná šířka a délka. Následuje kolekce uzlů `<node>`, cest `<way>` a relací `<relation>`. Každý ze základních primitiv obsahuje jednoznačné ID, časovou známku poslední úpravy, informaci o autorovi a verzi.

- Uzel (`<node>`) definuje geoprostorový bod pomocí zeměpisné délky a šířky. Data jsou uložena ve stupních ve standardu WGS84 (reálné číslo s přesností na 7 desetinných míst).
- Cesta (`<way>`) je uspořádaný seznam elementů `<nd>` (2-2000), které odkazují na element `<node>`. Může být otevřená nebo uzavřená. Otevřená cesta reprezentuje vektor, např. silnice, železnice, řeky atd. V případě uzavřené oblasti je první a poslední uzel totožný, jedná se např. o jezera, parky, lesy atd.
- Relace (`<relation>`) je uspořádaný seznam elementů `<member>`, které odkazují na uzly, cesty, popřípadě další relace. Definuje logický nebo geografický vztah elementů, např. linka autobusu, dálnice atd.

Každý z výše zmíněných elementů může obsahovat neomezený počet vnořených značek `<tag>`. Jedná se o doplňující informace ve formátu klíč=hodnota, což mohou být libovolné textové řetězce, např. jméno, typ komunikace, dopravní omezení atd. V následující kapitole jsou popsány nejdůležitější tagy pro tento projekt.

2.4.2 Využité tagy

Mapy OpenStreetMap obsahují obrovské množství informací. Většina z nich není pro tento projekt využitelná. Data je potřeba filtrovat na základě tagů. Jak již bylo zmíněno, `<tag>` je doplňující informace ve formátu klíč=hodnota. Seznam všech používaných tagů lze dohledat na stránce [13].

Bezesporu nejdůležitější pro tento projekt jsou tagy týkající se pozemních komunikací. Klíčem je `highway`, hodnoty pak typy komunikací, např. `motorway` a `motorway_link` pro dálnice a dálniční přivaděče, `primary`, `secondary` a `tertiary` pro silnice první, druhé a třetí třídy. Na druhou stranu v tagu s klíčem `highway` jsou i hodnoty pro pěší či cyklistické trasy, které pro většinu navigací jsou nepotřebné.

Další důležité jsou tagy týkající se oblastí. Každou silnici je třeba zařadit do správného územního celku, např. města. K tomuto účelu slouží tag s klíčem `boudary` s hodnotou `administrative`. Související s touto problematikou je tag s klíčem `admin_level` s hodnotou v rozsahu 1–10, která určuje o jaký typ oblasti se jedná. Pro každou zemi se můžou hodnoty lišit. V případě České republiky odpovídá hodnota 6 krajům, 7 okresům a 8 obcím a městům.

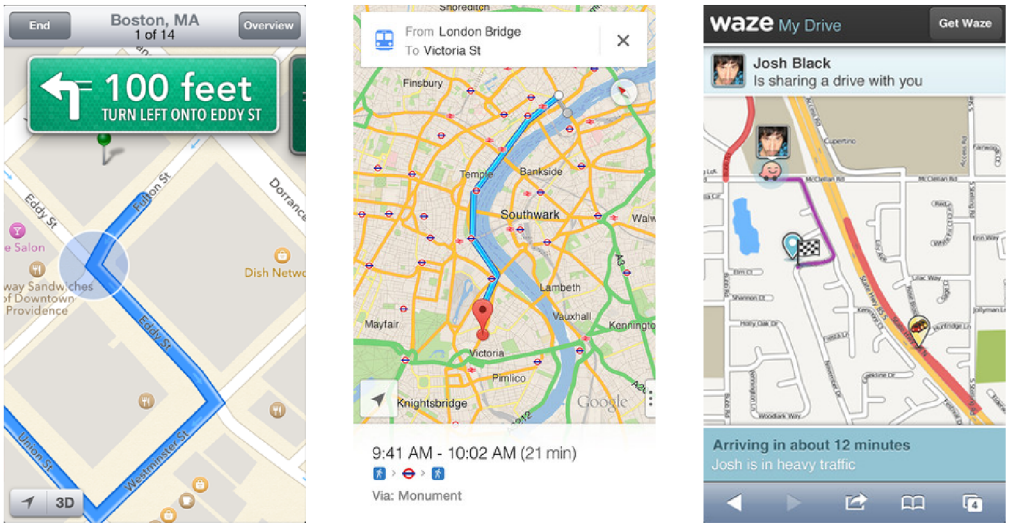
2.5 Analýza konkurenčních řešení

Před návrhem a vývojem navigace byla analyzována některá podobná konkurenční řešení. První vzbranou aplikací jsou vestavěné mapy v systému iOS od společnosti Apple, další 2 navigace jsou zvoleny na základě počtu stažení z elektronického obchodu AppStore.

V této kapitole naleznete krátké představení těchto aplikací a jejich základních vlastností. Testování grafického uživatelského rozhraní se věnuji v kapitole 5.2. Pomocí experimentu s reprezentativním vzorkem testerů zkoumám použitelnost vyvíjené aplikace oproti výše zmíněným aplikacím.

