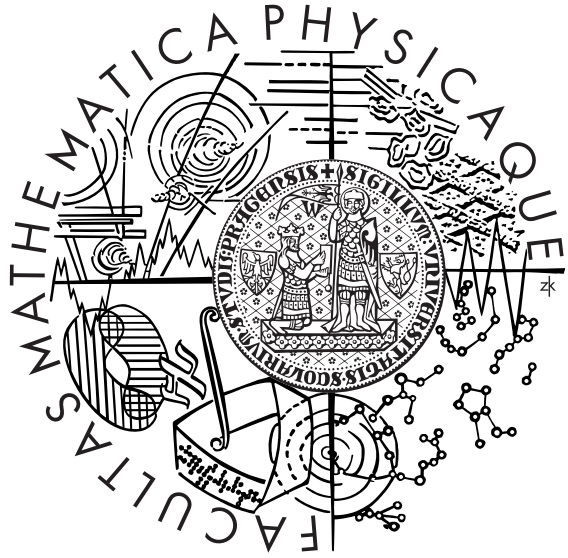
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



David Stach

Prohlížeč vektorové mapy České republiky pro mobilní telefony s podporou Javy

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Pavel Machek

Studijní program: Informatika, Programování

2007

Děkuji především Petru Bitnarovi, který na internetové stránky www.gps-maps.info umístil vektorovou mapu České republiky, kterou za pomoci kolektivu svých přátel vytvořil.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 29. května 2007 David Stach

Obsah:

1 ÚVOD 6

2 VEKTOROVÁ DATA 8

2.1. Výběr vektorových dat 8

2.1. Formát vektorových dat 9

3 ZPRACOVÁNÍ DAT 11

3.1. Důvody konverze a úpravy 11

3.2. Příprava dat ke zpracování 11

3.3. Zpracování jednotlivých vrstev mapy 12

4 GRAFICKÁ REPREZENTACE 14

4.1. Hlavní nedostatky grafické knihovny a jejich řešení 14

4.2. Popis objektů mapy 17

4.3. Souhrn použitých objektů 18

5 PROGRAMOVÁ ČÁST APLIKACE 23

5.1. Základní popis 23

5.2. Běh aplikace 23

5.3. Třída poi 25

5.4. Třída po 25

5.5. Třída Vlakno 26

5.6. Zpracování jednotlivých řádků třídou Vlakno 28

5.7. Třída Mapa 31

6 OVLÁDÁNÍ 40

6.1. Základní ovládací prvky 40

6.2. Urychlení obsluhy aplikace 41

7 POROVNÁNÍ METOD 42

7.1. Mapa jako obrázek 42
7.2. Mapa ve webovém prohlížeči 42
7.3. J2ME prohlížeč rastrových map 43
7.3. Online mapy 44
7.4. Vektorové mapy 44
7.5. GPS navigace 45

8 MOŽNÁ VYLEPŠENÍ 46

8.1. Binární data 46
8.2. Vyhledávání objektů 46
8.3. Vyhledávání pomocí pozice 47
8.4. Vkládání bodů zájmu, ukládání tras 47
8.5. Filtrace dat 47
8.6. Uložení poslední pozice 47
8.7. Nápověda 48
8.8. Inverzní mapa 48
8.9. Načítání libovolných map 48
8.10. GPS navigace 48

9 ZÁVĚR 49

LITERATURA 50

PŘÍLOHY 51

Obsah přiloženého CD 51

Název práce: *Prohlížeč vektorové mapy České republiky pro mobilní telefony s podporou Javy*

Autor: *David Stach*

Katedra: *Ústav formální a aplikované lingvistiky*

Vedoucí bakalářské práce: *Mgr. Pavel Machek*

email vedoucího: *pavel@ufal.mff.cuni.cz*

Abstrakt: *Bakalářská práce je zaměřena na vytvoření aplikace do mobilního telefonu, která umožní zobrazovat mapu České republiky reprezentovanou vektorovými daty. Vytvoření vektorových dat není součástí této práce, aplikace přebírá již hotovou vektorovou mapu. Úkolem práce je především zpracovat vektorová data a umožnit jejich zobrazení na display mobilního telefonu. Aplikace musí používat takové algoritmy, které umožní rychlou práci s mapou i přes omezené zdroje zařízení.*

Klíčová slova: *mapa, vektorová data, Česká republika, J2ME, mobilní telefon*

Title: Viewer of a vector map of the Czech Republic for mobile phones supporting Java

