UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Zobrazení výškového povrchu v 2D a v 3D zobrazení pomocí OpenGL

Pavel Kozel

Bakalářská práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Pavel Kozel**
Osobní číslo:	**I08087**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Zobrazení výškového povrchu v 2D a v 3D zobrazení pomocí OpenGL.**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je zobrazení výškového povrchu v 2D a 3D (3D myšleno jako plastické mapy) pomocí OpenGL.
Teoretická část:
Seznámení s projektem SRTM (databáze nadmořských výšek Země). Popis možností OpenGL pro zobrazení 2D a 3D, princip zobrazení ve vertex bufferu, jeho výhody a nároky.
Analýza:
Porovnání vykreslení 2D grafiky pomocí OpenGL s jinou grafickou platformou (Java, Qt C++).
Implementační část:
Vytvoření aplikace v Qt C++ pro zobrazení výškových dat. Aplikace bude mít možnosti přepínat mezi 2D a 3D zobrazením a základní pohyb po mapě.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***PRATA, Stephen. Mistrovství v C++. 3. aktualizované vydání. Brno : Computer Press, 2007. 1119 s. ISBN 978-80-251-1749-1.**
***SHREINE, Dave, et al. OpenGL : Průvodce programátora. Vyd. 1. Brno : Computer Press„ 2006. 680 s. ISBN 80-251-1275-6.**
***internetové zdroje : qt.nokia.com, www.jpl.nasa.gov/srtm/, www.msdn.com**

Vedoucí bakalářské práce: **Ing. Jaroslav Štroch**
Katedra informačních technologií

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 18.8.2013

Pavel Kozel

Poděkování

Tímto bych rád poděkoval všem, kteří mě při tvorbě mé bakalářské práce podporovali nebo mi jakkoliv pomohli. Zejména pak mému vedoucímu panu Ing. Jaroslavu Štrochovi za jeho odbornou pomoc a cenné rady. Dále bych chtěl poděkoval mé rodině a přátelům za podporu během studia.

Anotace

Cílem bakalářské práce je zobrazení výškového povrchu v 2D a 3D (3D myšleno jako plastické mapy) pomocí OpenGL. Teoretická část obsahuje seznámení s projektem SRTM (databáze nadmořských výšek Země). Popis možností OpenGL pro zobrazení 2D a 3D, princip zobrazení ve vertex bufferu, jeho výhody a nároky. Implementační část zahrnuje vytvoření aplikace v Qt C++ pro zobrazení výškových dat a porovnání rychlosti kreslení pomocí OpenGL a třídy QPainter v Qt C++.

Klíčová slova

Výškové soubory, SRTM, OpenGL, vertex buffer, C++, Qt Framework, Qt Creator

Title

Displaying altitude of the surface in 2D and 3D using OpenGL.

Annotation

The aim of this thesis is rendering of elevation surfaces in 2D and 3D (3D as relief maps) using OpenGL. The theoretical part includes explanation of the SRTM project (database of the Earth altitudes,) description of the OpenGL 2D and 3D capabilities, rendering using vertex buffer and its benefits. The implementation section includes creating an application in Qt C++ to view elevation data and compare the speed of drawing with OpenGL and QPainter class in Qt C++.

Keywords

Altitude files, SRTM, OpenGL, vertex buffer, C++, Qt Framework, Qt Creator

Obsah

Seznam zkratek

WGS	World Geodetic System
UTM	Universal Transverse Mercator
NATO	North Atlantic Treaty Organization
GPS	Global Positioning System
DEM	Digital Elevation Model
SRTM	Shuttle Radar Topography Mission
NASA	National Aeronautics and Space Administration
NGA	National Geospatial-Intelligence Agency
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
VTOPÚ	Vojenský topografický ústav Dobruška
ZABAGED	Základní báze geografických dat
GUI	Graphical User Interface
IDE	Integrated Development Environment
GPL	General Public License
LGPL	Lesser General Public License
OpenGL	Open Graphics Library
API	Application Programming Interface

Seznam obrázků

Seznam tabulek

1 Úvod

Tématem mé bakalářské práce je Zobrazení výškového povrchu v 2D a 3D pomocí OpenGL.

K vypracování mé bakalářské práce mě vedlo využití grafického rozhraní OpenGL, jako jedné z technologií pro zobrazení zdrojových dat. Již delší dobu jsem se o OpenGL zajímal na teoretické úrovni a tato práce mě umožnila využití těchto znalostí v praxi.

Cílem bakalářské práce je seznámení s projektem SRTM (databáze nadmořských výšek Země) a dalšími výškovými soubory. Dalším cílem je porovnání rychlosti zobrazovaných dat pomocí OpenGL s využitím vetex bufferu a třídy QPainter v Qt C++.

Teoretická část zahrnuje seznámení s pojmy z oblastí informačních technologií, geografie a kartografie. V jedné samostatné kapitole se věnuji digitálním výškovým modelům, jejich pořizování a seznámení s nejznámějšími modely. Hlavní pozornost je věnována projektu SRTM.

Praktická část obsahuje představení použitých technologií při vývoji aplikace, která slouží pro zpracování výškových souborů projektu SRTM do vlastního formátu a jejich zobrazování pomocí OpenGL a Qt C++.

2 Vymezení základních pojmů

Úvodem této bakalářské práce je vhodné vymezit základní pojmy z oblastí informačních technologií, kartografie a geografie.

2.1 Oblast Informačních technologií

2.1.1 Systém

Systém lze definovat jako souhrn spolu souvisejících prvků, které jsou sdruženy do smysluplného celku. Většinou se skládá z různých částí, které jsou propojeny za účelem toku informací, energie, či materiálu.

Jednotlivé prvky systému mohou být též systémem, subsystémem, to znamená, že daný prvek se skládá z dalších prvků.

2.1.2 Informace

Samostatný pojem informace je velmi široký, s množstvím různých významů. V návaznosti na systém může být informace chápána jako údaj o prvcích, jejich stavu a procesech v nich probíhajících. [1]

V informačních technologiích lze informace definovat v nejjednodušší formě, jako PRAVDA, či NEPRAVDA. Toto v programové technice odpovídá signálům „0“ a „1“ také zvané jako bity.

2.1.3 Data

Soubor informací získaných pozorováním určité veličiny, či jevu se označuje pojmem data. Tedy záznamem, který může představovat obecné vlastnosti jevu, nebo jeho vztahu vůči nějaké spojité stupnici.

Za data jsou v informatice považovány veškeré informace v digitální, číselné podobě určené ke zpracování počítačem. Data (např. text, zvuk, obrázek atd.) jsou uloženy na paměťovém médiu (operační paměť, pevný disk, ...) v podobě posloupnosti čísel (bajtů).

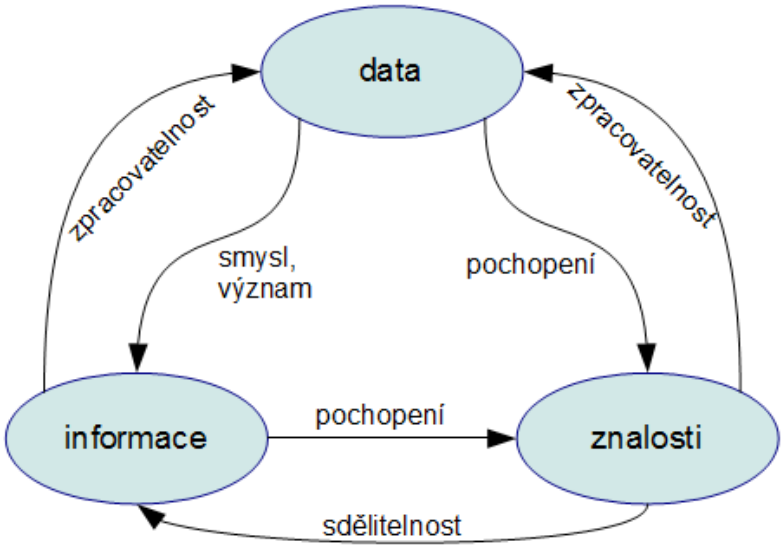
2.1.4 Informační systém

Pojem Informační systém (IS) nemá přesnou definici, jelikož každý uživatel nebo tvůrce IS používá jiné terminologie a upřednostňuje jiné aspekty. IS však lze chápat jako systém vzájemně propojených informací a procesů, které s informacemi pracují. Procesy tedy zpracovávají vstupní informace do systému a transformují je na výstupní informace ze systému.

Do funkce IS se promítá i jeho okolí. Toto okolí tvoří veškeré objekty, které změnou svých vlastností ovlivňují samotný systém, a také naopak objekty, které mění své vlastnosti v závislosti na systému.

IS je tedy většinou softwarové vybavení firmy, které na základě zpracovávaných informací je schopné řídit procesy podniku nebo tyto informace poskytovat řídícím pracovníkům.

[2]



Obrázek 1 – Relace mezi daty, informacemi a znalostmi

Zdroj: autor

Schéma na Obrázku 1 zobrazuje relace mezi daty, informacemi a znalostmi, které tvoří součást informačního systému. *Data* mohou představovat firemní databázi a *znalosti* uživatele systému. Vazby mezi prvky jsou tedy následující:

- Data odrážejí nějakou skutečnost na základě informací. Slouží ke zpracování a je možné je využít jako informace nebo je dát k dispozici uživatelům.
- Informace je možné zpracovat, převést do digitální podoby, nebo je předat uživatelům k pochopení.
- Znalosti mohou být chápany jako uživatel, který informace a data studuje a třídí, následně na základě získaných vědomostí je může reprodukovat jako nové informace či data.

2.1.5 Program

Program popisuje realizaci úlohy počítačem v přesně stanovené formě, posloupnosti instrukcí. Program vytváří programátor zápisem algoritmu v programovacím jazyce, výsledný zdrojový kód je poté převáděn do strojového kódu, který je pak vykonáván procesorem.

2.1.6 Soubor

Soubor představuje pojmenovanou množinu dat uloženou na datovém médiu, se kterou lze pracovat jako s jedním celkem. Podle toho, jak má být soubor interpretován, lze soubory rozdělit následovně:

- Textové – obsah souboru, jednotlivé bajty, reprezentují znaky ve znakové sadě.
- Binární – obsahuje jakákoliv data, která musejí být zpracována počítačovým programem.

Obsahem souboru může být jeden druh dat (například text, program, obrázek, …), stejně tak mohou být soubory složeny z dalších souborů a jiných objektů (knihovny, ISO obraz disku, archivní soubory).

2.1.7 Hardware

Hardware označuje veškeré fyzicky existující technické vybavení počítače. Typicky základní deska, procesor, pevné disky, operační paměti, grafická karta a další.

2.1.8 Software

Software je v současné době obecně používaný jako pojem pro veškeré programové vybavení počítače, či jiná zařízení. Tedy od základního systému, operačního systému, až po multimediální programy. Operační systém slouží k ovládání hardwarového vybavení počítače.

2.2 Oblast geografie a kartografie

Cílem této podkapitoly je seznámit čtenáře s pojmy z oblasti geografie a kartografie, dále s nejpoužívanějšími souřadnicovými systémy pro určování polohy na zemi.

2.2.1 Geografie

Pojem geografie vychází ze dvou řeckých slov *geos* (česky pozemský, zemský) a *grafein* (psáti), česky zeměpis. Geografie je tedy věda zabývající se studiem prostorových jevů na Zemi, zemské krajiny, jejich vzájemnou interakcí a působením v čase.

Geografie je popisována na několika základních úrovních. Jsou to fyzická geografie, sociální geografie, regionální geografie a kartografie. Pro potřeby této práce je zde popsána pouze úroveň kartografie.

[3]

2.2.2 Kartografie

Jak již bylo zmíněno, kartografie je jednou z úrovní geografie. Kartografie je vědním oborem mající svůj předmět zkoumání, odbornou terminologii, vlastní jazyk pro popis praktických i teoretických aspektů a také matematicky podložené teorie a zákonitosti. Nejčastějším výsledkem práce kartografů jsou různé mapy.