2.5.1 Apple maps

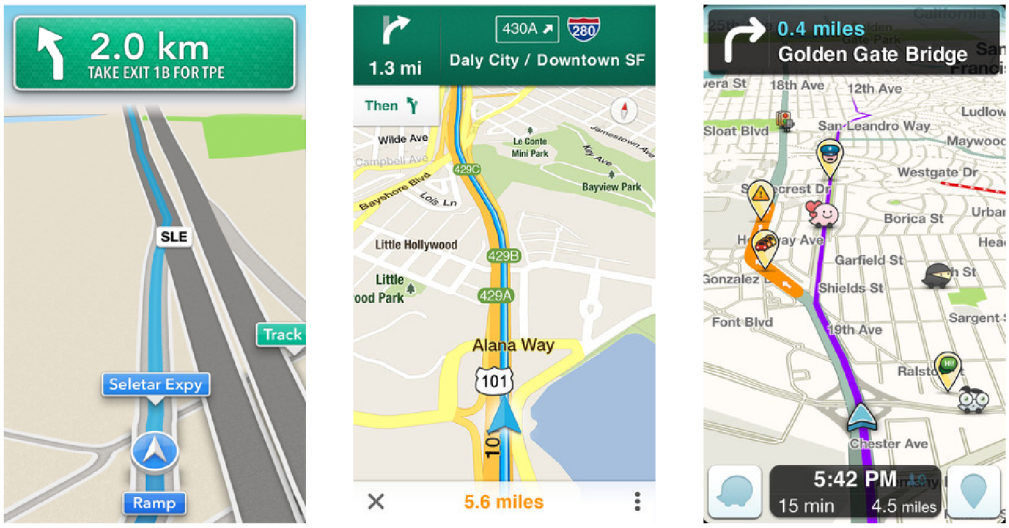
Apple maps [2] je vestavěná aplikace v systému iOS. Poprvé se objevila ve verzi iOS 6.0 (nahradila předchozí aplikaci založenou na mapových podkladech společnosti Google). Zobrazuje mapové podklady od společnosti TomTom, která využívá a rozšiřuje mapové podklady OpenStreetMap. Jedná se o online aplikaci, která veškerá data nahrává z internetu. Vyhledávání cesty probíhá na vzdáleném serveru. Po vyhledání cest aplikace nabídne více variant a uživatel může vybrat nejvhodnější cestu. Mapy podporují turn by turn navigaci. Aplikace obsahuje databázi bodů zájmu (opět pouze na vzdáleném serveru). Mapy podporují 2D i 3D pohled. Zajímavostí 3D zobrazení je zobrazování 3D modulů některých významných budov. Zajímavostí je možnsot krokovat trasu před zahájením navigování.



Obrázek 2.3: Ukázka 2D pohledu z analyzovaných aplikací. Z leva Apple maps (převzato z [2]), Google maps (převzato z [7]), Waze (převzato z [23]).

2.5.2 Google maps

Google maps [7] je aplikace od společnosti Google. Využívá vlastní mapové podklady, které v dnešní době nemají co se týče detailů konkurenci. Jedná se o online aplikaci. Vyhledávání cesty probíhá, stejně jako u aplikace Apple maps, na vzdáleném serveru. Lze zadávat různá kritéria pro vyhledání cesty a aplikace podporuje turn by turn navigování. Design se vymiká zvyklostem tvorby GUI pro systém iOS. Podle očekávání z experimentu použitelnosti (viz kapitola 5.2) jasně vyplynulo, že aplikace pro značnou část testerů působila nepřehledně z důvodu používání atypických grafických prvků. Mapy umožňují zvolit 2D či 3D pohled, které podporuje zobrazování modelů budov. Aplikace je propojena s různými službami Googlu. Nejzajímavější je zřejmě Street View, který dovoluje prohlížet ulice z pohledu první osoby.



Obrázek 2.4: Ukázka 3D pohledu z analyzovaných aplikací. Z leva Apple maps (převzato z [2]), Google maps (převzato z [7]), Waze (převzato z [23]).

2.5.3 Waze

Aplikace Waze [23] je aplikace od stejnojmenné společnosti, která se zaměřuje na potřeby řidičů. Jedná se o online autonavigaci s vestavěnými prvky sociálních sítí. Každý uživatel se stává agentem a informuje ostatní uživatele o situaci na silnici. Dochází zde k hlášení autonehod, dopravních kolon či jiným překážek na silnicích, ale také poskytuje informace například o cenách pohonných hmot na jednotlivých stanicích. Veškerá data jsou nahrávána ze serveru včetně vyhledávání trasy. Aplikace poskytuje i turn by turn navigování s instrukcemi. Dovoluje uživateli poměrně velké možnosti zvolit si pohled kamery podle potřeby.

Kapitola 3

Návrh aplikace

Kapitola se zabývá analýzou požadavků, popisuje návrh aplikace a zvolené metody. Cílem práce je implementace mobilní navigace pro platformu iOS. Jedná se o univerzální aplikaci, kterou lze využít pro zařízení iPhone i iPad. Každé z těchto zařízení má specifický způsob ovládání a jiné návyky koncových uživatelů. Grafické uživatelské rozhraní má být velmi jednoduché a intuitivní. Aplikace má sloužit především k vyhledávání trasy k cíli a následné turn by turn navigaci.

