VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

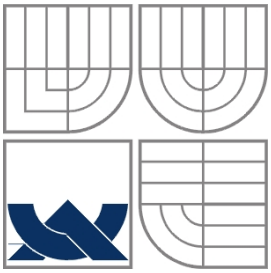
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NAČÍTÁNÍ GIS DAT Z DATABÁZÍ PROJEKTU TERRAGEAR

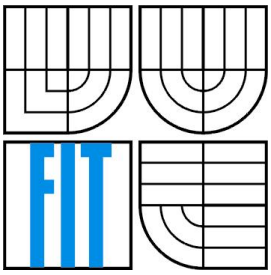
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE MICHEL SAMIA
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NAČÍTÁNÍ GIS DAT Z DATABÁZÍ PROJEKTU TERRAGEAR

OPERATING DATA IN TERRAGEAR GIS PROJECT

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MICHEL SAMIA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Martin Hrubý, Ph.D.

BRNO 2007

Abstrakt

Práce se zabývá návrhem a realizací systému pro poskytování, načítání a zobrazování geografických dat z databází projektu TerraGear s využitím knihovny SimGear. Systém zpřístupňuje digitální model terénu uložený v souborech ve formátu Binary TerraGear skrze jednoduchý aplikační protokol nad protokolem TCP/IP a zobrazuje jej pomocí knihovny FreeGLUT.

Klíčová slova

TerraGear, FlightGear, SimGear, GIS, Geografický informační systém, geoinformatika, WGS 84, UTM, modely terénu, DEM, DTM, výšková mapa, GLUT, TankGame

Abstract

This thesis is concerned with design and implementation of system for providing, loading and viewing geographical data from TerraGear GIS project with taking advance of SimGear library. System accesses digital terrain model and land cover data saved in Binary TerraGear file format via simple application protocol above TCP/IP and views it by FreeGLUT library.

Keywords

TerraGear, FlightGear, SimGear, GIS, Geographical information system, geoinformatics, WGS 84, UTM, terrain models, digital elevation models, digital terrain models, height maps, GLUT, TankGame

Citace

Michel Samia: Načítání GIS dat z databází systému TerraGear, bakalářská práce, Brno, FIT VUT v Brně, 2007

Načítání GIS dat z databází projektu TerraGear

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Martina Hrubého, Ph.D.
Další informace mi poskytli Bc. Tomáš Mrkvička a Ing. Adam Herout, Ph.D.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michel Samia
10.5.2007

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce, panu Ing. Martinu Hrubému, Ph.D., za odbornou pomoc při konzultacích.

Obsah

1 Úvod

1.1 Cíl práce

V současní době existuje celá řada formátů pro uložení geografických dat, neboť každý je vhodný pro určitý typ aplikací. Jedním z těchto formátů je i Binary TerraGear, jenž je určen pro snadné zobrazování krajiny v náročných 3D hrách. Byl vyvinut pro projekt FlightGear – svobodný letecký simulátor.

Cílem mé bakalářské práce je navrhnout a vytvořit software, který umožní přistupovat ke geografickým datům z databází ve zmiňovaném formátu Binary TerraGear, který je výstupním formátem aplikace TerraGear, a zpřístupnit jej pomocí knihovny.

Software se bude skládat ze tří částí: geografického serveru, klienta a samostatné knihovny pro načítání libovolné obdélníkové oblasti nezávisle na rozdělení zeměkoule do „dlaždic“ (anglicky tiles).

Tato knihovna by měla být díky své samostatnosti na projektu do jisté míry nezávislá, a proto ji budou moci použít i jiní programátoři pro vývoj svých vlastních projektů nebo pro začlenění do projektů již existujících (například Tank Game).

Důležité přitom je to, že uživatel této knihovny již nemusí znát detaily týkající se formátu uložení geodat ani protokol pro přenos dat mezi serverem a klientem. Klient pak bude sloužit pouze jako aplikace demonstrující možnosti této knihovny.

1.2 Průvodce kapitolami

V této práci se nejprve seznámíme se systémy souřadnic a s modely terénu (kapitola 2). Tato část nesouvisí přímo s projektem TerraGear ani s mým projektem, ale obsahuje základní pojmy z geografie a geoinformatiky, bez jejichž znalostí bychom se dále neobešli.

Další část (kapitola 3) bude věnována projektům, na které navazuji a ze kterých vycházím. Těmito projekty jsou letecký simulátor FlightGear, aplikace pro zpracování vstupních geodat TerraGear a podpůrná knihovna SimGear, jež obě předchozí aplikace využívají. Zmíníme se také o projektu FlightGear Scenery designer, jde o aplikaci kterou lze použít k přímé editaci

souborů typu Binary TerraGear. Bude zde rovněž popsán formát Binary TerraGear.

Následující část (kapitola 4) bude pojednávat již o mém projektu, o jeho návrhu, dekompozici, způsobu implementace a podobně. Předposlední kapitola (kapitola 5) bude sloužit jako praktický návod na použití knihovny, na instalaci a nastavení serveru a podobně. Poslední kapitola (kapitola 6) celou moji práci shrne, zhodnotí její výsledky a nastíní možnosti dalšího vývoje.

2 Základní geografické pojmy

Nejprve si vysvětlíme několik základních pojmů z geografie, bez kterých bychom se dále neobešli. Seznámíme se s dnes nejužívanějšími systémy souřadnic užívanými v geografii, obzvláště v geografických informačních systémech a s modely terénu.

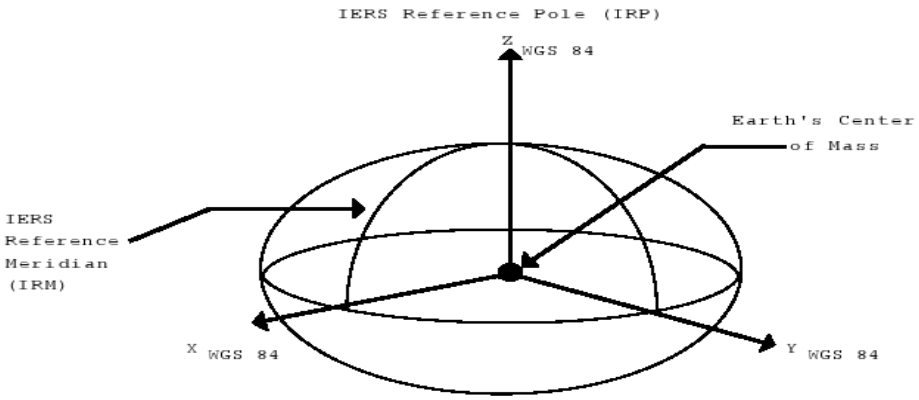
2.1 Systémy souřadnic

Souřadnicových systémů a projekcí v geografii existuje celá řada. Některé jsou určeny pouze pro lokální použití (S42, S-JTSK), jiné pokrývají celý svět. Pro účely této práce si ale vystačíme se systémem WGS-84 a UTM.