Díky dlouhému historickému vývoji kartografie vznikla řada definic, například:

„Kartografie je věda o sestavování map všech druhů a zahrnuje veškeré operace od počátečního vyměřování až po vydání hotové produkce. (United Nations, Department of Social Affairs, 1949)“

„KARTOGRAFIE je věda o zobrazování a studiu prostorového rozmístění, spojení a vzájemných vazeb jevů přírody a společnosti (i jejich změn v čase) prostřednictvím zvláštních obrazově znakových modelů – kartografických vyobrazení. (Saličev, 1976)“

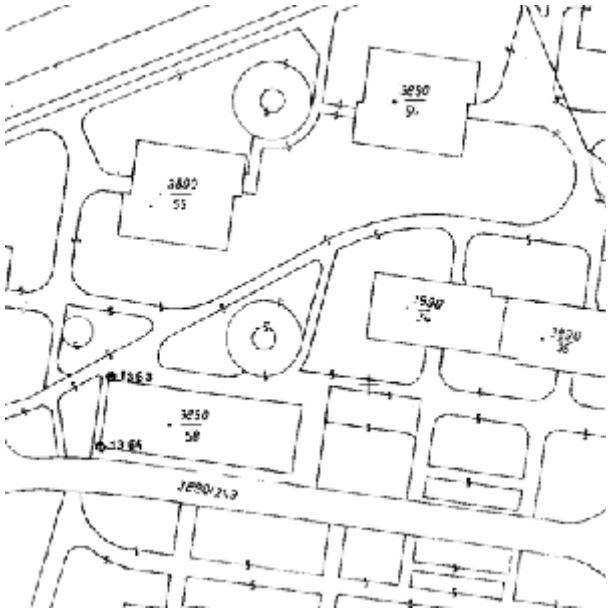
[4]

2.2.3 Rastrové zobrazení map

Rastr je obrazová „buňka“ libovolného tvaru obsahující informaci o barvě. V praxi je však nejpoužívanější čtvercová mřížka. Ta se osvědčila jako nejefektivnější a je označována pojmem pixel, který je tak nejmenší obrazovou jednotkou.

Rastrové zobrazení, též zvané jako „bitmapa“, je tvořeno vertikálně a horizontálně opakujícími se rastry, které obsahují vždy jeden odstín barvy. Bitmapy jsou hlavně využívány při pořizování digitálních fotografii, zobrazení na výstupním zařízení či při tisku.

Nevýhodou tohoto formátu je velká paměťová náročnost (při vysokém rozlišení a barevné hloubce), vzniká deformace při zmenšování, zvětšování a rotaci obrazu.



Obrázek 2 – Ukázka rastrového zobrazení mapy

Zdroj: [5]

2.2.4 Vektorové zobrazení map

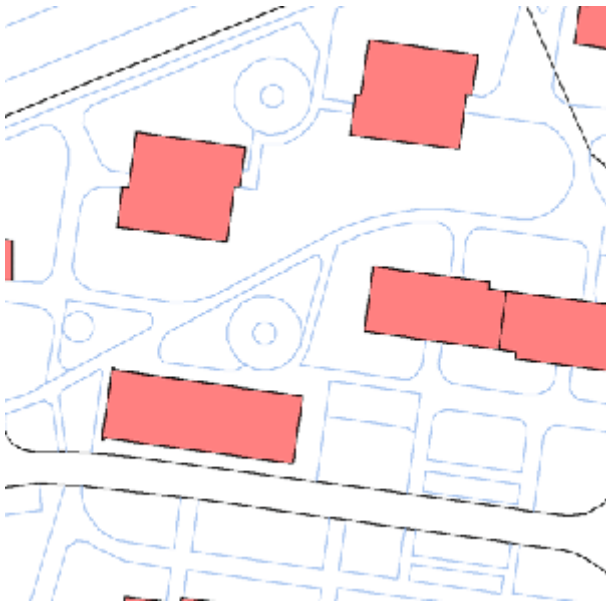
Vektor si lze představit jako přímku, která má určitý směr a velikost. Bývá definován počátečním a koncovým bodem.

Vektorové zobrazení, stejně jako bitmapové, je využíváno k zobrazování obrazových dat, s tím rozdílem, že neobsahuje informace o jednotlivých buňkách, ale soubor informací o vektorech. Jednotlivé body jsou pak dopočítávány a barva optimalizována distribučními algoritmy. Jelikož jsou vektory reprezentovány přímkami nebo body, jsou pak schopné zobrazit jakýkoliv tvar.

Výhodou vektorového zobrazení je, že při transformaci obrazu nedochází k deformacím ani ztrátě informací o obrazu.

Hlavní nevýhodou proti bitmapovému zobrazení je získání vektorových dat.

U vektorových map je třeba brát ohled na vzájemnou polohu objektů, zvlášť pokud by se měli překrývat.



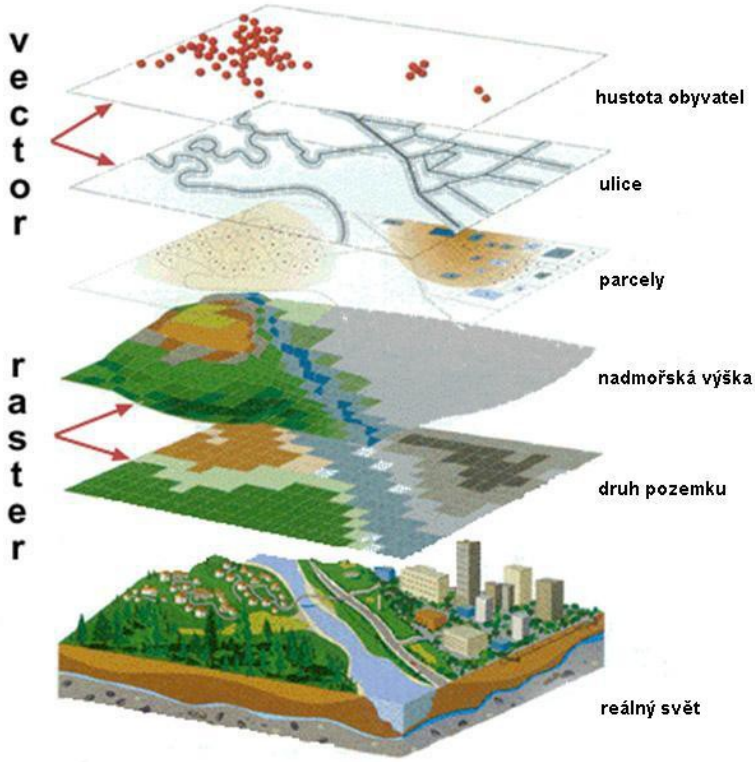
Obrázek 3 – Ukázka vektorového zobrazení mapy

Zdroj: [5]

2.2.5 Geodata

Geodata nebo také mapové podklady lze definovat jako soubor dat, která mají nějaký vztah k místu na Zemi a mohou obsahovat informace o územním celku nebo jeho součástech.

Kromě informací o povrchu, lze geodata propojovat i s dalšími soubory různého typu a získat tak nové informace o daném územním celku. Dodatečné informace mohou například obsahovat názvy ulic, zajímavá místa, poštovní směrovací čísla, dále plánování pozemkových celků, zdroje nerostných surovin, nadmořské výšky nebo dočasné informace o počasí.



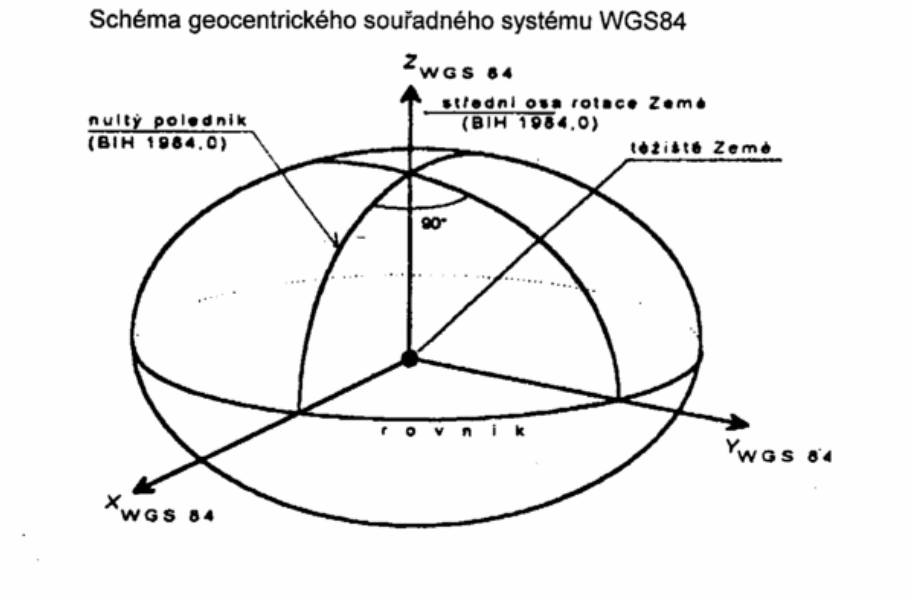
Zdroj: [6]

2.2.6 Souřadnicové systémy

Soustava souřadnic je soustavou základních údajů, jakou jsou body, přímky nebo křivky, umožňující určit polohu objektu v daném systému. Ve spojení s geografii je přesnějším pojmem označení zeměpisné souřadnice (koordináty), které slouží pro jednoznačné určení polohy na Zemi.

Následně budou popsány dva souřadnicové systémy. První systém WGS 84 je světově uznávaný geodetický standart, který byl schválen v roce 1984. Původně byl určen pro vojenské účely používaný státy NATO. Referenčními body systému jsou elipsoid a geoid, pro navigaci (GPS) a geodezii. Systém tvoří pravotočivá kartézská soustava se středem v těžišti Země. Kladná osa Z je totožná s osou rotace Země a směřuje k severnímu pólu. Osa X směřuje k průsečíku nultého poledníku a rovníku a osa y je kolmá na obě předchozí.

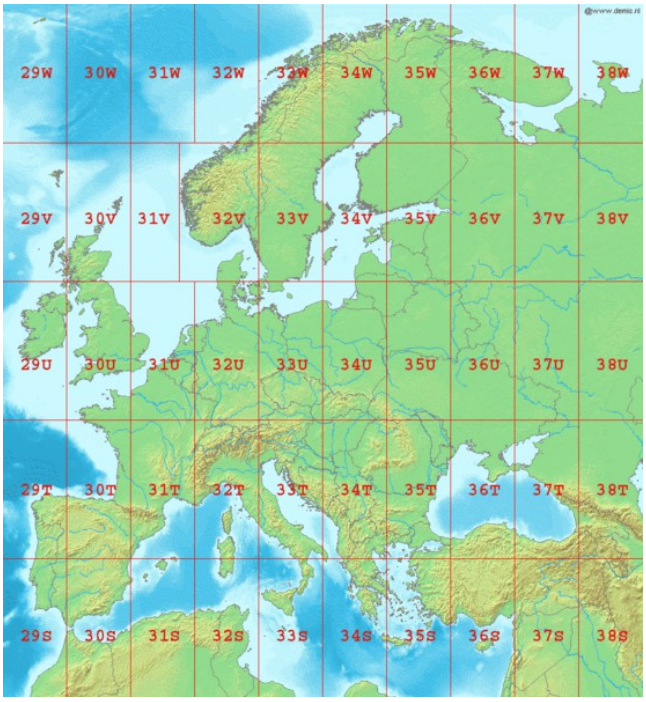
[7]



Obrázek 5 – Schéma souřadnicového systému WGS84

Zdroj: [7]

Univerzální Transverzální Mercatorův systém souřadnic (UTM) určuje polohu míst na Zemi pomocí sítě šedesáti zón. Jelikož se nejedná o zobrazení elipsoidu, jako u systému WGS 84, lze na mapách UTM měřit vzdálenost dvou bodů pomoci Pythagorovy věty, avšak pouze v případě, že body leží ve stejné zóně. Z toho plyne, že každá zóna má jiný střed souřadnic.