3.1 Analýza požadavků

Z analýzy konkurenčních řešení vyplynulo, že většina jednoduchých navigací využívá online přístup. K veškeré práci vyžadují připojení k internetu, ať už se jedná o nahrávání mapových podkladů, vyhledávání cest nebo adres. Na druhou stranu existují robustní offline aplikace, které veškerá data obsahují. S tím samozřejmě souvisí i mnohonásobně větší paměťová náročnost.

Rozhodl jsme se vytvořit navigaci, která bude kombinací těchto přístupů. V případě dosahu internetu poskytne uživateli podrobné mapové podklady a umožní procházet online databáze adres po celém světě. Pokud uživatel nebude v dosahu sítě, poskytne možnost jednoduché offline navigace.

Aplikace bude disponovat dvěma módy zobrazení mapových podkladů - 2D a 3D. První z nich slouží k procházení mapy či vyhledávání trasy. Pokud již dojde k vyhledání trasy, aplikace se přepne do navigačního 3D módu a zobrazí se panel turn by turn navigace se směrovými instrukcemi.

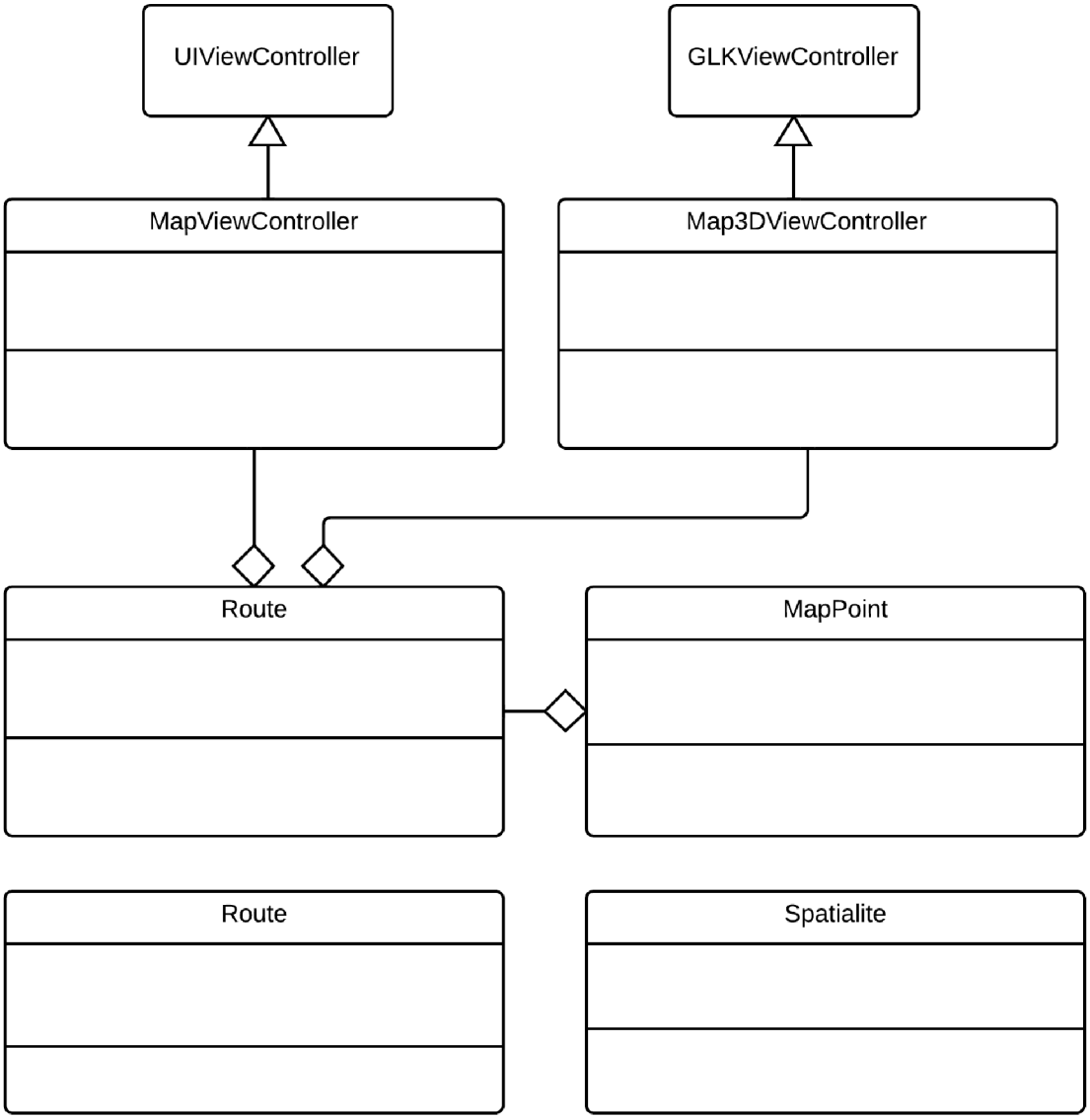
3.2 Databáze

Ukládání prostorových dat úzce souvisí s každým GIS. K tomuto účelu poslouží databázový systém SpatiaLite, který byl podrobně analyzován v kapitole 2.3. Z analýzy požadavků je jasné, že zde musí existovat několik typů ukládaných informací. Nejdůležitější ukládané informace v tomto projektu jsou bezesporu silnice, které slouží k offline vyhledání trasy a následné navigaci k cíli. Jedná se o datový typ LINESTRING, doplněný o další potřebné informace.