2.1.1 WGS 84

World Geodetic System 1984 (zkratka WGS 84), česky *světový geodetický systém 1984*, je světově uznávaný geodetický standard vydaný ministerstvem obrany USA roku 1984, který definuje souřadnicový systém, referenční elipsoid a geoid pro geodézii a navigaci. V roce 1996 byl rozšířen o zpřesněnou definici geoidu Earth Gravity Model 96 (EGM96). Nahrazuje dřívější systémy WGS 60, WGS 66 a WGS 72.

Souřadnicový systém WGS 84 je pravotočivá kartézská soustava souřadnic se středem ve těžišti zeměkoule (včetně moří a atmosféry). Kladná osa *x* směřuje k průsečíku nultého poledníku a rovníku, kladná osa *z* k severnímu pólu a kladná osa *y* je na obě předchozí kolmá ve směru doleva (90° východní délky a 0° šířky). [1] Právě v tomto systému souřadnic jsou uložena geoprostorová data v souborech Binary TerraGear, a tedy i v mém projektu.



Obr. 1: Souřadnice v systému WGS 84 [2]

2.1.2 UTM

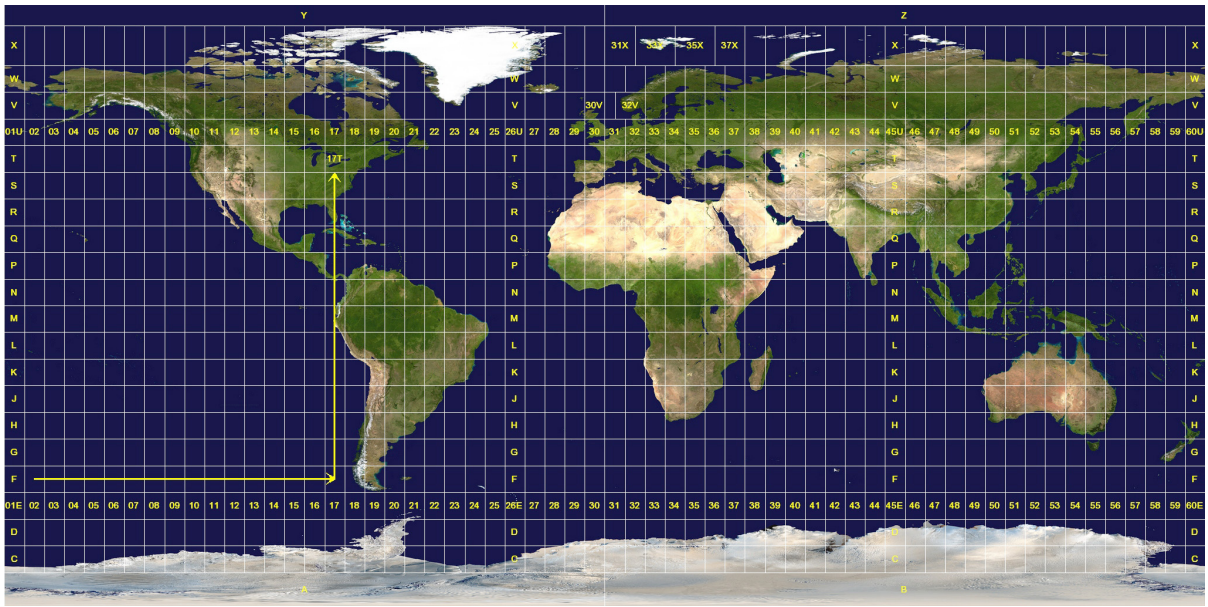
Univerzální Transverzní Mercatorův systém souřadnic (**UTM**) je způsob určování polohy na povrchu Země založený na mřížkách. Od systému šířka-délka se liší v několika zásadních směrech.

Nejedná se o jedno mapové zobrazení ale o síť šedesáti zón zobrazených pomocí transverzního Mercatorova zobrazení. Protože se jedná o zobrazení částí elipsoidu do roviny, lze na mapách v UTM měřit vzdálenost dvou bodů pomocí Pythagorovy věty. Ale pouze v případě, že oba body leží ve stejné zóně.

Poledníkové zóny se v UTM číslují od jedné do šedesáti tak, že zóna č. 1 je tvořena oblastí se zeměpisnou délkou v rozsahu 180 až 174 stupňů západní délky, zóna č. 2 leží mezi 174. a 168. poledníkem západně od Greenwiche a takto se pokračuje dále na východ vždy po šesti stupních.

Střed souřadnic je pro každou zónu jiný a tvoří jej průsečík středového poledníku zóny s rovníkem. Od tohoto středu se měří vzdálenosti v metrech po ose x rostoucí od středového poledníku směrem na východ (tzv. *eastings*) a po ose y rostoucí od rovníku směrem na sever (tzv. *nordings*). [3]

Tento systém je vhodný pro práci s geodaty o malé rozloze, která se vejdou do jedné UTM zóny. Navíc vyměrování v rovině ve 2D kartézském systému je vhodné i pro geografické laiky, neboť je mnohem intuitivnější než systém šířka-délka. Můj projekt bude rovněž umožňovat převod souřadnic systému WGS 84 do systému UTM.



Obr. 2: Rozdělení zeměkoule do UTM zón

2.2 Modely terénu

2.2.1 Digitální výškový model

Digitální výškový model (angl. *digital elevation model*, zkráceně DEM) je reprezentace povrchového reliéfu nebo terénu. Může být reprezentován rastrem (mřížkou čtverců či obdélníků) a nebo nepravidelnou trojúhelníkovou sítí (angl. *triangular irregular network*, zkráceně TIN). Oba tyto formáty jsou mezi sebou navzájem převoditelné.

Digitální výškové modely jsou nejčastěji získávány pomocí dálkového průzkumu země (DPZ), méně často pak přímým průzkumem v terénu. Mocným nástrojem pro získávání DEM je interferometrický radar se syntetickou aperturou, jenž dokáže pořídit pomocí pouhých dvou průletů nad danou oblastí DEM o šířce desítek kilometrů a s rozlišením okolo deseti metrů. [4,5] Používá se především pro modelování krajiny a aplikace určené pro vizualizaci.

2.2.2 Digitální model terénu

Digitální model terénu (angl. *digital terrain model*, zkráceně DTM) je model vzniklý z DEM odstraněním vegetace, budov, mostů a dalších povrchových objektů. Modeluje skutečný terén tvořený kopci, údolími, hřebeny hor, proláklinami a podobně.

Hlavní význam DTM spočívá na rozdíl od DEM v modelování hydrologických jevů (např. povodně), modelování odvodnění, pro geologické výzkumy nebo třeba při hledání nových nalezišť nerostů.

3 Projekty, na které navazuji

Nyní si představíme čtyři multiplatformní open-source projekty, díky kterým mohl můj bakalářský projekt vzniknout.