Author: *David Stach*

Department: *Institute of Formal and Applied Linguistics*

Supervisor: *Mgr. Pavel Machek*

Supervisor's e-mail address: *pavel@ufal.mff.cuni.cz*

Abstract: *The bachelor thesis is focused on creating of aplication for mobile phones, which provides to view map of the Czech Republic represented by vector data. Creation of vector data is not a part of this thesis, aplication uses already finished map. Processing of vector data and displaying of them is main point of thesis. Aplication has to use algorithms, which will enable fast work with map despite of limited sources of unit.*

Key words: *map, vectorical data, Czech Republic, J2ME, mobile phone*

Kapitola 1

Úvod

V době relativně výkonných mobilních telefonů vznikla myšlenka vytvořit mapu tak, aby ji uživatel měl vždy po ruce – stejně tak jako má vždy po ruce mobilní telefon – tedy umístit mapu přímo do zařízení.

První a nejjednodušší nápad byl vzít rastrovou mapu a vložit ji do telefonu. Ta pak šla prohlížet jako normální obrázek. Vylepšením bylo vytvoření aplikace, která je schopna stránkovat rozsáhlejší mapy, popřípadě indexovat některá místa. Bohužel takovéto mapy byly a jsou příliš objemné a přitom ne vždy jsou dostatečně podrobné.

Druhým krokem byl vznik online map. Uživatel zadá místo, které chce zobrazit a odešle na server požadavek na potřebnou mapu. Proces se s různými požadavky uživatele následně opakuje. Tato metoda podává kvalitní výsledky, neboť jsou požadavky vyhodnocovány serverem, který může jednak pracovat mnohokrát rychleji, ale také s daleko větším objemem dat. Velkou nevýhodou této metody je však velký datový přenos, který je mobilními operátory zpoplatněn nemalými částkami.

Poslední variantou je vektorová mapa podobná té, která se používá v GPS přijímačích. Výhody vektorových map jsou zřejmé – jednoduše lze měnit měřítko mapy a tím i zobrazené detaily na mapě. Při kvalitě srovnatelné s rastrovou mapou, mají daleko menší objem dat. Avšak stejně jako předchozí metody má i vektorová mapa nevýhody - největší je dostupnost dat, která jsou obvykle buď velmi drahá, nebo nedosahují potřebných kvalit (zejména co se týče obsahu). Především z tohoto důvodu ještě žádná vektorová mapa České republiky do mobilního telefonu v jazyce Java nevznikla.

Ve své bakalářské práci jsem se proto rozhodl vytvořit aplikaci reprezentující vektorovou mapu ČR do mobilního telefonu. Cílem bakalářské práce však není vytvářet vektorovou mapu, pouze vytvořit její reprezentaci v jazyce Java 2 Micro Edition.

Mým vlastním požadavkem na aplikaci bylo, aby ji mohl uživatel volně používat - tedy především sehnat volně dostupná a šířitelná mapová data. Dále by aplikace neměla využívat API jednotlivých výrobců tak, aby byla přenosná na co nejvíce zařízení podporujících MIDP 2.0 a CLDC 1.1. [3]

Kapitola 2

Vektorová data

2.1. Výběr vektorových dat

V první fázi - ve fázi přípravy bylo potřeba najít vhodná vektorová data, která bude program zpracovávat. Existuje několik kvalitních vektorových map ČR. První takovou je ZABAGED (Základní báze geografických dat) od ČÚZK (Český úřad zeměměřický a katastrální) druhou, IZGARD (Digitální atlas ČR) vlastní Vojenský geografický a hydrometeorologický úřad. Další vektorové mapy ČR jsou vlastněny soukromým sektorem. Ani v jednom případě však vlastník nedává data volně k dispozici.

Částečnou výjimku tvoří ZABAGED, kde vlastník dovoluje použít zdarma vektorová data, ale pouze pro účely bakalářské či diplomové práce. Navíc úřad poskytne jen malou část celé mapy.

Ucelenou mapu České republiky, která splňovala kriteria (tedy volně šířitelná a kvalitně zpracovaná alespoň k soukromému využití) lze stáhnout na internetových stránkách www.gps-maps.info. Je to mapa vytvořená sloučením několika různých zdrojů mapových dat a dále doplněna vlastními daty (naměřenými nebo transformovanými z rastrových map). Autor webového portálu uvádí, že jsou všechna data použitá k vytvoření této mapy volně šířitelná a proto je možné i volné šíření této mapy.