3.3 Diagram tříd

Aplikace se vytváří podle návrhového vzoru Model-View-Controller (MVC), který rozděluje aplikaci do tří logicky oddělených částí: datový model, řídící model a uživatelské rozhraní.

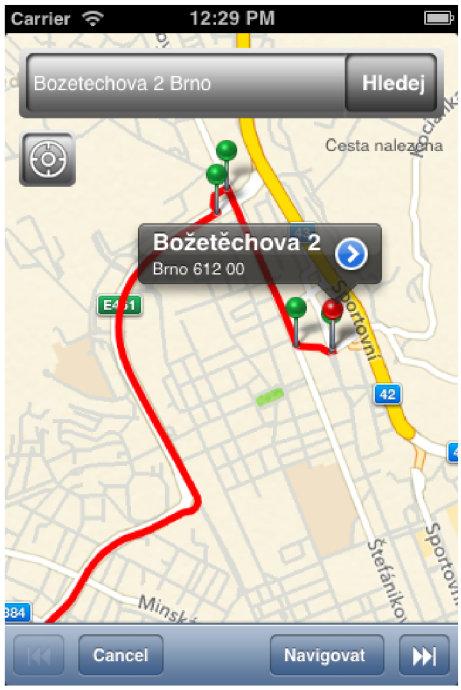
Na obrázku 3.1 je znázorněn zjednodušený diagram tříd. Datový model je v tomto případě třída zajišťující komunikaci s databázovým systémem SpatiaLite. Veškerá manipulace s daty se provádí pomocí třídy SpatiaLite. Třída Route zajišťuje veškerou logiku nad vyhledanou trasou (např. získávání směrových instrukcí nebo vzdálenost do cíle). Problematice navigačního algoritmu se věnuji v kapitole 3.5. Řídící model obsahující řadu tříd (odvozených od ViewController), který propojuje datový model s uživatelským rozhraním.



Obrázek 3.1: Zjednodušený diagram tříd

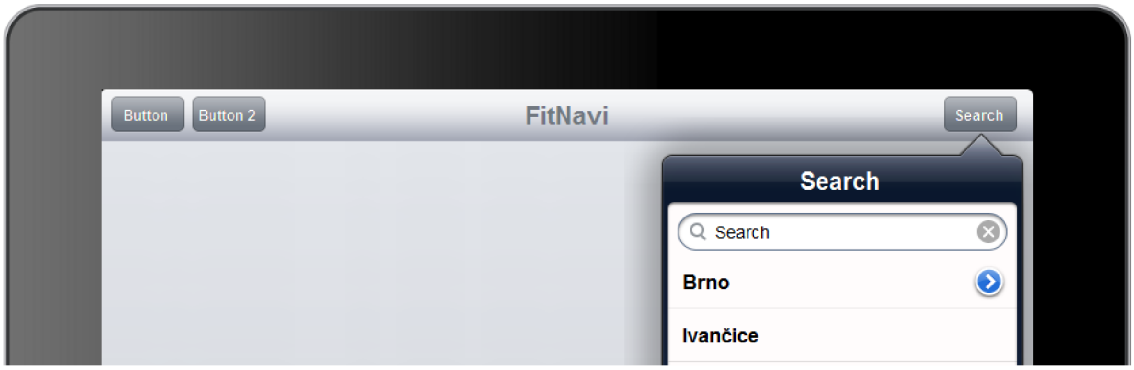
3.4 Grafiké uživatelské rozhraní

Při návrhu jsem se snažil o maximální jednoduchost a intuitivnost. Jsou zde dva odlišné vzhledy, jeden pro iPhone verzi a druhý pro iPad. Ze zkušeností s oběma typy zařízení vím, že každý vyžaduje odlišný přístup.



Obrázek 3.2: Návrh uživatelského rozhraní pro iPhone

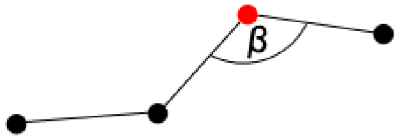
Zatímco uživatelé mobilního telefonu iPhone obvykle pracují s aplikací pomocí překrývajících oken propojených pomocí navigačního panelu, díky velikosti obrazovky si uživatelé iPadu mohou dovolit provádět většinu úkonů na jedné obrazovce pomocí vyskakovacích (popup) oken.



Obrázek 3.3: Návrh uživatelského rozhraní pro iPad

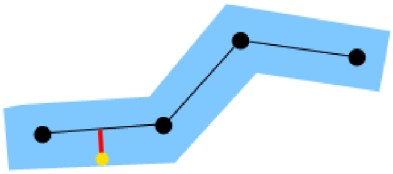
3.5 Navigační algoritmus

Při vyhledání cesty musí dojít i k vytvoření navigačních instrukcí či vypočítání vzdálenosti do cíle. Vyhledaná trasa je kolekce uzlů, přes které cesta vede. Je třeba rozlišit uzly obyčejné (kopírují pouze tvar cesty) a křižovatky (místo, kde je třeba uživatele informovat o změně směru). Pro každou křižovatku je potřeba vypočítat úhel, který svírá s předchozím a následujícím uzlem pomocí Kosinové věty. Podle hodnoty je třeba rozhodnout o jaký typ křižovatky se jedná. Na následujícím obrázku 3.4 je vidět ukázka části reálné cesty a zobrazena křižovatka. V tomto případě svírající úhel je v rozsahu odpovídající pravotočivé zatáčce.