3.1 FlightGear

Jedná se o velice kvalitní open-source letecký simulátor obsahující nejen věrohodné modely skutečných letadel a jejich chování, ale také digitální model terénu uložený pomocí TIN včetně materiálů (řeky, města, jezera, letiště, pobřeží a řada dalších) pokrývající celou zeměkouli.

Data popisující terén jsou uložena ve formátu Binary TerraGear. V základní distribuci je sice jen malá část USA (oblast kolem letiště v San Franciscu), ale další oblasti lze snadno stáhnout z webové stránky projektu [6]. Data jsou tam uložena v balících o velikosti 10x10 obloukových stupňů systému šířka-délka.

Autorem leteckého simulátoru FlightGear je výzkumný pracovník University Of Minesota, Curtis Olson a oficiální webové stránky projektu naleznete na adrese http://www.flightgear.org.

3.2 TerraGear

TerraGear je sada konzolových nástrojů a renderovacích knihoven určená k přípravě geodat pro letecký simulátor FlightGear. Autorem TerraGearu je taktéž Curtis Olson a oficiální stránky projektu naleznete na webové adrese http://www.terragear.org.

3.2.1 Vstupy a výstupy TerraGearu

Vstupem této aplikace jsou digitální model terénu v podobě rastru a vektorové mapy řek, měst, jezer, letišť, pobřeží a dalších tematických vrstev. Výškové rastry lze stáhnout například ze stránek U. S. Geological Survey [7] a vektorová data pro Evropu pak z FTP archívu National Geospatial-Intelligence Agency (NGA) [8]. Jeho výstupem jsou pak soubory ve formátu Binary TerraGear třízené do pevně dané adresářové struktury, jež si nyní více přiblížíme.

3.2.2 Adresářová struktura souborů TerraGear

Data popisující terén jsou obsažena ve složce *Terrain*, která dále obsahuje podsložky obsahující jednotlivé oblasti zeměkoule o rozloze 10 x 10 stupňů. Tyto podsložky mají opět pevně daný název, který lze charakterizovat regulárním výrazem *{e|w}[0-1][0-9]0{n|s}[0-9]0*. Např. oblast mezi 40. a 50. severní šířkou a mezi 10. a 20. východní délkou se nachází ve složce *Terrain/e010n40/*.

Každá taková složka má další podsložky, které ji rozdělují na 100 oblastí o velikosti 1 x 1 obloukový stupeň. Ty mají opět podobně uspořádané názvy charakterizovatelné regulárním výrazem *{e|w}[0-1][0-9][0-9]{n|s}[0-8][0-9]*. Např. oblast mezi 49. a 50. rovnoběžkou a mezi 16. a 17. poledníkem bude uložena v adresáři *Terrain/e010n40/e016n49/.*

Tyto složky již obsahují soubory ve formátu Binary TerraGear s koncovkou btg.gz a stejnojmenné malé textové soubory s metainformacemi k btg souborům s koncovkou stg.

3.2.3 Pojmenování souborů

I pojmenování jednotlivých souborů s příponami .btg.gz a souborů s metainformacemi s příponami .stg se řídí pevně danými pravidly. Každý Binary TerraGear soubor má i přes pečlivou adresářovou strukturu svoje jedinečné jméno tvořené celým kladným číslem vzniklým následujícím postupem.

K zeměpisné délce oblasti ve stupních přičteme 180 (abychom se dostali do nezáporných hodnot) a posuneme o 14 bitů doleva. K výsledku přičteme zeměpisnou šířku ve stupních zvětšenou o 90 (opět kvůli znaménku) a posunutou o šest bitů doleva. Dále připočteme číslo řádku v dané oblasti (0 až 7) posunuté o tři bity doleva a nakonec číslo sloupce v dané oblasti (0 až 7, v některých oblastech méně – závisí na zeměpisné šířce). Výsledný vztah tedy vypadá takto:

$$index = ((lon + 180) << 14) + ((lat + 90) << 6) + (y << 3) + x \qquad (1)$$

kde *index* je výsledné unikátní číslo btg souboru (tvořící jeho název), *lon* je zeměpisná délka oblasti ve stupních, *lat* její zeměpisná šířka, *y* číslo řádku v dané oblasti a *x* číslo sloupce v dané oblasti. Například soubor obsahující terén v oblasti Brna a jeho blízkého okolí (lon = 16, lat = 49, x = 2, y = 1) je uložen v souboru *Terrain/e010n40/e016n49/ 3220170.btg.gz.*

3.2.4 Formát btg souborů

Vlastní btg soubory jsou (každý zvlášť) komprimovány pomocí knihovny zlib, do formátu gzip (proto tedy přípona btg.gz). Tato knihovna nabízí programátorům velice snadnou práci s komprimovanými soubory naprosto stejným způsobem jako se soubory nekomprimovanými. Jediným rozdílem pro programátora je, že všechny funkce pro práci se souborem (`fopen, fgetc, fread, fclose...`) nahradí funkcemi knihovny zlib, jejichž název se liší pouze v přidání prefixu *gz* a odebráním písmene *f* (například `gzread, gzwrite, gzputc...`).

A nyní již k vlastnímu obsahu samotných (dekomprimovaných) btg souborů. Každý btg soubor začíná takzvaným *magic number* o velikosti čtyř bajtů. První dva bajty udávají číslo verze btg formátu (v současné době je aktuální verze 6) a následující dva bajty jsou „GS“. Je tomu tak proto, protože tato posloupnost vzniká bitovými posuvy znaků SG (zkratka SimGear – viz další podkapitola) a čísla verze ukládaných do datového typu *long*. Ten se pak do souboru uloží jako takzvaný *little endian*, čili nejprve se zapíší bajty s nižším významem. Proto jsou informace v *magic number* pozpátku.

Za touto hlavičkou pak následuje datum a čas uložení souboru uložený bezznaménkově na čtyřech bajtech jako počet sekund uplynulých od začátku tzv. Epochy, tedy půlnoci 1.1.1970 (někdy též označováno jako tzv. unixový nebo POSIXový čas). Čas v tomto formátu lze snadno získat voláním funkce `time(NULL)` ze standardní knihovny jazyka C `time.h`.

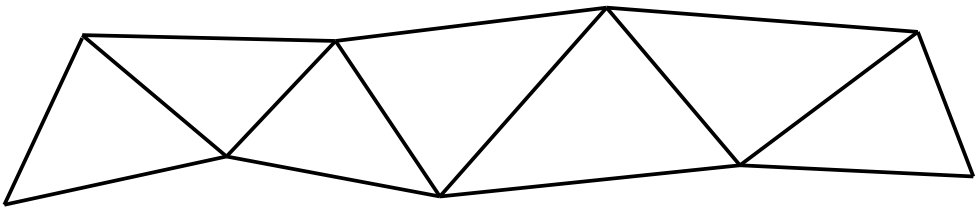
Po časovém razítku pak následuje dvoubajtové číslo (tzv. typ *short*) určující celkový počet takzvaných top level objektů uložených v souboru. Těmito objekty mohou být: SG_BOUNDING_SPHERE, SG_VERTEX_LIST, SG_COLOR_LIST, SG_NORMAL_LIST, SG_TEXCOORD_LIST, SG_POINTS, SG_TRIANGLE_FACES, SG_TRIANGLE_STRIPS, SG_TRIANGLE_STRIPS, nebo SG_TRIANGLE_FANS. Tyto objekty se v jednom souboru mohou teoreticky vyskytovat 0 až n krát. Prakticky se ale každý takovýto objekt vyskytuje v jednom souboru pouze jednou a nebo ani jednou. Význam některých objektů si nyní vysvětlíme.