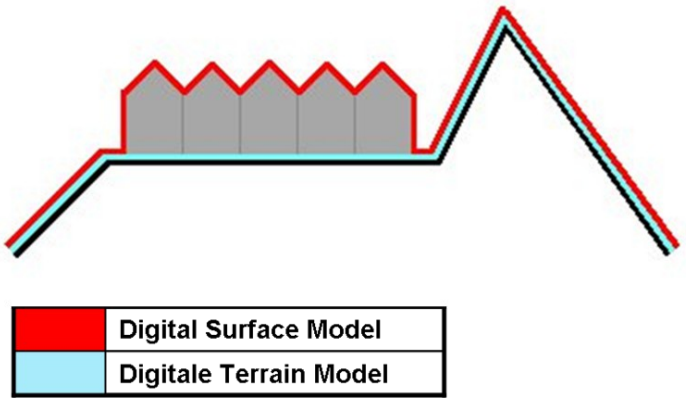


Obrázek 6 – Rozložení UTM zón v Evropě

Zdroj: [8]

3 Digitální výškové modely, projekt SRTM

Digitální výškový model představuje prostorové zobrazení povrchu Země v digitální podobě. Výškový model lze ještě rozdělit na terénní a povrchový model. Rozdíl velmi dobře zobrazuje následující obrázek, kde světle modrou čarou je zobrazen terénní model a červenou povrchový, ten navíc zahrnuje výšky staveb a vegetace. [16]



Obrázek 7 – Zobrazení rozdílu mezi povrchovým a terénním modelem

Zdroj: [9]

Existují tři způsoby jak vytvořit digitální výškový model.

- Přímé povrchové měření – pomocí speciálního snímače se zaměřují jednotlivé body. Většinou je toho využíváno při menších projektech, kde je potřeba vysoká přesnost zaměření.
- Digitalizace map – z fyzických map jsou čteny vrstevnice a ty jsou poté převedeny do digitálního modelu. Přesnost se může značně lišit, záleží jak na přesnosti fyzické mapy, tak na samotné digitalizaci.
- Fotogrammetrické metody – ty spočívají v získávání informací z fotografických snímků pořízených například při leteckém či satelitním snímání povrchu.

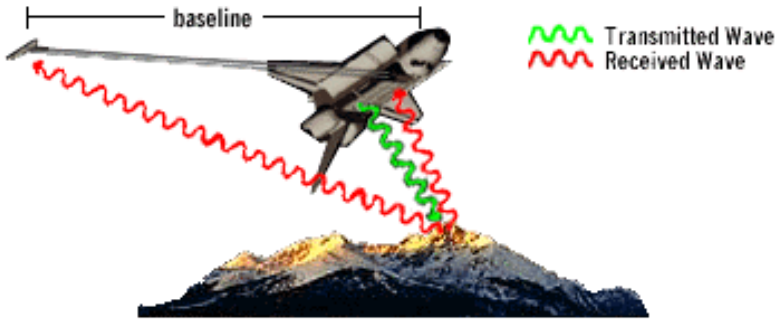
Digitální výškové modely lze využít například při tvorbě reliéfních map, fyzických modelů, 3D vizualizaci, při leteckých simulacích, povrchové analýze, archeologii, či v systémech GIS, GPS atd. [9]

V následujících podkapitolách bude podrobně popsán projekt SRTM, jakým způsobem data získali a formát SRTM souborů. V závěru poté stručně popsány další digitální výškové modely.

3.1 SRTM

SRTM (Shuttle Radar Topography Mission) představuje jednu z nejkompletnějších topografických databázi nadmořských výšek planety Země. Byl vytvořen v rámci mezinárodního projektu vedeného organizacemi NASA a NGA s přispěním německé a italské kosmické agentury.

V roce 1996 začíná plánování a o čtyři roky později 11. února 2000 startuje raketoplán Endeavour na jedenáctidenní misi, při které pomocí speciálně upraveného radarového zařízení byl nasnímán povrch Země. Tyto radary mají tu výhodu, že pracují na frekvenci, při které nezáleží na denní době ani na meteorologických podmínkách. Měřící zařízení se skládalo ze dvou antén, jedné umístěné v nákladovém prostoru raketoplánu a druhé ve volném prostoru upevněné ve vzdálenosti 60 metrů na zdvihovém zařízení.

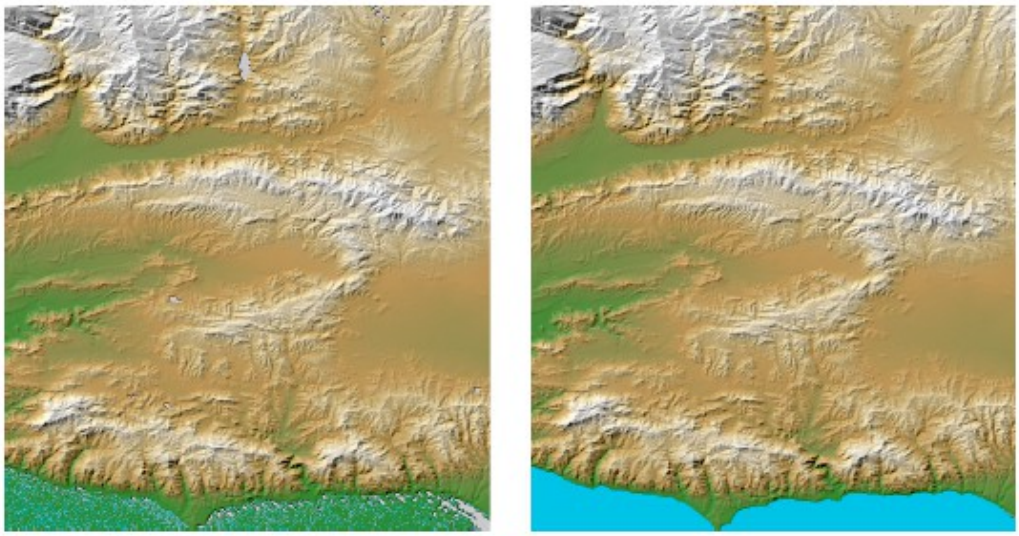


Obrázek 8 – Umístění antén na raketoplánu

Zdroj: [10]

Hlavním cílem projektu bylo vytvoření digitálních topografických dat pro 80% zemského povrchu (veškerá pevnina mezi 60° severní a 56° jižní šířky). Datové body leží v rozmezí 3" (přibližně 90 metrů na rovníku) zeměpisné šířky/délky. V oblasti USA leží datové body v rozmezí 1" (30 metrů na rovníku). Přesnost naměřených nadmořských výšek se pohybuje v rozmezí 16 metrů.

Soubory SRTM jsou k dispozici ve dvou verzích. Verze 1 představuje data, která jsou výsledkem zpracování nasnímaných dat z radarů. Verze 2 zahrnuje opravy problémových oblastí jako vodní plochy, pobřežní oblasti či horské oblasti s velkým sklonem a další.



Obrázek 9 – Porovnání verzí SRTM souborů

Zdroj: [12]

[10][11]

3.2 Formát SRTM souborů

SRTM data jsou k dispozici ve dvou úrovních:

- SRTM1 pro USA a jeho teritoria, kde jsou vzorky v 1" na intervalu šířky a délky.
- SRTM3 pro celý svět, kde je vzorkování po 3".

Data jsou rozděleny po 1° zeměpisné šířky a délky do souborů představující rastrové zobrazení pro dané území. Názvy souborů reprezentují zeměpisnou šířku a délku rastru v levém dolním rohu – například soubor N37W105 má levý dolní roh na 37° severní šířky 105° západní délky.

Výškové soubory mají příponu .HGT a jsou distribuovány v balíčcích ZIP. Soubory mají jednoduchou binární podobu. Každou výšku reprezentuje celé číslo o velikosti dvou bajtů. Bajty jsou v souboru uloženy ve formátu „big-endian“, nejvýznamnější bajt je na prvním místě, přímo čitelné na systémech Motorola, Sun SPARC, počítače Macintosh s procesory PowerPC. Pro většinu současných počítačů postavených na architektuře x86 používající „little-endian“, kde je nejvýznamnější bajt na posledním místě a tedy může být potřeba jednotlivé bajty prohodit.

Jednotlivé výšky jsou v metrech. Problémová místa mají nastavenou výšku na -32768.

Soubory SRTM3 obsahují 144201 výšek v 1201 řádcích a 1201 sloupcích. Řádky na severním a jižním okraji, stejně tak východní a západní krajní sloupce se překrývají se sousedními rastry (soubory). Soubory SRTM1 obsahují 12967201 výšek v 3601 řádcích a 3601 sloupcích s podobným překrytím.

[12]

3.3 Aplikace využívající SRTM databázi

Databáze SRTM je přístupná široké veřejnosti zdarma k využívání. Zde je uvedeno několik aplikací, které SRTM data používají.

3.3.1 Maps-For-Free[1]

Online webová aplikace Maps-For-Free zobrazuje plastické mapy spolu s množstvím dalších mapových vrstev integrovaných do existujících Google map. Kromě základních funkcí, Google map umožňuje výběrem z rolovacího menu přímé zobrazení hlavních měst, států, sopek a dalších míst.

3.3.2 MATLAB

MATLAB je interaktivní programové prostředí umožňující počítání s maticemi, vykreslování 2D a 3D grafů funkcí, počítačovou simulaci, analýzu a prezentaci dat a mnoho dalšího. Samostatný MATLAB se SRTM soubory pracovat neumí, avšak existují knihovny, které umožňují import a manipulaci s těmito soubory.

Například Matlab SRTM Library[2].

3.4 Další digitální výškové modely

3.4.1 ASTER GDEM

ASTER GDEM (Global DEM) je stejně jako SRTM globální výškový model, který vznikl ve spolupráci NASA a japonského Ministerstva ekonomie, obchodu a průmyslu.

Tento model je pojmenován po multispektrálním barevném skeneru ASTER, který je umístěn na družici Terra. Pořizuje data ve čtrnácti pásmech od viditelného až po infračervené záření s povrchovou přesností až 15 metrů. První verze souborů byla zdarma zpřístupněna v roce 2009.

ASTER GDEM v porovnání se starším SRTM3 pokrývá pevninský povrch mezi 83° severní a 83° jižní šířky a také dosahuje vyšší přesnosti. Datové body jsou v rozmezí 1" (přibližně 30 metrů).

Stejně jako SRTM jsou ASTER GDEM rastery rozděleny po 1° s překryvem jednoho pixelu se sousedními rastry. Velikost rastru je 3601x3601 pixelů. Každý rastr (soubor) nese označení zeměpisných souřadnic odpovídající levému dolnímu rohu.

[13]

1 www: http://www.maps-for-free.com/
2 www: http://srtm-matlab.sourceforge.net/

3.4.2 WorlDEM

V roce 2014 by měl být k dispozici nový globální digitální výškový model od Evropské společnosti Astrium. Snímání je prováděno německými satelity s vysokým rozlišením TerraSAR-X a TanDEM-X a bude pokrývat veškerý pevninský povrch i se severním a jižním pólem.

Mezi hlavní přednosti tohoto modelu patří velmi vysoká kvalita a přesnost snímání. Vzdálenost jednotlivých výškových bodu je 12 metrů s vertikální přesností 2 metry (relativní) a 10 metrů (absolutní).

[14]

3.4.3 Digitální modely ČR

V České republice se zabývá tvorbou digitálních terénních modelů několik firem a organizací. Mezi hlavní patří armáda ČR (VTOPÚ Dobruška), Český úřad zeměměřičský a katastrální (ČUZK), dále například T-Mapy spol. s r.o.

Vojenský topografický ústav v Dobrušce poskytuje svá data všem složkám armády ČR, také i civilním zájemcům. Mezi nabízené modely patří DMR-2, který byl vytvořen ruční vektorizací vrstevnic na mapách v měřítku 1:25 000. Vzdálenost výškových bodů je 100x100 metrů, přesnost výšek se pohybuje v rozmezí 3 až 15 metrů. Pokrývá území České a Slovenské republiky. Ve vývoji je model DMR-3 s předpokládanou vzdáleností bodů 50x50 metrů.

Český úřad zeměměřičský a katastrální poskytuje digitální model ZABAGED, který je odvozen z mapy ČR v měřítku 1:10 000 s výškovou přesností 3 až 5 metrů. Tento model má charakter GIS obsahující vektorové složky s topografickými relacemi objektů a dalšími informacemi o objektech.

Společnost T-Mapy spol. s r.o. má v nabídce digitální vektorovou databázi vrstevnic v měřítku 1:50 000 obsahující vrstevnice s konstantní vzdáleností 10 metrů, výškové kóty a stínovaný reliéf.

[15]

4 Použité technologie

V této kapitole budou představeny technologie, které jsou využity hlavně pro vytvoření praktické části. Volba programovacího jazyku, nadstaveb a vývojového prostředí, až po představení technologie zvolené pro grafické zobrazení zdrojových dat.