Obrázek 3.4: Křižovatka

Během cesty je nutné kontrolovat pozici zařízení s pozicí v rámci mapy. V případě, že dojde k vychýlení z trasy je nutné přepočítat trasu včetně navigačních instrukcí. Při každé změně pozice dojde k zjištění vzdálenosti od nejbližší cesty, pokud je zařízení v toleranci, pokračuje se, jinak dojde k přepočítání.



Obrázek 3.5: Navigační algoritmus

Kapitola 4

Implementace

Obsahem kapitoly je popis zajímavých částí implementace. Aplikace FitNavi je implementována v jazyce Objective-C s vyjímkou úseků využívající C API externích knihoven. Vývoj probíhal ve vývojovém prostředí Xcode a byly využity i další nástroje ze sady iOS SDK (viz 2.1.2). Cílovou platformou je iOS, kontrétně verze 6.1.

Při vývoji byla snaha dodržet návrhový vzor MVC a oddělit tak datovou, řídící a prezentační logiku. Během vývoje došlo k menším změnám oproti navrhovanému diagramu tříd popsanému v kapitole 3.3.

4.1 OpenStreetMap parser

K získání potřebných dat z projektu OpenStreetMap bylo potřeba vytvořit program pro zpracování dokumentu XML. Script byl implementován v jazyce Python. Jelikož mapové poklady jsou soubory s velikostí v řádek stovek megabytů, k parsování XML dokumentu bylo vhodné využít SAX technologii, konkrétně balíček `xml.sax`. Nenačítá celý dokument do paměti, jedná se o stavově řízené procházení dokumentu. Pomocí tohoto scriptu došlo k vytvoření potřebných tabulek v databázovém systému a naplnění daty. Aplikace obsahuje data z Jihomoravského kraje dostupné ze serveru CloudMade.

4.2 CLLocationManager

V kapitole 2.2 je popsána vrstva `Core Location`, která zajišťuje komunikaci mezi aplikací a hardwarem nutným k zjištění aktuální polohy. Pracuje s několika technologiemi určování polohy, s čímž souvisí převážně přesnost a rychlost určení.

Vstrvu Core Location zpřístupňuje třída CLLocationManager, která je součástí MapKit frameworku. Instancí třídy CLLocationManager žádáte uživatele o povolení sledovat současnou polohu jeho zařízení, resp. získávat informace o nasměrování přístroje. Nastavením proměnných `desiredAccuracy` a `headingFilter` lze zvolit požadovanou přesnost. V tomto případě je snaha získát přesné údaje (konstatnty `kCLLocationAccuracyBestForNavigation` a `kCLHeadingFilterNone`). Metoda `startUpdatingLocation` a `startUpdatingHeading` pro nasměrování, spouští monitoring. V případě odsunutí aplikace na pozadí či vypnutí se systém iOS postará o korektní ukončení.

ViewController, který třídu `CLLocationManager` využívá, musí implementovat rozhraní `CLLocationManagerDelegate`. Metoda `locationManager:didUpdateLocations:` je volána v případě změny polohy, metoda `locationManager:didUpdateHeading:` v případě změny

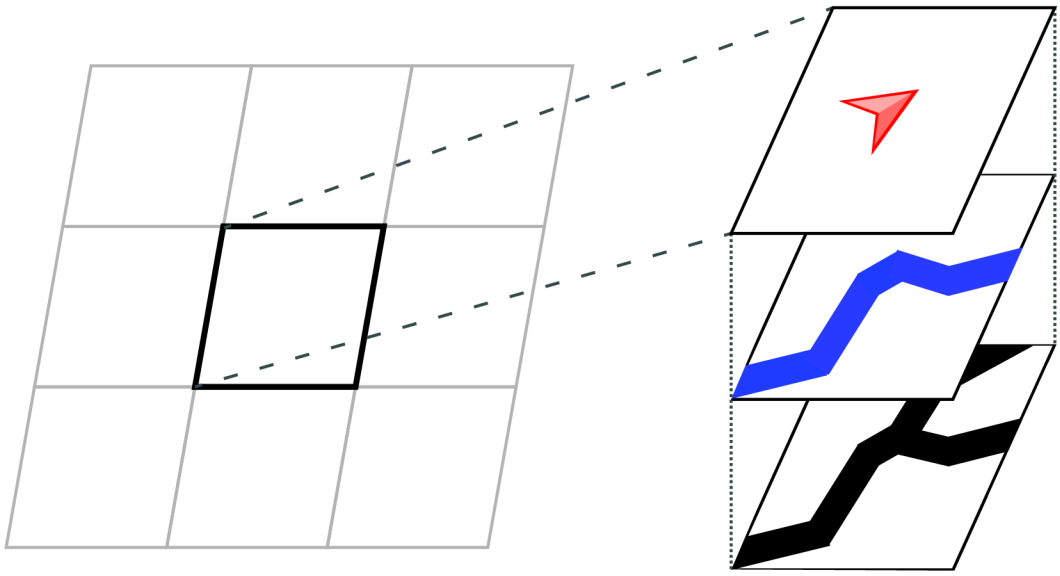
nasměrování zařízení a metoda `locationManager:didFailWithError:` v případě chyby či nemožnosti zjistit polohu.

iOS simulátor nabízí jen částečnou možnost simulace polohy uživatele. Dovoluje nastavit pouze přesnou polohu, avšak neumožňuje simulovat jízdu. K odsimulování různých tras bylo nutné doimplementovat třídu NSLocationManagerSimulator. Nabízí stejné možnosti jako CLLocationManager a dodřuje rozhraní CLLocationManagerDelegate. Po určitých časových intervalech jsou volány metody rozhraní s fiktivními daty.