SG_BOUNDING_SPHERE - popisuje souřadnice středu segmentu (v souřadnicovém systému WGS 84, tedy x, y, z, střed ve středu Zeměkoule... – viz. kapitola 2) a poloměr takzvané *bounding sphere*, tedy nejmenší koule, do které lze celý segment „zabalit“.

SG_VERTEX_LIST – obsahuje souřadnice vertexů (bodů v prostoru), ze kterých se segment skládá. Hodnoty v souboru neudávají přímo souřadnice

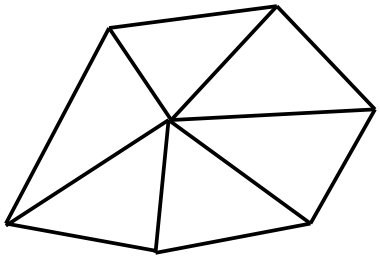
ve WGS 84, ale jsou vztaženy ke středu segmentu uloženém v objektu SG_BOUNDING_SPHERE. Jestliže tedy chceme získat skutečné souřadnice těchto bodů ve WGS 84, musíme k těmto hodnotám připočíst souřadnice středu segmentu. Jedná se vlastně o součet dvou trojrozměrných vektorů.

SG_TRIANGLE_STRIPS – obsahuje informaci o tom, které body v prostoru (vertexy) spolu tvoří takzvané *strips* (pásky, proužky). Více prozradí následující obrázek.



Obr. 3: příklad útvaru zvaného „strip“

SG_TRIANGLE_FANS – obsahuje informaci o tom, které body spolu tvoří takzvané *fans* (k tomuto slovu neexistuje žádný vhodný český protějšek, nejblíže se tomu blíží slova větrníky nebo vějíře). V podstatě se jedná o množiny trojúhelníků se společným vrcholem. Každý „vějíř“ je tvořen posloupností indexů vertexů, přičemž první vertex tvoří střed. Viz obrázek č. 4.



Obr. 4: příklad útvaru zvaného „fan“

Po jejich počtu již následují vlastní top level objekty. Každý začíná nejprve jednobajtovou konstantou udávající, o který z výše uvedených typů se jedná. Poté následují dvě čísla typu short, první se jmenuje *nproperties* udává počet vlastností daného objektu, druhé se jmenuje *nelements* a určuje počet elementů v objektu. Některé objekty nemají vlastnosti žádné a mají jen jeden element (například SG_BOUNDING_SPHERE). Některé, jako například SG_TRIANGLE_FANS, mají elementů a vlastností více.

Zajímavostí je, že při ukládání objektu SG_VERTEX_LIST není použita hodnota *nelements* k uložení počtu vertexů. Na pozici *nelements* se zapíše konstanta „1“ a poté se uloží hodnota *nbytes* vypočtená jako trojnásobek počtu vertexů (ukládají se tři souřadnice) vynásobený velikostí datového typu float.

Při načítání se pak získává počet vertexů opět vydělením počtu uložených bajtů trojnásobkem velikosti typu float. Efektivita tohoto postupu je podobná efektivitě počítání krav na louce tak, že spočteme nohy a dělíme čtyřmi. Bohužel, nikde se mi nepodařilo najít, proč autor naimplementoval ukládání vertexů takto z mého pohledu neefektivně.

3.3 SimGear

Z kapitoly o formátu btg je patrné, že takovýto formát je sice velice důmyslně navržený, ale také, že manipulace s ním bez pomocné knihovny by byla velmi problematická a složitá. A právě toto, stejně jako mnoho dalších problémů, řeší balík knihoven s názvem SimGear. Autorem knihoven SimGear je opět Curtis Olson a oficiální stránky jsou k dispozici na webové adrese http://www.simgear.org/ .

SimGear je balík celkem 24 knihoven (nejen) pro podporu aplikací FlightGear a TerraGear. Tyto knihovny obsahují celou paletu nejrůznějších funkcí: například pro přepočet různých souřadných systémů, pro načítání a ukládání souborů ve formátu TerraGear, pro výpočet poloh nebeských těles na obloze, pro práci s daty ve formátu XML, pro práci se sítí, pro zjišťování počasí parsováním formátu METAR a spoustu dalších užitečností.

3.4 FlightGear Scenery Designer

Jedná se o open-source editor scén (to znamená terénu a povrchových objektů, např. letišť) pro letecký simulátor FlightGear. Pro snadnější manipulaci se soubory ve formátu TerraGear využívá funkce z knihovny SimGear.

Za projektem stojí francouzský vývojář Frederic Bouvier a oficiální stránky naleznete na webové adrese http://fgsd.sourceforge.net/. Právě Scenery Designer mi velice pomohl k pochopení způsobu, jakým se funkce z knihovny SimGear používají.

4 Návrh a implementace

Při návrhu systému jsem vycházel ze současné situace v oblasti hardwaru, tedy z následujících faktů:

- výpočetní výkon klientských PC a serverů se liší relativně málo
- rychlé počítačové sítě jsou cenově dostupné a masově rozšířené (ADSL)

Z toho vyplývají některá moje rozhodnutí ohledně návrhu celého systému. Vzhledem k tomu, že na jeden server se bude moci připojit i několik stovek klientů najednou, musí být co nejvíce výpočetně náročné práce prováděno na straně klienta. Naopak datově náročná část systému, tedy sklad geografických dat, musí být rozhodně na straně serveru. A to hned z několika důvodů:

1. Data mohou být velice obsáhlá a zbytečně by zabírala místo na klientském stroji.
2. Aktuálnost dat – data se mohou časem měnit (zkvalitňovat) a je jednodušší je aktualizovat na jednom serveru, než na stovkách klientů.
3. Dostupnost rychlých sítí – klient si může požadovanou oblast kdykoli získat ze serveru

Takže nyní již víme, že data budou uložena na straně serveru ve formátu Binary TerraGear a že jejich příprava k zobrazení – tedy renderování, bude prováděna na straně klienta. Nyní již zbývá poslední rozhodnutí, a to jak data přenášet.

Možnosti jsou dvě. Buď lze po síti přenášet celé g-zipované btg soubory tak jak jsou, a nebo jen jejich obsah. Ať už ve formě textové a nebo binární. Výhodou prvního způsobu je menší zatížení serveru (nemusí soubor nijak zpracovávat, pouze jej odešle) i sítě (data se přenášejí komprimovaná). Proto jsem se rozhodl právě pro přenos celých btg souborů a jejich další zpracování je již plně v kompetenci klienta.