V závěru kapitoly jsou také stručně zmíněny alternativy, které by byly vhodné pro vývoj aplikace.

4.1 C++

Jazyk C++ je multiplatformní objektově orientovaný programovací jazyk, který vznikl jako rozšíření jazyka C v Bell Laboratories, kde ho počátkem 80. let vyvinul Bjarne Stroustrup. C++ není čistě objektovým jazykem, kromě objektově orientovaného programování, podporuje i proceduální a generické programování aplikací.

Původní verze jazyka byla označována jako „C with Classes“ (česky C s třídami). Až v roce 1983 Rick Mascitti vymyslel jméno „C++“, kde „++“ zdůrazňuje rozšíření jazyka C („++“ je operátor inkrementace v C).

C++ v mnohých věcech ovlivnil vývoj ostatních programovacích jazyků, které jsou v současnosti velmi využívané jako například jazyky C#, Java či PHP. S trochou nadsázky můžeme prohlásit, že jde o rozšíření jazyka C. Rozšíření o možnost použití tříd, dědění, přetěžování operátorů, volání virtuálních funkcí, zachytávání vyjímek a dalších.

V současné době patří C++ mezi nejrozšířenější programovací jazyky využívané pro programování ovladačů zařízení, operačních systémů, aplikačního softwaru, výkonných klientských a serverových aplikací až po vývoj her v herním průmyslu.

[16]

Pro realizaci bakalářské práce byl použit jazyk C++ zejména z důvodu snadnější implementace funkcí grafického rozhraní OpenGL, podpoře Qt frameworku, který poskytuje jednoduchou tvorbu grafického uživatelského rozhraní. V neposlední řadě také díky poměrně kvalitní dokumentaci.

4.2 Qt

Qt je jeden z nejpopulárnějších multiplatformních frameworků pro tvorbu aplikací s grafickým uživatelským rozhraním. Aplikace využívající Qt se převážně píší v programovacím jazyce C++, ve kterém je Qt taky vytvořeno. Dne 3. července 2013 byla vydána zatím poslední finální verze Qt 5.1. [17]

Framework lze chápat jako soubor knihoven, nadstavbu určité platformy, kde očekává určité vlastnosti, které využívá ve svých funkcích a přidává obsažené prvky ve vlastních knihovnách.

Přestože Qt nativně podporuje jazyk C++, existuje i pro programovací jazyky Python (PyQt), Ruby (QtRuby), C, Perl, Pascal, C#, Java. [17]

4.2.1 Historie

První veřejně dostupná verze Qt frameworku se objevila v květnu 1995. Na vývoji se podíleli vývojáři Haavard Nord (prezident společnosti Trolltech) a Eirik Chambe-Eng. Haavard s Eirikem se setkali při studiu oboru počítačových věd na norském institutu technologií v Trondheimu.

Počátky Qt frameworku sahají až do roku 1988 kdy byl Haavard pověřen rozvojem GUI pro jazyk C++. V létě 1990, Haavard s Eirikem pracují na C++ databázové aplikaci, která má běžet v GUI na operačních systémech Windows, Unix a Macintosh. V této době Haavard začíná uvažovat o vytvoření vlastního grafického frameworku pro C++.

Již v roce 1991 začíná Haavard psát třídy, ze kterých se později stalo Qt, se spoluprací Eirika na designu. O rok později Eirik přichází s nápadem signálů a slotů, v roce 1993 vyvinuli první Qt grafické jádro, které jim umožnilo implementaci vlastních widgetů. Koncem tohoto roku začínají společně podnikat a věnovat se vývoji C++ GUI Framework.

V květnu 1995 byla tedy vydána první veřejná verze Qt 0.90 určená pro vývoj aplikací pod Windows a Unix. V tuto dobu bylo Qt dostupné pod komerční GNL licencí.

V roce 1998 začal vývoj grafického prostředí KDE pro GNU/Linux a Qt byla zvolena jako hlavní knihovna pro implementaci.

Trolltech zajišťuje vývoj až do roku 2008, kdy společnost kupuje Nokia, ta se stará o vývoj až do podzimu 2012. V současnosti se o vývoj stará společnost Digia, která Qt koupila od Nokie.

[18]

4.2.2 Licence

Qt framework je nabízen ve třech možných licenčních modelech.

1. **Komerční licence** – po zakoupení licence lze vyvíjet aplikace i modifikovat Qt framework bez povinnosti zveřejnit zdrojové kódy či platit další poplatky z prodaných aplikací.
2. **Licence GPL v2.1** – možnost zdarma vyvíjet nekomerční aplikace, je zde však povinnost zveřejňování zdrojových kódů.
3. **Licence LGPL v3.0** – poslední licence umožňuje komerční vývoj aplikací, kdy jsou vývojáři povinni zveřejňovat pouze změny v kódu samotného Qt frameworku.

[18]

4.2.3 Podporované platformy

Mezi operační systémy podporované na desktopových stanicích patří:

- Windows – Microsoft Windows XP, Vista a 7
- Linux – X Window System 32bit a 64bit (Linux, HP-UX, Solaris, AIX…)
- OS X – Apple Mac OS X 10.6 „Snow Leopard“ a Apple Mac OS X 10.5 „Leopard“

Od verze Qt 5 byla rozšířena podpora mobilních platforem na Android, iOS, Windows Phone 8, Windows RT a Blackberry 10.

V nejnovějších verzích Qt přestávají být podporovány málo používané platformy Maemo, Symbian a Widnows CE 5.0 díky tomu není zaručena plná podpora.

[17]

4.2.4 Signály a sloty

Signály a sloty se využívají ke komunikaci mezi objekty tříd, které jsou přímo nebo nepřímo zděděny ze třídy QObject. Jsou hlavním mechanismem Qt frameworku a pravděpodobně je to ta část, díky které se odlišují od jiných frameworků.

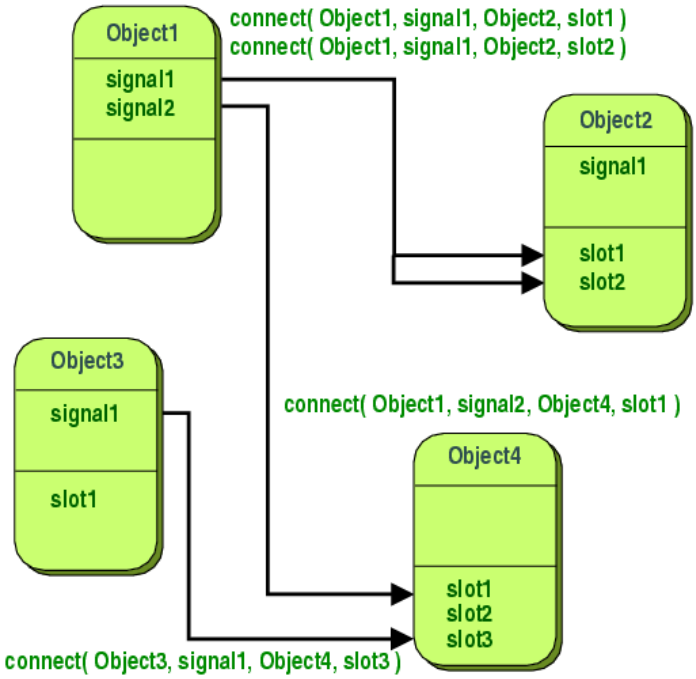
Při programování GUI aplikací, když chceme změnit jeden objekt třídy, tak většinou chceme, aby se tato změna projevila také někde jinde. Například, když dojde ke stisknutí tlačítka „Otevřít“, očekává se, že se otevře nějaké dialogové okno.

Při propojování signálů a slotů může být spojeno více signálů s jedním slotem a stejně tak, jeden signál může být napojen na více slotů. Slot může být vyvolán po přijetí signálu a také použit jako standardní funkce objektu. Lze také propojit signál se signálem, díky tomu se při vyvolání signálu vyvolá signál jiný.

Propojení signálu se slotem se provádí pomocí následující funkce:

```
connect(Objcet1, SIGNAL(signal()), Object2, SLOT(slot()));
```

[19]



Obrázek 10 – Signály a sloty

Zdroj: [19]

4.3 Qt Creator

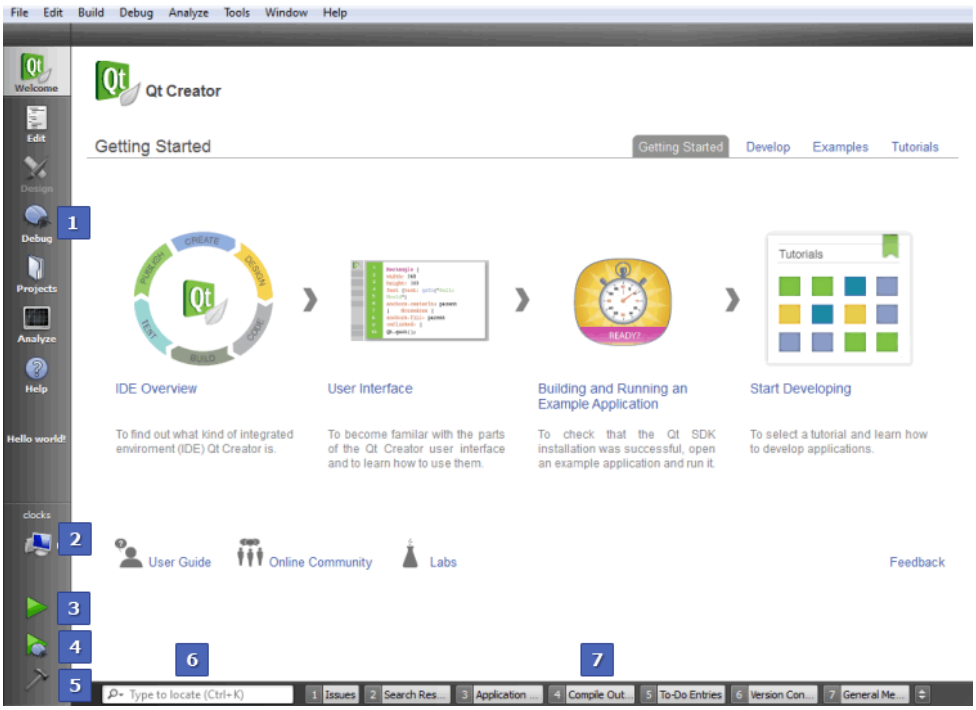
Qt Creator je multiplatformní vývojové prostředí (IDE) vytvořené pro práci vývojářů s Qt frameworkem. Qt Creator je stejně jako Qt framework vyvíjen společností Digia.

Qt Creator je zaměřen na poskytování funkcí, které pomáhají novým uživatelům Qt se rychleji zdokonalovat a také zvýšit produktivitu zkušených vývojářů.

Základní vlastnosti Qt Creatoru:

- Editor kódu v C++, podpora QML a ECMAscript
- Nástroje pro rychlou navigaci ve zdrojovém kódu
- Zvýrazňování syntaxe a doplňování kódu
- Podpora refaktoringu zdrojového kódu
- Kontextové nápověda
- Kontrola a zvýraznění závorek stejné úrovně

[20]



Obrázek 11 – Uživatelské rozhraní Qt Creatoru

Zdroj: [21]

Volba režimů (1) Qt Creatoru (Uvítací obrazovka, editor, návrhář, ladění, nastavení a analyzování projektu, poslední režim nápovědy).

Nastavení typu sestavení (2) pro spuštění (3), ladění (4), nebo sestavení (5) aplikace. Výstup z těchto akcí se zobrazuje ve výstupním panelu.

Pole (6) pro vyhledávání v projektech, souborech, třídách, dokumentaci atd.