4.3 Zobrazování mapových podkladů

Zobrazení offline mapových podkladů je realizováno pomocí OpenGL ES. Jedná se o knihovnu hardwarově akcelerovanou a určenou k vykreslování 2D/3D objektů.

Pro zobrazení bylo využito principu dlaždic znázorněném na obrázku 4.1. Každá dlaždice odpovídá určitému úseku mapy s daným přiblížením (zoom level). Pro každý zoom level existuje matice čtverců s různými rohovými souřadnicemi. Každý čtverec je adresovatelný pomocí `x` a `y` souřadnic a daného přiblížení. Při změně polohy zařízení dojde k přepočítání, do kterého čtverce souřadnice náleží.



Obrázek 4.1: Ukázka dlaždicového principu k zobrazení mapových podkladů.

Celá scéna je tvořena několika čtverci, které spolu sousedí. Pomocí knihovny Quartz pro vykreslování vektorové 2D grafiky dojde v vytvoření textury, která se aplikuje na příslušnou dlaždici. Každá dlaždice se skládá z více vrstve. Nejnižší vrstva zobrazuje mapové podklady, vyšší pak vyhledanou cestu nebo šipku znázorňující nasměrování zařízení.

Uživatel může pomocí gest ovládat scénu. Může nastavit sklon (vybírá zda chce oblast sledovat ve 2D či 3D režimu), posunovat se v rámci mapy či zapínat a vypínat sledování aktuální polohy.

Kapitola 5

Testování

Vytvoření softwaru funkční, jednoduché navigace s intuitivním ovládáním bylo cílém této práce. Tato kapitola se zabývá testováním, konkrétně se zaměřuje především na testování funkčnosti navigačního algoritmu a testování grafického uživatelského rozhraní (GUI). Navigační algoritmus byl otestován pomocí simulace cest a sledování očekávaných reakcí aplikace. Testování grafického uživatelského rozhraní probíhalo formou experimentu. Snahou bylo zdokonalit intuitivnost a použitelnost GUI. Byly stanoveny úkoly, které měli uživatele cílové skupiny plnit. Reakce, poznámky a postupy se zaznamenaly. Experiment probíhal na jak na vyvíjené aplikace, tak na aplikacích třetích stran popsaných v kapitole 2.5. Testování probíhalo v průběhu vývoje a výsledky napomáhaly zlepšovat GUI navigace.

5.1 Navigační algoritmu

Jak bylo zmíněno v kapitole 4.2, vestavěný simulátor nedisponuje rozsáhlými možnosti simulace trasy. Díky implementaci vlastní simulační třídy CLLocationManagerSimulator bylo možné testovat aplikaci v simulátoru s reálnými daty jako v případě jízdy autem. Během několika zkušebních jízd byla sbírána reálná data (poloha, rychlost a nasměrování přístroje) několika tras. V případě překladu aplikace pro simulátor dojde k vložení simulační třídy.

5.2 Grafické uživatelské rozhraní

Testování GUI probíhalo formou experimentu. Každý z testerů dostal stejnou sadu úkolů, které měl za úkol plnit. V průbehu testu byly reakce, poznámky a postupy zaznamenávány, včetně času stráveného plněním jednotlivých úkolů. Z důvodu porovnatelnosti výsledů byly testovány 3 apliakce třetích stran s vyvíjenou aplikací. Aplikace třetích stran byly blíže přiblíženy v kapitole 2.5. Jedná se o aplikace Apple Maps, Google Maps a Waze.

Experiment probíhal na reprezentativním vzorku deseti testerů. Každý z nich má jiné zkušensti s používáním počítačů a mobilních telefonů. Jelikož GUI aplikace má být maximálně jednoduché a použitelné, nekladl jsem žádná omezení pro výběr testerů, vyjma vlastnění mobilního telefonu.

- Tester 1, 2 - běžný uživatel počítače, zkušenost s mobilním telefonem s hardwarovými tlačítky

- Tester 3 - pokročilý uživatel počítače, zkušenost s dotykovým mobilním telefonem (Nokia)
- Tester 4, 5, 6 - pokročilý uživatel počítače, zkušenost s dotykovým mobilním telefonem (Android)
- Tester 7, 8 - pokročilý uživatel počítače, zkušenost s dotykovým mobilním telefonem (iOS)
- Tester 9 - infromatik, zkušenost s dotykovým mobilním telefonem (Android) a tabletem (Android)
- Tester 10 - infromatik, zkušenost s dotykovým mobilním telefonem (iOS) a tabletem (iPad)

Experiment probíhal ve dvou fázích. V první došlo pouze k testování použitelnosti aplikací třetích stran. V druhé fázi došlo k testování vyvíjené aplikace. Všichni testeři byly vyzváni ke splnění následujících úkolů pro každou aplikaci:

1. Seznamte se s aplikací (1 minuta), popište k čemu pravděpodobně slouží jednotlivé prvky uživatelského rozhraní.
2. Vyhledejte na mapě Fakultu informačních technologií VUT v Brně (Božetěchova 2, 612 66 Brno).
3. Vyhledejte trasu k této fakultě.
4. Nechte se navigovat k cíli.
5. Popište klady a zápory testované aplikace.