4.1 Návrh a popis protokolu

Protokol pro přenos souborů od serveru ke klientovi jsem navrhl jako převážně textový aplikační protokol běžící nad transportním a síťovým protokolem TCP/IP.

Skládá se ze tří příkazů: **LIST**, **GET** a **QUIT**. Každý příkaz je ukončen znakem `'\n'`.

4.1.1 Příkaz LIST

Klient nejprve pošle serveru hraniční souřadnice obdélníkové oblasti, kterou chce získat, a server mu na to odpoví seznamem Binary TerraGear souborů, které daná oblast obsahuje. Toto je nutné zejména pro to, že některé soubory nemusí na serveru vůbec být (například soubory, které by obsahovaly pouze moře). K tomu slouží příkaz **LIST**. Syntaxe příkazu **LIST** je následující:

```
LIST left_lon, right_lon, bottom_lat, top_lat
```

kde `left_lon` a `right_lon` jsou celá čísla udávající zeměpisnou délku levého, resp. pravého okraje oblasti a `bottom_lat` a `top_lat` jsou celá čísla udávající zeměpisnou šířku dolního, resp. horního okraje oblasti. Například pro získání seznamu souborů popisujících oblast mezi 15. a 17. stupněm východní délky a 48. a 50. stupněm severní šířky by vypadal zápis příkazu takto:

```
LIST 15, 16, 48, 49
```

Server na tento požadavek odpoví posloupností jmen souborů oddělených znakem `'\n'`. Na konci každé oblasti 1 x 1 stupeň pak vloží ještě řádek s textem `<end of tile>`. Klient si totiž neukládá jen názvy souborů, ale i v jakém segmentu o velikosti 1 x 1 stupeň leží. Tato informace je potřebná při poslání požadavku na stažení souboru ze serveru.

4.1.2 Příkaz GET

Klient se pak pokouší tyto soubory postupně otvírat z jeho vlastní lokální cache. Jestliže se mu nějaký soubor nepodaří z cache otevřít (funkce `fopen()` vrátí `NULL`), znamená to, že soubor ještě do té doby nebyl potřeba a nebo byla cache smazána. Je tedy potřeba soubor stáhnout ze serveru příkazem **GET**. Jeho syntaxe je následující:

```
GET longitude, latitude, filename
```

kde `longitude` je celé číslo udávající zeměpisnou délku oblasti s daným souborem, `latitude` celé číslo udávající zeměpisnou šířku oblasti s daným souborem a `filename` je jméno požadovaného souboru (včetně přípony `.btg.gz`). Takže například příkaz pro stažení souboru popisujícího terén v okolí Brna by vypadal následovně:

```
GET 16, 49, 3220170.btg.gz
```

Na tento dotaz odpoví server zasláním retězce obsahujícího délku požadovaného souboru v bajtech v dekadické textové podobě a ukončeného opět znakem `'\n'`. Po něm již následuje právě tolik bajtů, kolik daný soubor obsahuje, tedy vlastní obsah binárního souboru. Například odpověď na výše uvedený dotaz by klient dostal následující odpověď (uvádím zde pochopitelně jen prvních několik málo bajtů):

```
61953
*-J!-IĄ!őzëE({e$´KR˝Šďçwţ8ˇëyť]çvçšn×u÷??×şÍŐ,1ŽÁÍ5fv~0ˇÉ|
NÚˇď?óqĎM[ˇ_Ö_eÉýNu6\\s&ow?Ţ...
```

4.1.3 Příkaz QUIT

Jestliže již klient nechce dále využívat služeb serveru (například při ukončení), zašle mu příkaz **`QUIT`**. Tento příkaz je bez parametrů. Server na něj zareaguje uzavřením socketu, tedy i TCP/IP spojení.

4.2 Návrh a implementace serveru

Prvním mým požadavkem na serverovou aplikaci byla rychlost. S tím souvisí i řada rozhodnutí ohledně návrhu komunikačního protokolu (viz výše). Proto server odesílá pouze komprimované soubory tak jak jsou a neprovádí s nimi žádné další složité operace (filtrování, spojování..).

Dalším mým požadavkem bylo, aby služba vytěžovala počítač pouze ve chvíli, kdy opravdu zpracovává požadavek nějakého klienta. Toho je docíleno tím, že server využívá takzvaných blokujících schránek. Tedy, že ve chvíli, kdy čeká na připojení uživatele nebo na příkaz (dotaz) již připojeného uživatele, je jeho podproces přenesen na pozadí, uspán.

Dalším požadavkem bylo, aby se na server mohl připojit neomezený počet klientů, a to i ve stejnou chvíli. To znamená, aby byl server takzvaně *konkurentní*. To je vyřešeno tak, že ve chvíli, kdy se připojí uživatel, vytvoří hlavní proces podproces, který se již věnuje pouze danému uživateli. Čili pro každé nově příchozí spojení vznikne nový podproces. Ve chvíli, kdy klient dokončí komunikaci se serverem příkazem **`QUIT`** a spojení mezi klientem a serverem se uzavře, ukončí se i daný podproces. K vytvoření nového podprocesu při připojování nového klienta je použita standardní unixová funkce `fork()`.

Server je implementován v programovacím jazyce C++, neboť se jedná o standardizovaný jazyk dostupný prakticky na všech platformách. Při vývoji nebylo použito žádných nestandardních rozšíření jazyka ani žádných nepřenositelných knihoven. Tím by měla být zaručena jeho přenositelnost. Vyvíjen byl na platformě GNU/Linux (distribuce Debian 4.0 Etch) a jako překladač byl použit g++ ve verzi 4.1.2. Kromě toho byl server úspěšně testován také na školním studentském linuxovém serveru Merlin (distribuce CentOS).

4.3 Návrh a implementace klienta

Nejdůležitějšími požadavky na klientskou aplikaci (dále jen klient) byla přenositelnost a 3D grafický výstup. Jako řešení obou požadavků jsem zvolil použití multiplatformní knihovny GLUT (zkratka GL Utility Toolkit), konkrétně pak její modernější varianty FreeGLUT, verze 2.4.0-5.1. Jedná se o nadstavbu nad grafickou knihovnou OpenGL, která abstrahuje rozdíly mezi různými operačními systémy a usnadňuje tak programátorovi práci při psaní aplikací s 3D výstupem. Odstiňuje například rozdíly mezi Unixovým X Window system a GDI v Microsoft Windows.

Dalším mým požadavkem bylo, aby proces klienta nevytěžoval procesor neustále, ale pouze v době, kdy je potřeba. Toho jsem docílil tak, že okno se překresluje jen při nezbytných událostech (posunutí scény, přiblížení, oddálení...). Pro vykreslování není, na rozdíl od většiny 3D her a aplikací, použita vykreslovací smyčka ani časovač (timer). Proto nelze ani měřit počet překreslení za sekundu.