[21]

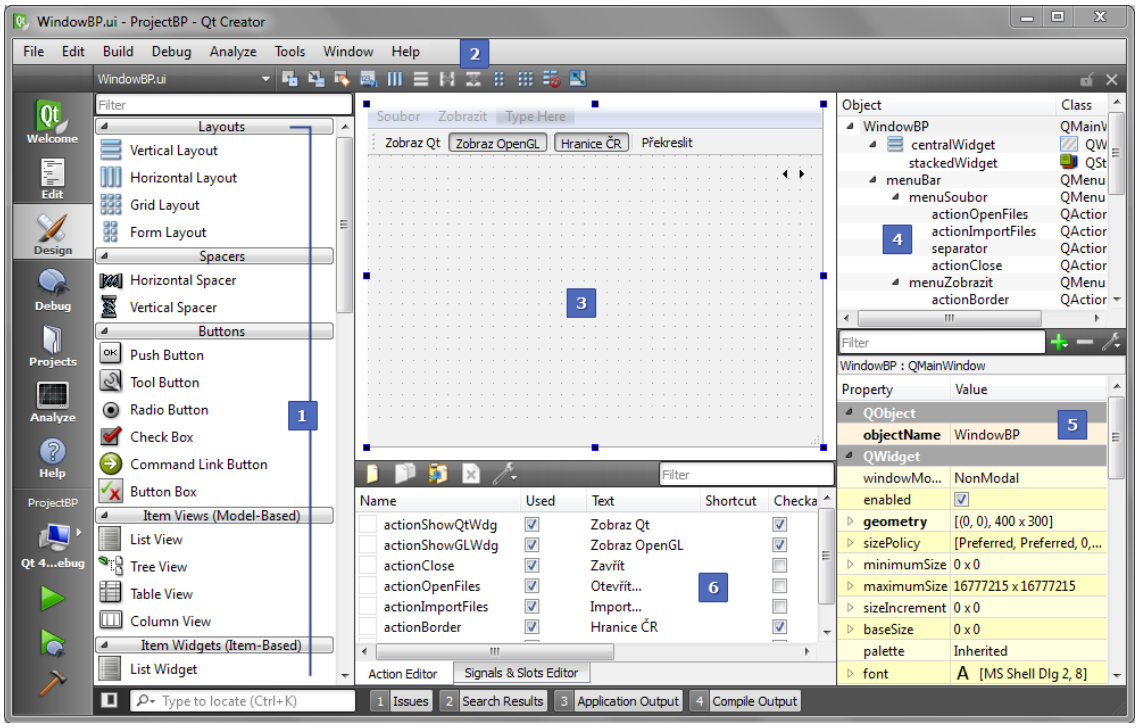
4.3.1 Qt Designer

Qt Designer je součástí Qt Creatoru a slouží pro snadnou a rychlou tvorbu grafického uživatelského rozhraní (GUI). GUI může být napsáno ručně pomocí zdrojového kódu, ale právě Qt Designer tuto práci usnadňuje možností prostým přetažením widgetu do okna.

Widgety jsou vizuální prvky uživatelského rozhraní jako například tlačítka, menu, posuvníky, vkládací pole a další. U každého widgetu lze upravovat různé vlastnosti a začleňovat je do přehledného rozložení (layoutu).

Dle potřeby lze jednoduše graficky nastavit propojení signálů a slotů.

Návrh v Qt Designeru je reprezentován souborem *.ui, který lze s projektem převést do jazyka C++ a poté zkompilovat ve výslednou aplikaci. Zdrojový kód *.ui souboru je psán v XML, ten však nelze přímo editovat.



Obrázek 12 – Rozhraní Qt Designeru

Zdroj: autor

Základní widgety (1) pro tvorbu grafického rozhraní. Nastavení rozložení (2) widgetů v okně aplikace. Plátno návrháře (3), na které rozmísťují potřebné widgety.

Hierarchie (4) widgetů v návrhu. Úprava nastavení (5) právě označeného widgetu.

Seznam abstraktních akcí (6), které lze vkládat na widgety. Například jako položky menu či nástrojové lišty.

4.4 OpenGL

Knihovna OpenGL byla navržena firmou SGI (Silicon Graphics Inc.) v roce 1992 jako nízkoúrovňové aplikační programové rozhraní (API) pro práci s trojrozměrnou grafikou na akcelerovaných grafických kartách.

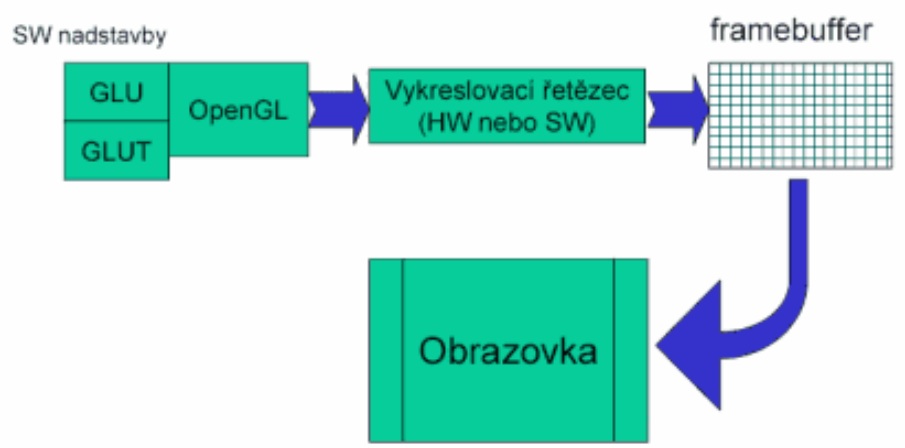
Při návrhu knihovny OpenGL byl kladen důraz, aby byla použitelná na různých grafických akcelerátorech, různých platformách, ale také na systémech, které grafický akcelerátor neobsahují – v tomto případě se využívá softwarová simulace.

V současnosti OpengGL podporuje všechny desktopové platformy: Windows, Linux, Max OS X. Stejně tak výrobci (AMD/ATI, Intel, Nvidia, S3graphic) grafických akcelerátorů začleňují podporu OpenGL do svých čipů.

Primárně jsou OpenGL hlavičkové soubory vyvíjeny pro práci s jazyky C a C++. Existují však další podobné soubory pro práci v jiných jazycích, například pro C#, Java (JOGL), Perl (POGL), Python (PyOpenGL).

Na některých platformách lze aplikaci rozdělit na serverovou a klientskou část, poté se příkazy pro vykreslování přenášejí přes síťové rozhraní. Jelikož je knihovna OpenGL nezávislá na operačním systému neobsahuje ani žádné funkce pro práci s okny pro vytváření grafického uživatelského rozhraní. Toto lze zajistit přímým voláním funkcí okenního správce vázaného na daný operační systém, nebo využití nadstaveb jako GLUT či výše popsaný Qt framework.

Postup vykreslení obrazové scény je zobrazeno na Obrázku 13. Funkce OpenGL vykreslí rastrový obrázek, ten je uložen do framebufferu, kde je každému pixelu přiřazena barva, hloubka, alfa složka případně další atributy. Z framebufferu se následně získá pouze barevná informace a ta je zobrazena na obrazovce.



Obrázek 13 – Postup vykreslení obrazové scény v OpenGL

Zdroj: [22]

Přestože je knihovna OpenGL multiplatformní, nezaručuje totožné zobrazení stejného programu na různých platformách. Způsobuje to například odlišná přesnost reprezentace čísel na grafické kartě, různé algoritmy pro počítání barev, texturových souřadnic nebo jen jinou bitovou hloubkou z-bufferu. Avšak by mělo být zachováno geometrické a barevné podání scény.

[22]

4.4.1 Vertex buffer

Také označován jako Vertex Buffer Objects (VBO) lze volně přeložit jako buffer pro objekty vrcholů. Hlavní rozdíl oproti předchozí metodě vykreslování je v tom, že se data ukládají přímo do paměti grafické karty. Při překreslení scény tedy není potřeba znova přenášet data z operační paměti do paměti grafické. Má to však nevýhodu, že vertex buffer musí podporovat grafická karta.

Vykreslení do VBO se skládá z pěti kroků:

1. Vygenerování názvu bufferu.

   ```
   glGenBuffers(GLsizei n, GLuint * buffers)
   ```

2. Aktivace bufferu.

   ```
   glBindBuffer(GLenum target, GLuint buffer)
   ```

3. Uložení dat do bufferu.

   ```
   glBufferData(GLenum target, GLsizeiptr size,
                const GLvoid * data, GLenum usage)
   ```

4. Vykreslení dat v bufferu.

   ```
   glDrawArrays(GLenum mode, GLint first, GLsizei count)
   ```

5. Zrušení bufferu z paměti.

   ```
   glDeleteBuffers(GLsizei n, const GLuint * buffers)
   ```

4.5 Alternativní technologie a jazyky

Jako alternativa k jazyku C++ pro vývoj programu v praktické části byla zvažována Java spolu s knihovnou Swing pro tvoru GUI a knihovna JOGL pro vazbu na OpenGL. Pro Javu bylo zvažované vývojové prostředí NetBeans, nebo JDeveloper.

Pro zobrazování zdrojových dat pomocí grafické karty je možné použít Direxť3D. Direct3D je součástí balíku knihoven DirectX. Jedná se o specializované rozhraní (API), které nabízí množství funkcí pro práci s 3D grafikou. DirectX vyvíjí společnost Microsoft a je tedy dostupné pouze pro systémy Windows.

5 Implementace programové části

Zadáním bakalářské páce bylo vytvořit aplikaci, která zpracuje výškové soubory projektu SRTM do podoby vhodné pro následné zobrazení výškového povrchu. Zobrazení bude prováděno pomocí grafického rozhraní OpenGL s využitím vertex bufferu a pro druhý způsob byla zvolena třída QPainter knihovny Qt.

Aplikace je napsána v C++ s použitím Qt frameworku. Vývoj byl proveden v prostředí Qt Creatoru a grafické uživatelské rozhraní bylo navrženo v Qt Designeru.

Oblasti pro zobrazení povrchu v aplikaci bude České republika spolu s částmi sousedních států. Pro lepší orientaci budou navíc zobrazeny hranice ČR.

Cílem aplikace je porovnat rychlost zobrazování pomocí jednotlivých způsobů zobrazení.

5.1 Souhrn vlastností aplikace

Zde jsou v bodech uvedeny možnosti vytvořené aplikace:

- Načtení a zpracování *.hgt souborů do formátu vhodného pro zobrazování, před zpracováním je možné omezit počet výškových bodů.
- Načítání zpracovaných *.hgt souborů pro zobrazení.
- Přepínání panelů, na kterých je zobrazen povrch pomocí OpenGL s využitím vertex bufferu a QPainter knihovny Qt.
- Vypnutí/zapnutí zobrazení hranic ČR.
- Možnost libovolného pohybu, přiblížení a oddálení povrchu.
- Manuální znovu zobrazení (překreslení) aktuálně vybraného panelu zobrazení.
- Dialogové okno zobrazující průběh déle trvajících událostí (zpracování souborů, překreslování)

5.2 Popis uživatelského rozhraní

Grafické uživatelské rozhraní je jednoduše navrženo, aby se v něm potencionální uživatelé rychle zorientovali. Při používání aplikace je možné se setkat se třemi okny, které jsou podrobněji popsány v následujících částech.

5.2.1 Hlavní okno aplikace (třída WindowBP)

Po spuštění aplikace se uživateli zobrazí okno ukázané na následujícím obrázku č. 14. Z tohoto okna jsou ovládány veškeré hlavní funkce aplikace a hlavně největší plochu zabírající prostor (1) pro zobrazování povrchu České republiky. Aktuálně je zobrazena pouze hranice ČR bez jakýchkoliv mapových podkladů.

Menu (2) *Soubor* obsahuje tři položky. Položka *Otevřít* slouží pro načtení zpracovaných .hgt souborů, *Import* pro výběr a zpracování .hgt souborů a položka *Konec* zavírá aplikaci. Menu (2) *Zobrazit* obsahuje stejné položky co jsou na nástrojové liště.

Nástrojová lišta obsahuje dvě tlačítka pro přepínání panelů zobrazení (3), tlačítko pro zapnutí zobrazení hranic ČR (4). Poslední tlačítko slouží pro manuální překreslení (5) aktuálního zobrazení.