5.2.1 Vyhodnocení experimentu

Podle očekávání nejmenší problémy s plněním úkolů měli nejvíce technicky zdatní testeři. V průměru byly až o polovinu rychlejší. Nejrychleji splnili úkoly testeři vlastnící mobilní žařízení s operačním systémem iOS.

V případě aplikace Apple maps testeři ocenili jednoduchost aplikace. Prakticky nikdo neměl větší problémy s vyhledáním trasy. Několika respondentům vadil způsob zapínání a vypínání natáčení mapy. V případě, že tlačítko pro natáčení vyplnuli, mapa zůstala v poslední pozici a tedy nesměrovala na sever.

Aplikaci Google maps ocenili z pohledu mapových podkladů, jelikož ze všech testovaných aplikací byly nejpodrobnější. Naopak tato aplikace získala nejvíce záporných ohlasů na design, respektive jeho použitelnost. Uživatelé měli problém s výběrem trasy a zapnutím navigačního módu. 1 tester na tuto možnost vůbec nepřišel.

Aplikaci Waze neznal žádný tester. Ocenili sociální prvky v aplikaci. Některým testerům vadila nutnost přihlášení k sociální síti k plnému využití.

Kapitola 6

Závěr

Cílem práce bylo vytvořit jednoduchou navigaci pro operační systém iOS, respektive pro zařízení iPhone a iPad. Aplikace měla zvládat zjišťování aktuální polohy, zobrazovat mapové podklady získané z projektu OpenStreetMap či vyhledat a navigovat uživatele k cíli pomocí instrukcí. Jelikož se mělo jednat o offline navigaci, bylo třeba řešit vhodným způsobem ukládání geografických dat.

Problematika navigací je velice rozsáhlý obor. Bylo potřeba nastudovat poznatky z oblastí geografie a kartografie. Důležité bylo pochopit princip GPS systému i možnosti, které nabízí cílová platforma iOS. Výběr OpenStreetMap projektu pro získání mapových podkladů se ukázal být použitelným řešením i při vývoji komerčních apliakcí. K ukládání geografických dat byla zvolena knihovna SpatiaLite. V této knihovně vidím velký potenciál pro mobilní geolokační aplikace, jelikož většina mobilních operačních systémů využívá databázový systém SQLite, nad kterým je knihovna postavena. Poskytuje jak základní funkce pro práci s geolokačními daty, tak i pokročilou práci s daty v databázi. Zajímavou částí je i implementace navigačního algoritmu či způsob vykreslování podkladových map. Během vývoje byl proveden experiment, který se snažil mapovat chování cílových uživatelů. Uživatelským testováním použitelnosti vyvíjené aplikace, i aplikací třetích stran s podobným zaměřením, byla snaha zjistit, jakým zkusobem běžný úživatel pracuje s geolokační navigací. Těmto poznatkům byl přizpůsoben návhr grafického úživatelského rozhraní.

Výsledkem snažení je jednoduchá, funkčí navigace, která splňuje základní požadavky zadání. Zajímavý je koncept kombinace online i offline přístupu. V případě připojení k internetu dokáže aplikace využít plný potenciál online dat, avšak dokáže se obejít i bez připojení a slouží jako jednoduchá offline navigace. V budoucnu by bylo dobré aplikaci rozšířit o další funkce. Jednu z možností vydím v úpravě algoritmu pro vyhledání trasy o možnost filtrovat jistý druh komunikace, či se vyhýbat dopravním kolonám a nehodám. Propracovat a optimalizovat lze i vykreslování mapových podkladů pomocí Open GL. Poměrně jednoduše by šly mapy rozšířit o modely 3D budov. V případě reálného nasazení aplikace je nutné rozšířit pokrývanou oblast. V apliakci nebylo z časových důvodů implementováno hlasové navigování.

Literatura

[1] Alasdair, A.: *Geolocation in iOS*. O'Reilly Media, 2012, iSBN 978-1-449-30844-5.

[2] Apple, Inc.: Apple Maps [online]. `http://www.apple.com/ios/maps/`, 2013 [cit. 2013-04-04].

[3] Chatelier, P.: From C++ to Objective-C [online]. `http://pierre.chachatelier.fr/programmation/fichiers/cpp-objc-en.pdf`, 2009 [cit. 2013-03-31].

[4] Furieri, A.: What's this SRID stuff? [online]. `http://www.gaia-gis.it/gaia-sins/spatialite-cookbook/html/srid.html`, 2011 [cit. 2013-03-10].

[5] Furieri, A.: SpatiaLite Cookbook [online]. `http://www.gaia-gis.it/spatialite-2.4.0-4/spatialite-cookbook/`, 2011 [cit. 2013-04-04].