Pro načítání dat ve formátu Binary Terragear z lokální cache používám knihovny SimGearu, konkrétně členskou funkci `read_bin` třídy `SGBinObject` z knihovny `sgio` (hlavičkový soubor `<simgear/io/sg_binobj.hxx>`).

Všechny textové výstupy aplikace (aktuální zeměpisná šířka, zeměpisná délka a výška nad povrchem, počet vykreslených polygonů, velikost načtené oblasti, probíhající stahování TerraGear souborů apod.) se vypisují do konzole. Proto je vhodné tuto aplikaci, ačkoli je okenní, spouštět z konzole. Údaje o poloze jsou navíc při pohybu po scéně vypisovány vždy na stejný řádek konzole, takže nedochází k rušivému zaplňování stále se opakujícími řádky.

Klient byl stejně jako server napsán v jazyce C++ na platformě GNU/Linux (distribuce Debian 4.0 Etch) a kompilován překladačem g++ ve verzi 4.1.2. I klient byl úspěšně otestován na školním serveru Merlin.

Klientskou aplikaci jsem pojmenoval Scenery View, neboť to vystihuje její použití. Klient se ovládá výhradně pomocí klávesnice. Více o ovládání klienta najdeme v kapitole č. 5.

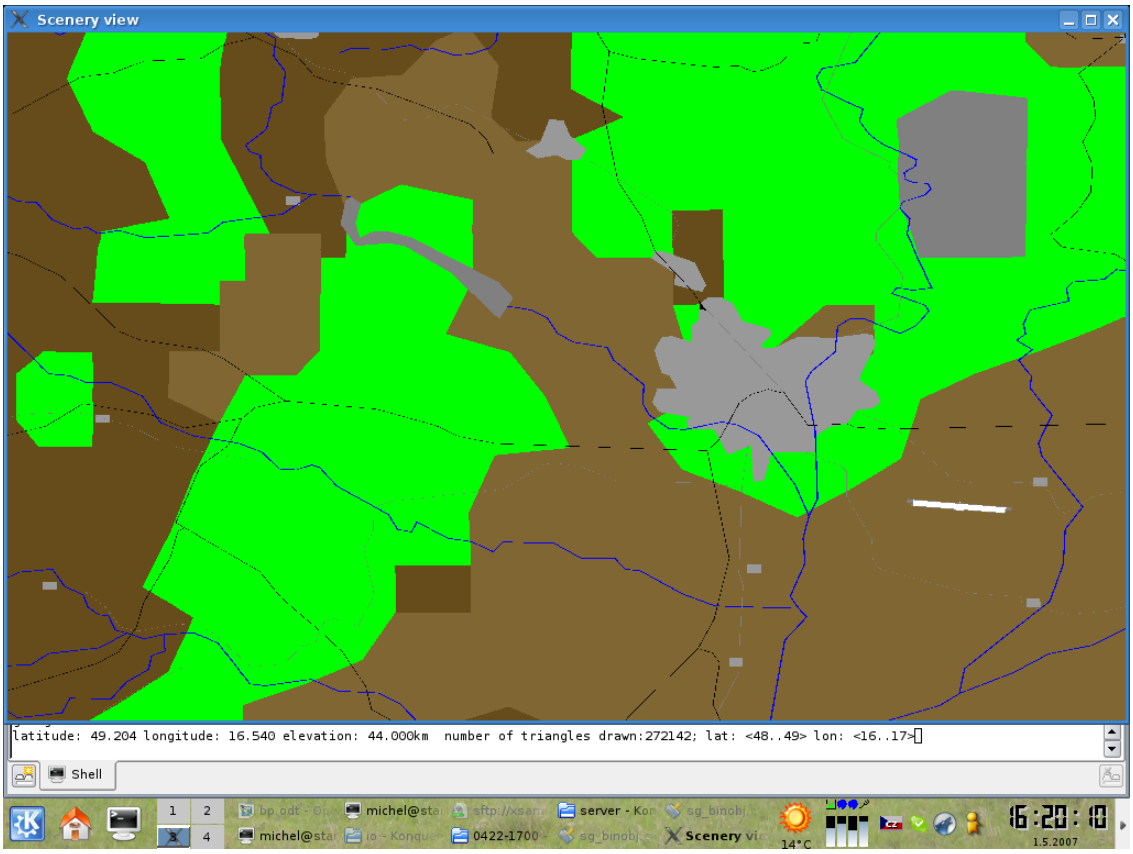
Při pohybu po scéně bylo potřeba vyřešit problém, o velkou vzdálenost se posunout při jednom stisku klávesy. Pohyb nesmí být příliš malý, neboť by pak přesuny trvaly příliš dlouho. Nesmí být ale ani moc velký, neboť by uživatel ztrácel pojem o tom, kde se vlastně momentálně nachází. To vše je navíc komplikováno možností měnit svoji vzdálenost od povrchu pomocí zoomování. Problém jsem vyřešil následovně. Nejprve jsem si empiricky zjistil, že vhodná velikost posunu na jedno stisknutí tlačítka je přibližně jedna pětina zorného pole. Potom již zbývalo jen odvodit vztah pro výpočet velikosti skoku (ve stupních) v závislosti na vzdálenosti od kamery. Ten je popsán vzorcem (2):