Mapa byla vytvořena pro potřeby GPS přijímačů firmy Garmin nadšenci, kteří obětovaly mnoho času k vybudování celého díla. Proto je mapa také dostupná ve formátu pro tyto přístroje. Mapu lze však konvertovat do tzv. polish formátu, což je

textový a velmi přehledný formát. V této podobě lze dále mapu dle libosti modifikovat a přidávat nová data.

2.1. Formát vektorových dat

Mapa se skládá ze čtyř různých vrstev. Čím nižší je vrstva, tím je detailnější a obsahuje více dat. (Data byla mírně modifikována (viz. dále) a proto neodpovídají přesně datům ze zdroje – www.gps-maps.info).

- Nejvyšší (třetí) vrstva obsahuje pouze největší města, velké toky a dálnice, rychlostní silnice a silnice 1. třídy.
- Nižší (druhá) vrstva je rozšířena o silnice 2. třídy. Všechny objekty (tedy křivky) jsou zde uloženy s daleko větší přesností (tedy detailněji).
- Další (první) vrstva obsahuje navíc všechny silnice, obce, významné body zájmu, většinu toků a také některé vodní a lesní plochy. Navíc dále zpřesňuje i všechny předchozí objekty.
- Poslední (nultá) nejnižší vrstva obsahuje navíc ulice v některých městech, potoky, různé druhy ploch a velké množství bodů zájmu.

V každé vrstvě jsou data rozdělena do tří skupin (ne každá vrstva musí obsahovat všechny tři skupiny):

- POI (point of interest - bod zájmu) - patří sem vše co se dá na mapě vyjádřit jako bod (město, část obce, nemocnice, divadlo, benzinová stanice, vrcholek, skála...).
- POLYLINE (křivka) - zahrnuje objekty, které jsou na mapě značeny čarou (vodní toky, silnice, ulice).

- POLYGON (mnohoúhelník) - veškeré plochy, které je třeba na mapě zvýraznit (vodní plochy, široké řeky, lesy, území velkých obcí, hřiště, parky, průmyslové zóny...).

V každé skupině je ještě kódem určeno, do které kategorie objekt patří. Tedy např. pro POI lze určit, zda jde o hotel, skálu či benzinovou pumpu.

Data jsou uložena v textovém souboru v tomto formátu:

```
[POI]
Type=0x100
Label=Praha
Data2=(50.08667,14.42505)
[END]
```

První řádek určuje, že jde o bod zájmu, druhý řádek značí typ objektu, v tomto případě velkoměsto. Třetí řádek označuje název objektu. V případě vrcholu může být ještě za názvem výška ve stopách (ta je v programu převedena na výšku v metrech). Čtvrtý určuje zeměpisnou polohu, tedy severní šířku, východní délku. Každá jednotlivá část souřadnice má právě dvě číslice před desetinnou čárkou a pět číslic desetinných. Poslední řádek pouze zakončuje data jednoho objektu.

Pokud jde o křivku ([POLYLINE]), nebo o mnohoúhelník ([POLYGON]), je namísto jedné souřadnice uveden seznam souřadnic, kterými daná křivka prochází (popř. určuje výplň plochy).

Kapitola 3

Zpracování dat

3.1. Důvody konverze a úpravy

Získaná vektorová mapa je určena k použití v GPS přijímačích, které jsou navrženy tak, aby co nejefektivněji pracovali s tímto typem dat. Na rozdíl od toho je mobilní telefon určen k obecnému použití a jeho paměťové nároky jsou velice omezené. Proto je potřeba získaná vektorová data dále upravit tak, aby je mohl telefon zpracovávat.

3.2. Příprava dat ke zpracování

Původní data jsou rozdělena do několika souborů podle správního členění na kraje, dále nejsou již dělena. Každý soubor obsahuje všechny tři datové vrstvy. Aby bylo možno s daty pracovat jako s celkem (tedy celou republikou), je potřeba data sloučit. To provedeme programem GPSMapEdit, pomocí kterého také převedeme data z formátu pro přijímače Garmin do textové podoby (polish formát). Tímto dostaneme jeden velký textový soubor, který je dále potřeba upravit, neboť mobilní telefon není schopen s takovými soubory pracovat.

Ve druhém kroku je třeba oddělit jednotlivé vrstvy od sebe, neboť program bude pracovat vždy pouze s jednou vrstvou. Každá nižší vrstva vždy obsahuje stejná data jako vyšší, jen jsou upřesněna a rozšířena. Program pak bude sám volit, z jaké vrstvy data načte a nemusí přitom číst informace z jiných vrstev.