[6] GEOS komunita: GEOS - Geometry Engine, Open Source [online]. `http://trac.osgeo.org/geos/`, 2013 [cit. 2013-03-10].

[7] Google, Inc.: Google Maps [online]. `http://itunes.apple.com/app/id585027354`, 2013 [cit. 2013-04-04].

[8] Hrubý, M.: *Geografické Informacní Systémy (GIS) Studijní opora*. FIT VUT v Brně, 2006.

[9] Kochan, S. G.: *Objective-C 2.0 Výukový kurz programování pro Mac OS X a iPhone*. Computer Press, a.s., 2010, iSBN 978-80251-2654-7.

[10] Mark, D.; Nutting, J.; LaMarche, J.; aj.: *Beginning iOS 6 Development: Exploring the iOS SDK*. Apress Media LLC, 2013, iSBN 978-1-4302-4513-1.

[11] National Imagery and Mapping Agency: Department of Defense World Geodetic System 1984 [online]. `http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf`, 1984 [cit. 2013-04-03].

[12] Open Geospatial Consortium: Simple Feature Specification [online]. `http://www.opengeospatial.org/standards/sfs`, 2013 [cit. 2013-03-11].

[13] OpenStreetMap komunita: Map Features [online]. `http://wiki.openstreetmap.org/wiki/Map_Features`, 2013 [cit. 2013-01-15].

[14] OpenStreetMap komunita: OpenStreetMap [online]. http://www.openstreetmap.org/, 2013 [cit. 2013-01-15].

[15] OpenStreetMap komunita: OSM XML [online]. http://wiki.openstreetmap.org/wiki/.osm, 2013 [cit. 2013-01-15].

[16] Proj4 komunita: PROJ.4 - Cartographic Projections Library [online]. http://trac.osgeo.org/proj/, 2013 [cit. 2013-03-10].

[17] Rideout, P.: *iPhone 3D Programming: Developing Graphical Applications with OpenGL ES*. O'Reilly Media, 2010, iSBN 978-1-4493-8830-0.

[18] Smithwick, M.: *Pro OpenGL ES for iOS*. Apress Media LLC, 2011, iSBN 978-1-4302-3841-6.

[19] Spatialite komunita: C APIs reference list [online]. http://www.gaia-gis.it/spatialite-2.3.0/spatialite-C-API-2.3.0.html, 2011 [cit. 2013-03-09].

[20] Spatialite komunita: SpatiaLite manual [online]. http://www.gaia-gis.it/spatialite-2.3.1/spatialite-manual-2.3.1.html, 2011 [cit. 2013-03-10].

[21] Spatialite komunita: SQL functions reference list [online]. http://www.gaia-gis.it/spatialite-2.4.0-4/spatialite-sql-2.4-4.html, 2011 [cit. 2013-03-10].

[22] Spatialite komunita: SpatiaLite [online]. https://www.gaia-gis.it/fossil/libspatialite/index, 2013 [cit. 2013-02-15].

[23] Waze, Inc.: Waze [online]. http://itunes.apple.com/app/id323229106, 2013 [cit. 2013-04-04].

[24] Wei-Meng, L.: *Beginning iOS 5 Application Development*. John Wiley & Sons, Inc., 2012, iSBN 978-1-118-14425-1.

[25] Wikipedia: Global Positioning System [online]. http://en.wikipedia.org/wiki/Global_Positioning_System, 2013 [cit. 2013-04-10].

[26] Wikipedia: iOS (Apple) [online]. http://cs.wikipedia.org/wiki/IOS_(Apple), 2013 [cit. 2013-04-15].

[27] Wikipedia: iOS [online]. http://en.wikipedia.org/wiki/IOS, 2013 [cit. 2013-04-17].

[28] Zbořil, F.: *Základy umělé inteligence : Studijní opora*. FIT VUT v Brně, 2012.

Seznam příloh

Příloha A

Obsah CD

- Zrojové kódy vyvíjené aplikace v adresáři **/FitNavi/**, včetně Xcode projektu
  - **/FitNavi/FitNavi/** – iOS aplikace
  - **/FitNavi/Spatialite/** – knihovna SpatiaLite
  - **/FitNavi/Proj4/** – knihovna Proj4
  - **/FitNavi/Geos/** – knihovna GEOS
- Tato práce ve formátu PDF ve složce **/thesis/**
- Adresář **/thesis/src/** obsahuje zdrojové soubory této práce ve formátu LaTeX
- Složka **/downloads/** obsahuje originální neupravený obsah stažený z internetu (využité knihovny, mapové podklady,...)

Příloha B

Manuál

K překladu a spuštění vyvíjené aplikace je třeba vlastnit počítač od společnosti Apple s operačním systémem Mac OS X, vývojové prostředí Xcode a iOS SDK. Aplikace byla vyvíjena a testována na následující konfiguraci:

- Mac OS X 10.7.5
- Xcode 4.6
- iOS SDK 6.1

Ke spuštění aplikace na přístroji iPhone nebo iPad je třeba mít v zařízení operační systém iOS minimálně verze 6.1. Vývojář musí vlastnit vývojářský účet a dané zařízení mít s účtem propojené. Vytvořený Provisioning profile je třeba nahrát do programu Xcode. Ve vyvíjené aplikaci v nastavení projektu je nutné přepsat Bundle Identifier na vývojářské ID.