$$step\_degrees = elevation * velocity / ONE\_DEGREE\_IN\_METERS \qquad (2)$$

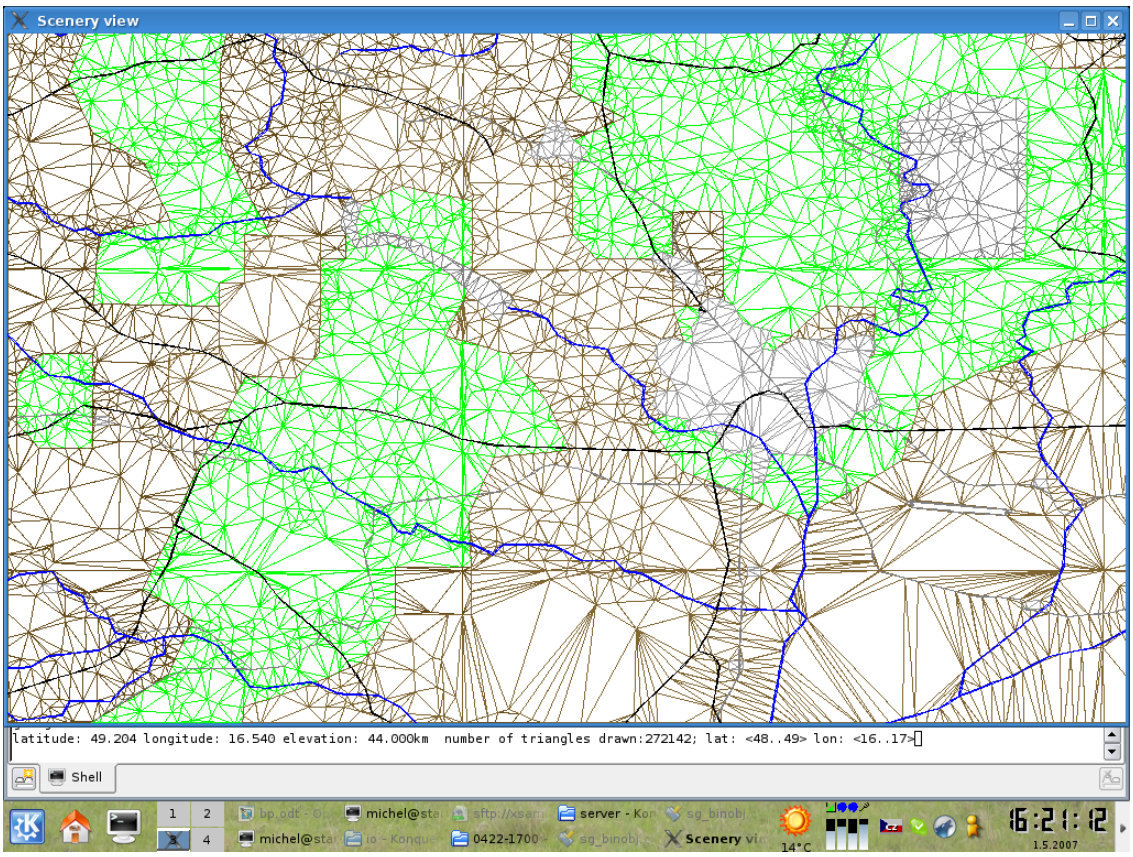
kde *step_degrees* je skok ve stupních (ať už po rovnoběžce ve směru východ-západ, nebo po poledníku ve směru sever-jih), *elevation* je výška nad povrchem v metrech, *velocity* je konstanta udávající o jak velkou část obrazovky chceme poskočit (například pro zmiňovanou jednu pětinu obrazovky by měla hodnotu 0.2) a *ONE_DEGREE_IN_METERS* je konstanta udávající délku jednoho stupně poledníku v kilometrech. Tento vztah funguje dobře pro směr sever-jih, neboť poledníky jsou všechny všude stejně dlouhé. Pro správný chod i ve směru výhod-západ v oblastech blízko pólů by bylo vhodné brát v úvahu i zkracování rovnoběžek.

Dalším problémem, který bylo nutné řešit, byla správa paměti. Každý segment zabírá v paměti několik megabajtů, a proto není vhodné mít v jedné chvíli načtené velké množství segmentů. Tento problém jsem vyřešil tak, že načítám vždy jen čtyři segmenty, které jsou nejblíže ke kameře.

Klient umožňuje kromě pohybu po scéně (na sever, na jih, na východ, na západ) a přibližování a oddalování scény také přepínání mezi dvěma režimy zobrazení. V prvním režimu (výchozí po spuštění) vidíme polygony vyplněné (v angličtině se někdy označuje jako *filled mode*), zatímco ve druhém režimu vidíme pouze spojnice mezi vertexy, takzvaný drátěný model (anglicky *wireframe mode*). Nyní následuje ukázka obou režimů zobrazení. Šedý útvar v pravé části obrázku je Brno. Na obrázcích lze dále najít řeky Svratku a Svitavu, Brněnskou přehradu v Bystrci, dálnici D1 a také letiště v Tuřanech.



Obr. 5: Klientská aplikace v režimu vyplněného vykreslování



Obr. 6: Klientská aplikace v režimu drátěného modelu

5 Návod na použití

Zde si vysvětlíme, jak celý systém používat a jak psát aplikace používající moji knihovnu pro načítání.

5.1 Volby při spuštění serveru

Server se spouští ve tvaru

```
./server [-p port_number] [-d data_directory]
```

Argument `-p` s následně uvedeným číslem portu je nepovinný a udává, na kterém portu bude server očekávat příchozí spojení (naslouchat). Jestliže nejsme systémový administrátor (root), neměli bychom volit číslo portu nižší než 1024. Jestliže číslo portu nezadáme, bude použito výchozí číslo portu 1027 (jedná se o první neregistrované číslo portu dle seznamu organizace IANA [9]).

Argument -d s následně uvedeným adresářem udává, kde jsou uložena data, která bude server poskytovat. Jestliže tento parametr nezadáme, použije se výchozí složka `./data`. Jiným postupem, jak použít jinou než výchozí cestu k datům, je vytvořit v pracovním adresáři serveru symbolický odkaz na složku s daty pojmenovaný také `./data` (nelze použít na souborových systémech nepodporujících symbolické odkazy, například FAT32 nebo NTFS).

5.2 Volby při spuštění klienta

Klient se spouští ve tvaru

```
./sv [-h hostname] [-p port_number]
```

Přepínač `-h` s následně udaným jménem serveru nebo IP adresou serveru určuje, na který server se klient připojí, z kterého bude stahovat potřebná data. Jestliže přepínač `-h` neuvedeme, použije se výchozí nastavení, kterým je IP adresa 127.0.0.1 odkazující na počítač, na kterém právě pracujeme. Jestliže tedy spouštíme server i klient na stejném stroji, není potřeba tento přepínač uvádět.

Přepínač `-p` s následně uvedeným číslem portu určuje, na kterém portu bude klient server kontaktovat. Výchozí hodnota je 1027. Jestliže jsme neměnili číslo portu na straně serveru, není potřeba ho tedy měnit ani na straně klienta a přepínač můžeme vynechat.

5.3 Ovládání klienta

Jak již bylo řečeno v kapitole o návrhu klienta, veškeré ovládání se provádí pomocí klávesnice. K pohybu po zeměkouli slouží klávesy „A“ (na západ), „D“ (na východ), „W“ (na sever) a „S“ (na jih). Rozložení těchto kláves odpovídá rozložení kurzorových kláves (šipek) a je používáno i v mnoha dalších aplikacích, především ve hrách.

K zoomování skouží klávesy „K“ (oddálení od povrchu) a „L“ (přiblížení k povrchu). Jedná se o velice ergonomické rozložení kláves, kdy uživatel levou rukou pohybuje scénou a pravou rukou zoomuje. Přepínání mezi drátěným modelem a vyplněným vykreslováním se provádí klávesou „P“ (od slova polygon mode).

5.4 Použití knihovny pro přístup ke GIS datům z TerraGear

Prvním krokem při použití knihovny tgload je pochopitelně dovoz jejího rozhraní direktivou `#include “tgload.h“` a její začlenění do projektu vhodnou úpravou Makefile.

Dalším krokem je pak volání `funkce tginit()`, která vytvoří složku `cache` v pracovním adresáři a také zajistí automatické odhlášení od serveru v případě, že dojde k ukončení vaší aplikace (registrací funkce `postQuitMessage` pomocí funkce `atexit`).

Dále je potřeba se připojit na server pomocí následující funkce: `bool server_connect(char * host, int port)`, kde `host` je adresa serveru a `port` je číslo portu, na kterém server naslouchá. Jestliže se připojení nezdaří, vypíše se důvod na standardní chybový výstup a funkce vrátí hodnotu 0.