Obrázek 14 – Hlavní okno aplikace

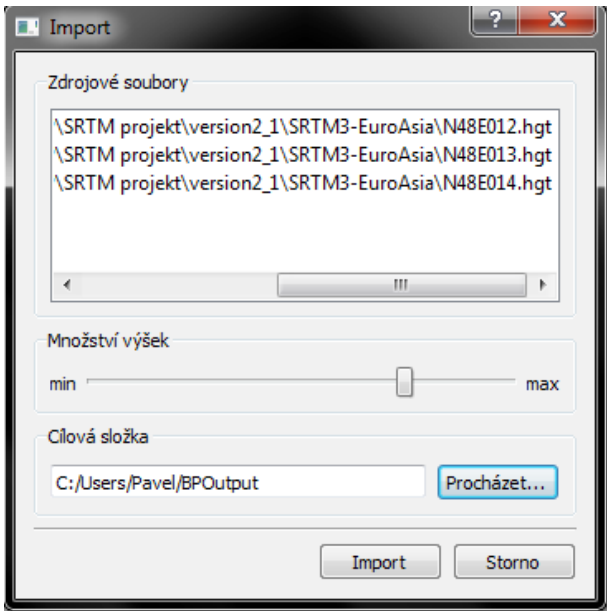
Zdroj: autor

5.2.2 Dialogové okno s nastavením importu (třída ImportDlg)

Toto dialogové okno se otevírá po výběru výškových .hgt souborů z menu Soubor→Import. Okno je svisle rozděleno na tři části:

- První část představuje seznam vybraných souborů.
- Druhá umožňuje omezit množství výšek, které se budou zpracovávat. Kde *max* představuje využití všech výšek z .hgt souboru a *min* využití každé páté výšky. Při zobrazení celé ČR je dostačující nastavení *min*.
- Poslední slouží pro výběr složky, kam se uloží vytvořené soubory.

Při potvrzení tlačítkem *Import* se otevře dialogové okno zobrazující průběh zpracování .hgt souborů. V opačném případě při stisknutí tlačítka *Storno*, nebo „křížku“ se import souborů přeruší.

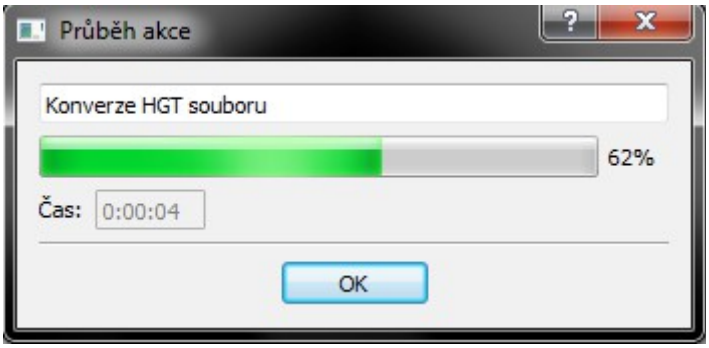


Obrázek 15 – Dialogové okno importu .hgt souborů

Zdroj: autor

5.2.3 Dialogové okno probíhajících událostí (třída ProgressDlg)

Velmi jednoduché dialogové okno zobrazující jaká událost aktuálně probíhá, její průběh a čas trvání. V případě úspěšného dokončení se vedle času zobrazí text „*Dokončeno!*“



Obrázek 16 – Dialogové okno probíhajících událostí

Zdroj: autor

5.3 Popis tříd

5.3.1 WindowBP

Tato třída je základem celé aplikace. Obsahuje tří hlavní datové položky, které jsou důležité pro fungování celé aplikace. Jedná se o objekty tříd DrawQtWdg, DrawGLWdg a ConvertHgt. První dva objekty se starají o zobrazování výškových dat a třetí o zpracování .hgt souborů.

Dále má tato třída tři hlavní úkoly. Prvním úkolem je přepínání panelů zobrazení, pokud to aplikace vyžaduje. Panely jsou uloženy ve třídě QStackedWidget, na indexu 0 je panel pro kreslení v Qt a na indexu 1 v OpenGL.

```
if (pact == ui->actionShowQtWdg) {
    pstabar->showMessage(QString("Zobrazeno kreslení pomocí Qt"));
    ui->stackedWidget->setCurrentIndex(0);
} else if (pact == ui->actionShowGLWdg) {
    pstabar->showMessage(QString("Zobrazeno kreslení pomocí OpenGL"));
    ui->stackedWidget->setCurrentIndex(1);
} else { return; }
```

Na základě proměnné `pact` se rozhoduje, který panel se má zobrazit. Zároveň je o této činnosti uživatel informován zobrazením textu na status bar (`pstabar`).

Druhým úkolem této třídy je zařídit zpracování .hgt souborů. Přesněji řečeno načíst seznam těchto souborů třídou QFileDialog, předání seznamu třídě ImportDlg, která si od uživatele vyžádá další údaje potřebné pro zpracování souborů. Získané údaje jsou předány třídě ConvertHgt, která se již postará o samotné zpracování.

```
QStringList nameFiles =
    QFileDialog::getOpenFileNames(this, "Vyberte soubory pro import",
                                  Qdir::rootPath(),
                                  "Výšková data (*.hgt)");
```

Do proměnné `nameFiles` se uloží seznam cest k souborům, získané funkcí `getOpenFileNames()`. Ve vyvolaném dialogovém okně jsou zobrazovány pouze soubory s příponou hgt.

```
ImportDlg import(nameFiles, this);
import.exec();
```

Tento seznam je předán třídě ImportDlg představující dialogové okno, které je hned nato zobrazeno.

```
QString dst;  int q;
import.getParams(nameFiles, dst, q);
m_conv.setParams(nameFiles, dst, q);
m_conv.start();
```

Po stisknutí tlačítka *Import* se provede předání nastavení třídě ConvertHgt (proměnná `m_conv`). Proměnná `dst` obsahuje cestu ke složce, kam se bude ukládat a `q` množství výšek. Celý proces zpracování .hgt souborů se spustí funkcí `start()`.

Posledním úkolem této třídy je načítání zpracovaných .hgt souborů, tyto soubory mají příponu .chgt. Seznam těchto souborů je poté předán třídám DrawQtWdg, DrawGLWdg (proměnné `m_QtWdg` a `m_GLWdg`), které si tyto soubory načtou a zpracují dle svých potřeb pro následné zobrazení.

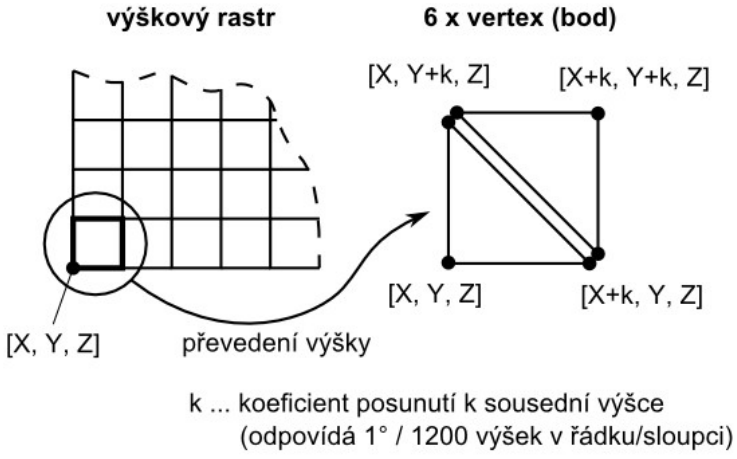
```
QStringList srcFiles =
    QFileDialog::getOpenFileNames(this, "Otevřít", QDir::homePath(),
                                  "Mapové podklady (*.chgt)");
if (srcFiles.isEmpty() == false) {
    m_QtWdg.createHeightsBuffer(srcFiles);
    m_GLWdg.createHeightsGenBuffer(srcFiles);
}
```

K třídě WindowBP je připojen formulářový soubor WindowBP.ui, ve kterém je uloženo kompletní uživatelské rozhraní aplikace. Rozmístěný komponent, jejich nastavení a případné propojení signálů se sloty komponent.

5.3.2 ConvertHgt

Tato třída se stará o kompletní převedení zdrojových .hgt souboru do formátu vhodného ke kreslení, výsledné soubory jsou poté ukládány s příponou .chgt.

Výškové soubory .hgt projektu SRTM obsahují pouze údaje o výškách. Ke každé výšce lze dopočítat její souřadnici z názvu souboru. To znamená, že o každém výškovém bodu lze zjistit souřadnici X, Y a Z představující výšku, na základě výšky bude danému bodu přiřazena barva (například 0-200 metrů sv. zelená, 200-300 metrů zelená až 1400-1800 metrů hnědá). Aby při kreslení nevznikali mezery, je každý výškový bod převeden na dva trojúhelníky, šest nových bodů, které zajistí „propojení“ se sousedními výškovými body. Dva trojúhelníky z toho důvodu, že funkce OpenGL glDrawArrays(), která vykreslí celé datové pole je schopna kreslit pouze body, čáry a trojúhelníky, nikoliv však ty vhodnější a to čtyřúhelníky.



Obrázek 17 – Převod výšky na 6 nových bodů

Zdroj: autor

Podobu výsledného datového pole pro jednu výšku z .hgt souboru znázorňuje následující tabulka, kde každý řádek představuje jeden bod. Každá buňka představuje reálné číslo o velikosti čtyř bajtů.

Tabulka 1 – Datové pole pro jednu výšku

X	Y	Z	R	G	B
X+k	Y	Z	R	G	B
X	Y+k	Z	R	G	B
X	Y+k	Z	R	G	B
X+k	Y+k	Z	R	G	B
X+k	Y	Z	R	G	B

Samotná třída ConvertHgt je potomkem třídy QThread, což umožňuje po reimplementaci funkce `run()` její spouštění v samostatném vláknu. Díky tomu není zatěžováno hlavní vlákno aplikace a tedy při zpracování souborů aplikace „nezamrzá“.

Vstupními parametry této třídy jsou zdrojové soubory, jejich seznam, složka kam se budou ukládat .chgt soubory a také koeficient určující množství výšek ve výsledném souboru.

Průběh zpracování jednoho souboru probíhá ve čtyřech krocích. Prvním krokem je testování, jestli soubor existuje a jestli v názvu souboru není chyba. Pokud jsou tyto podmínky splněny uloží se zeměpisné souřadnice z názvu do proměnných `latN` a `lngE`.

```
QString sSrc = m_srcFiles.at(i);
if (!QFile::exists(sSrc))     // existence souboru
    continue;

QString sName = sSrc.right(11);
if (!sName.contains(QRegExp("N[0-9]{2}E0[0-9]{2}.hgt")))  // test nazvu
     continue;
int latN = sName.mid(1,2).toInt();
int lngE = sName.mid(4,3).toInt();
```

V druhém kroku se provede načtení celého souboru do neznaménkového znakového pole. O čtení ze souboru se postará třída QDataStream, která v Qt slouží pro čtení a zápis do binárních souborů.

```
QDataStream in(&fSrc);
unsigned char *pSrcData = new unsigned char[fSrc.size()];
if(in.readRawData((char*) pSrcData, fSrc.size()) != fSrc.size()) {
    delete pSrcData;
    continue;
}
```

V třetím kroku je prováděna nejdůležitější část samotné třídy a to zpracování znakového pole. O tuto činnost se stará funkce `int16ToOpenGL()`, kde se v parametrech předávají zdrojová data, proměnná do které se umísťují zpracované výšky a zeměpisné souřadnice.

Posledním krokem je uložení zpracovaných výšek do souboru .chgt. K tomu je použita třída QDataStream tentokrát s funkcí `writeRawData()` pro zápis binárních dat.

```
QDataStream out(&fDst);
out.writeRawData((char*) &convertedData[0],  // zapisovaná data
                 convertedData.size()*sizeof(float));  // velikost v B
```

Zpracování dat funkcí `int16ToOpenGL()` probíhá následovně. Postupně se prochází načteným polem, krokem je `m_quantity` omezující počet zpracovávaných výšek. Na základě `iX` a `iY` se funkcí `getHeightXY()` se získá výška, poté barva odpovídající dané výšce a také se dopočítají zeměpisné souřadnice.

```
for(int iX = 1; iX <= 1200; iX += m_quantity) {  // řádky
    for(int iY = 1; iY <= 1200; iY += m_quantity) {  // sloupce
        float flX, flY, flZ=0.0f;  // souřadnice bodu
        float flR, flG, flB;       // barva v RGB
        float flHeight;            // výška
        getHeightXY(pbData, iX, iY, flHeight);  // získání výšky
        getFloatRGB(flHeight, flR, flG, flB);   // získání barvy
        flX = iWgsX + iX*1.0/1200.0;    // 1°/1200 výšek
        flY = iWgsY + iY*1.0/1200.0;
        flZ = flHeight;
```

Poté se už jen do proměnné `rfloats` (vektor hodnot float) uloží kombinace bodů, reprezentující danou výšku v podobě dvou trojúhelníků. Tyto body znázorňuje Tabulka 1. Dle umístění bodu se mění první dva řádky následujícího kódu.

```
        rfloats.push_back(flX+m_quantity*1.0/1200.0);
        rfloats.push_back(flY+m_quantity*1.0/1200.0);
        rfloats.push_back(flZ);
        rfloats.push_back(flR);
        rfloats.push_back(flG);
        rfloats.push_back(flB);
```

5.3.3 DrawGLWdg

Tato třída je potomkem třídy QGLWidget, díky tomu je možné .chgt soubory vykreslovat pomocí grafického rozhraní OpenGL. Třída QGLWidget nabízí tři virtuální funkce, které jsou reimplementovány pro potřeby této aplikace.