Nyní, když jsme již připojeni, může začít vlastní práce s daty. Ta spočívá ve dvou krocích. Prvním z nich je výběr oblasti, kterou chceme dále používat. K tomu slouží funkce `void load_data(int lon1,int lon2,int lat1,int lat2)`, kde `lon1` a `lon2` jsou celá čísla udávající zeměpisnou délku západní, respektive východní hranice oblasti a `lat1` a `lat2` jsou celá čísla udávající zeměpisnou šířku jižní, respektive severní hranice oblasti. Tato oblast pak bude načtena do paměti RAM takovým způsobem, že segmenty (reprezentované jednotlivými btg soubory), které již v paměti jsou, se znovu nenačítají, ty, které

nejsou v RAM, ale jsou na disku ve složce `./cache`, se načtou z ní a ty, které nejsou ani tam, se stáhnou ze serveru do složky `./cache` a následně se z ní načtou do RAM. Zavoláme-li funkce vícekrát, bude předchozí oblast odalokována.

Vlastní přístup ke geografickým datům je pak zprostředkován funkcí `Tbucket* getBucket(int lat, int lon)`, kde `lat` a `lon` jsou zeměpisná šířka a délka oblasti o rozměru 1x1 obloukový stupeň, ke které chceme přistoupit. Funkce vrací seznam (přesněji řešeno vektor) již načtených objektů typu SGBinObject, které daná oblast obsahuje. S nimi již můžeme pracovat pomocí jejich metod definovaných v `<simgear/io/sg_binobj.hxx>`. Důležité přitom je to, aby byla daná oblast předtím načtena do paměti výše uvedenou funkcí `load_data`. Jinak funkce `getBucket` vrátí prázdný seznam.

6 Závěr

V projektu jsem využil znalosti hned z několika předmětů: díky znalostem a zkušenostem z předmětu Základy počítačové grafiky jsem mohl jednoduše vykreslovat složité 3D objekty pomocí knihovny FreeGLUT, díky předmětům zaměřeným na sítě jsem mohl snadno implementovat vlastní protokol pro přenos souborů, z předmětu Geografické informační systémy jsem zde využil znalosti o systémech souřadnic a modelech terénu a díky předmětům zaměřeným na jazyk C a C++ jsem mohl vše spojit do jednoho celku. V případě síťové části aplikace došlo i k znovuvyužití několika mých funkcí z projektu do předmětu Síťové aplikace a správa sítí.

Největším problémem při implementaci bylo porozumění knihovnám SimGear a formátu Binary TerraGear, neboť k nim existuje jen velice málo dokumentace. Jediným materiálem o SimGear je její programová dokumentace vygenerovaná z jejích zdrojových textů pomocí programu Doxygen. Proto pro mne byly velice užitečné zdrojové texty projektu FlightGear Scenery Designer. Jejich důkladnou analýzou jsem pochopil použití funkcí SimGear pro načítání geodat. Tato část projektu byla časově velice náročná, a proto mi již nezbyl dostatek času na některé další funkce, které by více rozšířily možnosti použití klientské aplikace a knihovny pro načítání dat po síti. Na projektu však hodlám dále pokračovat a tyto funkce dotvořit.

Nejbližšími plánovanými rozšířeními budou přidání podpory souřadného systému UTM do knihovny pro načítání a zjemnění dílků, po kterých lze data z mé knihovny načítat. V současné době je nejmenším dílkem 1 x 1 obloukový stupeň, kdežto jednotlivé btg soubory pokrývají oblasti o hraně jedné osminy stupně. V další fázi pak dojde k úplné abstrakci systému dlaždic a uživatel bude moci zadat hranice oblasti jako reálná (tedy desetinná) čísla. Knihovna tak bude lépe připravena pro začlenění do projektu Tank Game.

Podstatným vylepšením by bylo rovněž začlenění veřejně dostupných ortofomap do projektu, neboť formát Binary TerraGear obsahuje i možnost uložení souřadnic textur. To by si již ale vyžádalo zásadnější změny v kódu celé aplikace.

Jako velice perspektivní pro další vývoj se jeví využití standardních otevřených geo-informatických protokolů, jež by významně rozšířilo interoperabilitu celého systému.

Dalším praktickým rozšířením systému by byla implementace určitého způsobu zjednodušování modelu terénu v závislosti na vzdálenosti od kamery. Tato změna úrovně detailů (anglicky *level of detail*, zkratka LOD), by mohla být zajištěna například pomocí algoritmu *progressive meshes*, který v rámci bakalářské práce vyvíjí spolužák Jan Zelený. Začlenění jeho algoritmu formou vhodné knihovny by poskytlo mé aplikaci vyšší výkon a možnost zobrazovat větší území.

Největším mým přínosem na projektu je přiblížení problematiky práce databázemi TerraGear, s formátem souboru Binary TerraGear a knihovnou SimGear určenou pro manipulaci s ním. Jedná se zřejmě o první dokument na toto téma psaný v češtině.

Literatura

[1] *Wikipedie: Otevřená encyklopedie: WGS 84* [online]. c2007 [citováno 18. 04. 2007]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=WGS_84&oldid=1366189>

[2] *National Imagery and Mapping Agency: World Geographic System 1984 Technical Report* [online]. c2000 [citováno 8. 5. 2007]. Dostupný z WWW: <http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf>, strana 24

[3] *Wikipedie: Otevřená encyklopedie: UTM* [online]. c2007 [citováno 18. 04. 2007]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=UTM&oldid=1409197>

[4] Hrubý, M.: *Digitální modely terénu* (3. přednáška předmětu GIS), 2007, Dostupný z WWW: <http://perchta.fit.vutbr.cz:8000/vyuka-gis/uploads/1/gis3rastr-rok07.pdf>

[5] Přispěvatelé Wikipedie: *Digital elevation model* [online]. Wikipedia, The Free Encyclopedia [citováno 21. 04. 2007]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Digital_elevation_model&oldid=124099946>

[6] OLSON, Curtis. *FlightGear scenery download* [online]. 2006 [cit. 2007-05-04]. Dostupný z WWW: <http://www.flightgear.org/Downloads/scenery.html>.

[7] U. S. Geological Survey*, Center For Earth Resources Observation and Science (EROS)*. *Elevation* [online]. 2006 , Page Last Modified: August 22, 2006 [cit. 2007-05-04]. Dostupný z WWW: <http://edc.usgs.gov/products/elevation/gtopo30/gtopo30.html>.

[8] *National Geospatial-Intelligence Agency. GeoEngine : Europe* [online]. 2006 [cit. 2007-05-04]. Dostupný z WWW: <http://geoengine.nga.mil/ftpdir/archive/vpf_data/v0eur.tar.gz>.

[9] *Internet Assigned Numbers Authority (IANA): Port Numbers*, citováno 02. 05. 2007, platné k 01. 05. 2007. Dostupný z WWW: <http://www.iana.org/assignments/port-numbers>