- Funkce `initializeGL()` – slouží pro nastavení kontextu a připojení ukazatelů na funkce knihovny.
- Funkce `resizeGL()` – nastavuje souřadnicový systém projekce. Volána vždy, když je widget upraven (změna velikosti, souřadnicového systému, ...).
- Nejdůležitější funkce `paintGL()` – se stará o veškeré vykreslování. Použita vždy, když je widget potřeba aktualizovat.

Při prvním vytvoření objektu DrawGLWdg je volána funkce `initializedGL()`, ta se postará o připojení ukazatelů funkcí na dll knihovny. Toto je nutné nastavit pro práci s vertex buffery.

```
glGenBuffers = (PFNGLGENBUFFERSPROC) wglGetProcAddress("glGenBuffers");
glBindBuffer = (PFNGLBINDBUFFERPROC) wglGetProcAddress("glBindBuffer");
glBufferData = (PFNGLBUFFERDATAPROC) wglGetProcAddress("glBufferData");
glDeleteBuffers =
        (PFNGLDELETEBUFFERSPROC) wglGetProcAddress("glDeleteBuffers");
```

Ve funkci `resizeGL()` je důležitý následující řádek, ten nastavuje systém souřadnic pro kreslení. Základní nastavení souřadnic odpovídá hodnotám 11, 20, 48,3 a 51,3 ve stejném pořadí jak ukazuje funkce `glOrhto()`.

```
glOrtho(flLeftGL, flRightGL, flBottomGL, flTopGL, -1.0f, 1.0f);
```

Jak už bylo zmíněno o zobrazování se stará funkce `paintGL()`, než je však možné něco kreslit, je potřeba vytvořit a nahrát data do vertex bufferů. O toto se starají funkce `createHeightsGenBuffer()`, který vytváří buffer z .chgt souborů a funkce `createBorderGenBuffer()`, která vytváří buffer pro hranice ČR.

Prvním krokem je vytvoření identifikátoru bufferu, to se provádí funkcí glGenBuffers(), parametry funkce jsou počet a proměnná do které se uloží identifikátor.

```
glGenBuffers( 1, &uiIdx);
pv->push_back(uiIdx);
```

Poté se buffer aktivuje funkcí `glBindBuffer()` napojením na daný cíl, v tomto případě na pole dat (vrcholy a barvy). Funkcí `glBufferData()` probíhá naplnění bufferu daty, nastavení velikosti a režimu. Režim `GL_STATIC_DRAW` slouží pro čtení a vykreslování prostřednictvím grafické karty.

```
glBindBuffer( GL_ARRAY_BUFFER, uiIdx);
glBufferData( GL_ARRAY_BUFFER, (GLsizeiptr) fFile.size(),
              pflData, GL_STATIC_DRAW );
```

V tomto momentě jsou v grafické kartě nahraná potřebná data pro kreslení, proto lze přejít k samotnému zobrazování funkce paintGL(). Prvním řádkem se vymaže obrazovka a hloubkový buffer. Funkce glEnableClientState() povoluje další funkce pro zobrazování.

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();   // Reset matice
glEnableClientState( GL_VERTEX_ARRAY );
glEnableClientState( GL_COLOR_ARRAY );
```

Poté se bude procházet seznam identifikátorů. Každý identifikátor se zpřístupní funkcí `glBindBuffer()`, poté se funkcemi glVertexPointer() a glColorPointer() specifikuje datový formát a umístění souřadnicových bodů a barev.

```
QVector<GLuint>* pv = &m_vuiHeightsData;
for (int niIdxHgt = 0; niIdxHgt < pv->size(); niIdxHgt++) {
    GLuint iIdxGl = pv->at(niIdxHgt);
    glBindBuffer( GL_ARRAY_BUFFER, iIdxGl);
    glVertexPointer(3, GL_FLOAT, 6*sizeof(GLfloat), 0);
    glColorPointer(3, GL_FLOAT, 6*sizeof(GLfloat),
                   (void*) (3*sizeof(GLfloat)));
```

Následně se již vykreslují tvary specifikované v prvním parametru funkce `glDrawArrays()`. Při kreslení povrchu se kreslí pomocí `GL_TRIANGLES`, pro kreslení hraníc se používá `GL_LINE_STRIP`, který kreslí čáry.

```
    glDrawArrays(GL_TRIANGLES, 0, 1200*1200*6);
}

glBindBuffer( GL_ARRAY_BUFFER, 0 );
glDisableClientState( GL_VERTEX_ARRAY );
glDisableClientState( GL_COLOR_ARRAY );
glPopMatrix();
```

5.3.4 DrawQtWdg

Druhý způsob vykreslování zemského povrchu reprezentuje třída DrawQtWdg, která je potomkem QWidget. Díky tomu je možné využít zděděné funkce `paintEvent()`, v této funkci poté probíhá vykreslování pomocí QPainter.

Součástí třídy jsou dvě hlavní datové položky. První QVector<QPointF> slouží pro uložení souřadnic hranice ČR, druhá QVector<QPair<QPolygonF, QColor> >, představující pole položek typu QPair. QPair slouží pro sloučení položek polygonu a jeho barvy. Do druhé položky je ukládán povrch země.

O naplnění první datové položky se stará funkce createBorderBuffer(), která načte celý hraniční soubor a poté bod po bodu souřadnice ukládá.

```
for (int i = 0; i < sizeFlData; i += 6) {
    m_borderData.push_back(QPointF(*(pflData+i)*xKrat,
                                   *(pflData+i+1)*xKrat));
}
```

Druhou položku plní funkce createHeightsBuffer() v jejímž parametru se předává seznam zdrojových souborů. Postupně prochází tímto způsobem jednotlivé soubory a ukládá je do QVectoru následujícím způsobem, kdy přečte barvu a tři body, které uloží jako polygon.

```
for (int i = 0; i < sizeFlData; i+=18) {
    QColor c = QColor::fromRgbF(*(pflData+i+3), *(pflData+i+4),
                                *(pflData+i+5));
    QPolygonF pf;
    pf << QPointF(*(pflData+i)*xKrat, *(pflData+i+1)*xKrat)
       << QPointF(*(pflData+i+6)*xKrat, *(pflData+i+7)*xKrat)
       << QPointF(*(pflData+i+12)*xKrat, *(pflData+i+13)*xKrat);
    m_heightsData.push_back(qMakePair(pf, c));
}
```

Zde je potřeba zmíní, že načítané souřadnice jsou násobeny číslem 100 (konstanta `xKrat`). Je to z toho důvodu, že funkce `setWindow()` očekává pouze celočíselné parametry, při vkládání reálných čísel docházelo k oříznutí desetinné části a následným deformacím zobrazované scény.

Jak již bylo zmíněno vykreslování probíhá ve funkci `paintEvent()` a je volána vždy, při změně okna aplikace nebo voláním funkce `repaint()`. Pro vykreslování potřebných tvarů reprezentující zemský povrch nebo hranice ČR je použita třída QPainter. Zároveň zde proběhne již zmíněné nastavení souřadnic pro kreslení.

```
QPainter painter(this);
painter.setWindow(flLeftQt, flTopQt,
                  flRightQt-flLeftQt, flBottomQt-flTopQt);
```

Vykreslení povrchu probíhám projitím QVectoru pomocí iterátoru. Nejdříve se nastavuje barva výplně a poté se vykreslují polygony (trojúhelníky). Následně je na povrch zobrazena hranice.

```
QVector<QPair<QPolygonF, QColor> >::ConstIterator it;
for (it = m_heightsData.constBegin();
            it != m_heightsData.constEnd(); it++) {
    painter.setBrush(it->second);
    painter.drawPolygon(it->first);
}
painter.setPen(Qt::black);
painter.drawPolyline(QPolygonF(m_borderData));
```

5.3.5 ImportDlg

Třída ImportDlg představuje dialogové okno sloužící pro nastavení parametrů, jako množství výšek a cílová složka, před zpracováním .hgt souborů. Toto nastavení je předáno parametry funkce `getParams()`.

```
void getParams(QStringList &srcFiles, QString &dstDir, int &q)
{
    srcFiles = m_listNameFiles;
    dstDir = ui->lineEditDirPath->text();
    q = ui->horizontalSliderQuantity->value();
}
```

5.3.6 ProgressDlg

ProgressDlg třída dialogového okna slouží pro zobrazení průběhu některých událostí, sledování času a následné uložení těchto informací do logovacího souboru. Obsahuje tři důležité datové položky.

- První objekt třídy QString uchovávající textový popis probíhající události.
- Druhou položkou je objekt třídy QTimer, který slouží pro emitování signálu v pravidelném intervalu. Je využíván při zobrazení probíhajícího času v dialogovém okně.
- Do třetího objektu třídy QTime se při spuštění záznamu uloží aktuální čas. Z tohoto času se poté dopočítá délka trvání události.

Záznam probíhá volání funkce *start()*, která postupně nastaví `progressBar` a `timeEdit` na počáteční hodnoty a přiřadí aktuální čas objektu `m_time`. Na závěr se spouští `m_timer` aby emitoval signály po jedné vteřině.

```
ui->progressBar->setValue(ui->progressBar->minimum());
ui->timeEdit->setTime(QTime());
m_time = QTime::currentTime();
m_timer.start(1000);
```

Voláním funkce `progressBar(int valp)`, kde se v parametru očekává číslo reprezentující průběh události v procentech. V momentě, kdy obdrží číslo 100 dochází k zastavení timeru a požadavku na uložení informací do logovacího souboru. Zápis je prováděn třídou QTextStream, která běžně slouží pro jednoduché čtení, nebo zápis textu do souboru.

```
QTextStream out(&txt);
QTime tDelka = QTime().addMSecs(m_time.msecsTo(QTime::currentTime()));
out << QDateTime::currentDateTime().toString("dd.MM.yyyy hh:mm:ss")
    << "\t" << this->m_discription
    << "\t" << tDelka.toString("mm:ss.zzz") << endl;
txt.close();
```

Do souboru je zaznamenáván datum a čas zápisu, následovaný popisem události a časem s přesností na tisíciny vteřiny.

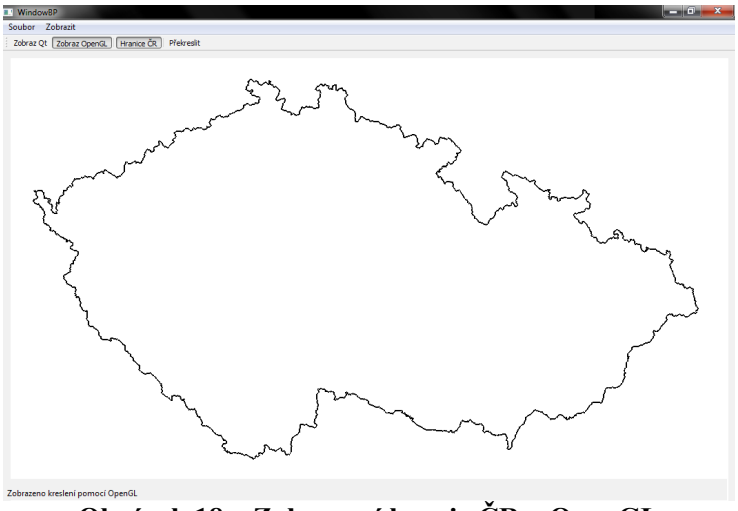
Tato třída je používána pro zobrazení průběhu zpracování zdrojových souborů třídou ConvertHgt a při překreslování panelu tříd DrawGLWdg a DrawQtWdg. Její grafická podoba byla popsána dříve.

6 Měření a analýza získaných výsledků z aplikace

Cílem vytvoření této aplikace bylo zjistit a porovnat rychlost překreslování zobrazených dat pomocí grafického rozhraní OpenGL s využitím vertex bufferu a pomocí třídy QPainter knihovny Qt.

Měření probíhalo formou opakovaného volání žádosti na překreslení aktuálně zobrazeného panelu. Každé měření bylo rozděleno do sad po 10 opakováních, aby se minimalizovali výkyvy při měřeni. Mezi jednotlivými sadami byla aplikace znovu spuštěna, popřípadě byl restartován počítač. Celkem proběhlo 100 opakování pro každou sledovanou oblast. Průběh byl zaznamenáván třídou ProgressDlg, která byla pro tyto účely navržena.

6.1 Zobrazení hranic ČR



Obrázek 18 – Zobrazení hranic ČR v OpenGL

Zdroj: autor

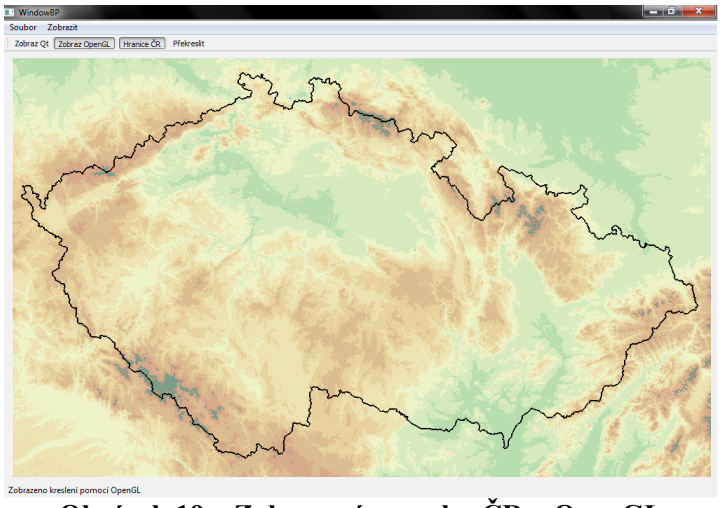
V první testované oblasti byl překreslován pouze soubor s hranicemi České republiky. Tento soubor obsahuje 2835 bodů, jeho velikost je 66,44 KB.

Tabulka 2 – Výsledky měření – hranice ČR

Zobrazení	Průměr (ms)	Min (ms)	Max (ms)
OpenGL vertex buffer	24,51	15	47
Qt QPainter	39,98	31	61

Přestože je třída QPainter o 15,47 ms pomalejší, tak díky tomu, že se časy pohybují v řadu setin vteřiny lze tento rozdíl považovat za minimální.

6.2 Zobrazení povrchu České republiky



Obrázek 19 – Zobrazení povrchu ČR v OpenGL

Zdroj: autor

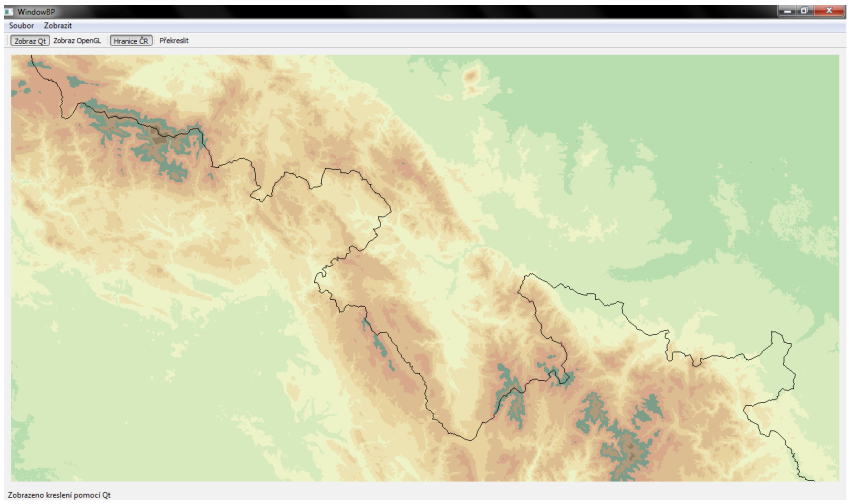
Druhá oblast zahrnuje zobrazení povrchu celé České republiky, spolu s částmi okolních států. Při vytvoření souborů ČR byla použita pouze každá pátá výška. Toto měření obsahuje 28 souborů o celkové velikosti 221,5 MB.

Tabulka 3 – Výsledky měření – Česká republika

Zobrazení	Průměr (ms)	Min (ms)	Max (ms)
OpenGL vertex buffer	56,99	33	78
Qt QPainter	23033,52	22558	23446

Zde už dochází k velmi výraznému zpomalení ze strany QPainter. V porovnání s předchozím měřením je OpenGL 2x pomalejší a QPainter až 576x.

6.3 Regionální zobrazení s přiblížením



Obrázek 20 – Zobrazení regionu v QPainter

Zdroj: autor

V posledním měření byla přiblížena oblast pohoří Krkonoš až sever Jeseníků. Tato oblast zahrnuje tři soubory o celkové velikosti 148,3 MB, ze zdrojových souborů byla použita každá druhá výška.

Tabulka 4 – Výsledky měření – Region

Zobrazení	Průměr (ms)	Min (ms)	Max (ms)
OpenGL vertex buffer	25,73	15	47
Qt QPainter	16296,00	16224	16427

Poslední měření dopadlo podobně jako druhé, přestože bylo překreslovaní ve srovnání s druhým měřením rychlejší, hlavně díky menšímu množství dat. Neustále je QPainter výrazně pomalejší.

7 Závěr

Cílem této bakalářské práce bylo seznámení s projektem SRTM a dalšími výškovými soubory. Jejich praktické využití a způsoby jakými lze mapovat zemský povrch. Dalším cílem bylo vytvoření aplikace, která zpracuje výškové soubory projektu SRTM do podoby vhodné pro následné zobrazení.

Pro vývoj aplikace byl zvolen jazyk C++, Qt framework a vývojovém prostředí Qt Creator pro jeho jednoduchost a přehledně zpracovanou offline nápovědu. Při vývoji aplikace jsem využil získané znalosti z průběhu mého studia předmětů Základy programování, Počítačová grafika, Programovací techniky v jazyce Java, Jazyk C++2 a 3. Díky těmto předmětům jsem mohl tuto aplikaci vytvořit bez větších problémů.

Účelem této aplikace bylo porovnání rychlosti zobrazovaných dat pomocí OpenGL s využitím vetex bufferu a třídy QPainter v Qt C++. Do této doby jsem se prakticky nesetkal s možnostmi využití akcelerované grafiky a proto jsem byl z výsledků měření z počátku překvapen.

Z provedených měření vyplývá, že QPainter je dostačující pro kreslení menšího množství dat (jednotky MB). Se zvyšujícím množstým dat (desitky, stovky MB) se naplno projevují výhody využití akcelerované grafiky, například právě pomocí OpenGL spolu s vertex buffery.

Tato práce prohloubila mé teoretické i praktické znalosti v oblasti Qt Framework, OpenGL a akcelerované grafiky.

Nakonec jsem splnil všechny požadavky zadané v zadání a rád bych se této oblasti věnoval i v budoucnu.

Zdroje

[1] **Informace**. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 26. 5. 2013 [cit. 2013-08-07]. Dostupné z: http://cs.wikipedia.org/wiki/Informace

[2] **ŠMÍD, Vladimír**. Pojem informačního systému. *Fakulta informatiky MU* [online]. [cit. 2013-08-07]. Dostupné z: http://www.fi.muni.cz/~smid/mis-infsys.htm

[3] **Geografie**. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 26. 5. 2013 [cit. 2013-08-07]. Dostupné z: http://cs.wikipedia.org/wiki/Geografie

[4] **Kartografie a Geoinformatika: Multimediální učebnice**. GEOGRAFICKÝ ÚSTAV PŘF MU BRNO. [online]. [cit. 2013-08-07]. Dostupné z: http://oldgeogr.muni.cz/ucebnice/kartografie/obsah.php?show=11&&jazyk=cz

[5] **Vektorové a rastrové modely, geometrie a topologie (4.díl).** *ISVS.cz* [online]. 20.4.207 [cit. 2013-08-07]. Dostupné z: http://www.isvs.cz/vektorove-a-rastrove-modely-geometrie-a-topologie-iv-dil/

[6] **Úvod do GIS.** *Mapový portál města Plzně* [online]. 9.7.2013 [cit. 2013-08-08]. Dostupné z: http://mapy.plzen.eu/gis/o-gis/uvod-do-gis/

[7] **ČADA, Václav**. Souřadnicové systémy. *ZČÚ, Fakulta aplikovaných věd* [online]. [cit. 2013-08-07]. Dostupné z: http://gis.zcu.cz/studium/gen1/html/ch02s03.html

[8] **Universal Transverse Mercator coordinate system.** In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 20. 6. 2013 [cit. 2013-08-07]. Dostupné z: http://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system

[9] **Digital elevation model.** In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 9. 5. 2013 [cit. 2013-08-08]. Dostupné z: https://en.wikipedia.org/wiki/Digital_elevation_model

[10] **Mission Summary.** *Shuttle Radar Topography Mission* [online]. 23.6.2008 [cit. 2013-08-08]. Dostupné z: http://srtm.usgs.gov/mission/missionsummary.php

[11] **SRTM DEM.** *Gisat s.r.o.* [online]. Praha [cit. 2013-08-08]. Dostupné z: http://www.gisat.cz/content/cz/produkty/digitalni-model-terenu/srtm-dem

[12] **SRTM Topography.** *U.S. Geological Survey* [online]. [cit. 2013-08-08]. Dostupné z: http://dds.cr.usgs.gov/srtm/version2_1/Documentation/SRTM_Topo.pdf

[13] **ASTER GDEM.** *Gisat s.r.o.* [online]. Praha [cit. 2013-08-08]. Dostupné z: http://www.gisat.cz/content/cz/produkty/digitalni-model-terenu/aster-gdem

[14] **WorldDEM™.** *Astrium* [online]. 2013 [cit. 2013-08-08]. Dostupné z: http://www.astrium-geo.com/en/168-tandem-x-global-dem

[15] **KOVAŘÍK, P. a M. ŠATÁNEK.** Komerčně dostupné DMT. *ČVUT, Fakulta stavební* [online]. Praha, 2005 [cit. 2013-08-08]. Dostupné z: http://lfgm.fsv.cvut.cz/~hodac/studenti/referaty/sk3_0506.pdf

[16] **PRATA, Stephen.** *Mistrovství v C. 3. aktualiz. vyd.* Překlad Boris Sokol. Brno: Computer Press, 2007, 1119 s. ISBN 978-80-251-1749-1.

[17] **Qt (knihovna).** In: *Wikipedia: the free encyclopedia [online]*. San Francisco (CA): Wikimedia Foundation, 13. 7. 2013 [cit. 2013-08-09]. Dostupné z: http://cs.wikipedia.org/wiki/Qt_(knihovna)

[18] **Grafika a multimédia v Qt.** In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 19. 2. 2013 [cit. 2013-08-09]. Dostupné z: http://cs.wikipedia.org/wiki/Grafika_a_multimédia_v_Qt

[19] **Signals & Slots.** *Qt Project Hosting* [online]. 2013 [cit. 2013-08-10]. Dostupné z: http://qt-project.org/doc/qt-5.0/qtcore/signalsandslots.html

[20] **Qt Creator.** *Qt Project Hosting* [online]. 2013 [cit. 2013-08-15]. Dostupné z: http://qt-project.org/wiki/Category:Tools::QtCreator

[21] **User Interface.** *Qt Project Hosting* [online]. 2013 [cit. 2013-08-15]. Dostupné z: http://qt-project.org/doc/qtcreator-2.8/creator-quick-tour.html

[22] **TIŠNOVSKÝ, Pavel.** Grafická knihovna OpenGL (1). *Root.cz* [online]. 1.7.2003 [cit. 2013-08-15]. Dostupné z: http://www.root.cz/clanky/graficka-knihovna-opengl-1/

Příloha A – Přiložené CD

Přiložené CD obsahuje zdrojové kódy aplikace, zkompilovanou aplikaci v podobě exe souboru spolu s knihovnami pro spuštěni a tento text ve formátu pdf.