Vzhledem k velikosti nejvyšší (druhé a nejobecnější) vrstvy v původních datech by nebylo možno zobrazit celou mapu České republiky na display. Proto jsem se rozhodl vytvořit ještě jednu – třetí - vrstvu, která je zobecněním druhé. Byly

odstraněny silnice tak, aby zůstaly jen dálnice, rychlostní silnice a silnice 1. třídy. Všechny křivky byly navíc generalizovány, aby se snížil jejich datový objem (tím se také snížila jejich přesnost, což ale v této vrstvě nevadí). Díky této úpravě bylo však potřeba rozšířit nižší vrstvu o některá města tak, aby byla mapa při přiblížení přehledná. Z první vrstvy byla proto vygenerována velká města, která ve druhé vrstvě obsažena nebyla. Tato města byla poté do druhé vrstvy vložena.

Tímto jsou data předpřipravena tak, aby bylo možno pracovat s různými částmi odlišně (tedy zpracovat jednotlivé vrstvy zvlášť).

3.3. Zpracování jednotlivých vrstev mapy

Vzhledem k tomu, že nelze načíst celou vrstvu do paměti telefonu (výjimku tvoří jen třetí (nejvyšší vrstva)), je třeba každou vrstvu rozdělit na určité části. Nejlépe na pravidelné útvary, aby se s nimi později dobře pracovalo. Protože naprostá většina mobilních telefonů má display obdélníkového tvaru s horizontální stranou delší, rozhodl jsem se zvolit obdobný formát pro jeden díl (stránku) mapy.

Dále bylo potřeba zjistit, jak velký má být jeden díl v jednotlivých vrstvách, aby se data vešla do paměti a aby bylo zařízení schopno načítat data co nejrychleji. Tyto rozměry byly zjištěny experimentálně a byly testovány pro každou vrstvu zvlášť.

- 4. vrstva je rozdělena na 4 části (horizontálně podle 50° rovnoběžky, vertikálně podle 15° poledníku). Rozdělení bylo voleno tak, aby všechny části byly(co se týče objemu dat) podobné.
- 3. vrstva je rozdělena na 84 (6 x 14) částí (dělící mříž je mezi 12° a 19° v.d. a 51°30‘ a 48°30‘ s.š.*, krok je v obou případech 0°30‘**)
- 2. vrstva je rozdělena na 1232 (22 x 56) částí(dělící mříž je mezi 12° a 19° v.d. a 51°15‘ a 48°30‘ s.š.*, krok je v obou případech 0°7‘30“)
- 1. vrstva je rozdělena na 4928 (44 x 112) částí(dělící mříž je mezi 12° a 19° v.d. a 51°15‘ a 48°30‘ s.š.*, krok je v obou případech 0°3‘45“)

* Vzhledem k tomu, že jsou data nadále zpracovávána graficky, není použita geografická soustava souřadnic nýbrž grafická (pravý dolní kvadrant kladný) a proto tomu odpovídá i rozdělení mapy.

** Ač je horizontálně i vertikálně provedeno dělení po 0°30‘, jednotlivé části mapy tvoří obdélník, protože v oblasti České republiky je vzdálenostně délkových 30 minut méně nežli šířkových.

Kapitola 4

Grafická reprezentace

Před grafickým zpracováním dat bylo třeba zjednodušit a upravit vykreslování, neboť J2ME má v této oblasti velmi omezené možnosti. Náročnější funkce a knihovny byly z J2ME totiž odstraněny, aby bylo dosaženo co možná nejmenšího objemu dat. Nedostatky bylo v aplikaci třeba vyřešit tak, aby neměly vliv na konečný vzhled a funkčnost prohlížeče.

4.1. Hlavní nedostatky grafické knihovny a jejich řešení

1. není možno vykreslovat tlusté čáry, J2ME dovoluje pouze jednopixelové linie
2. nelze vykreslovat plné plochy
3. text může být zobrazen vždy pouze vodorovně
4. nemožnost podbarvení textu
5. vykreslení nestandardních čar

Řešení:

1) Vzhledem k tomu, že ke zdůraznění některých linií (dálnice, hlavní silnice…) je třeba použít širší čáru, a J2ME to nepovoluje, bylo ke zdůraznění použito vykreslení několika čar vedle sebe posunutých o jeden obrazový bod (v horizontálním nebo vertikálním směru). Tato metoda se může zdát příliš těžkopádná, ale vzhledem k rychlosti vykreslování úseček nemá téměř vliv na konečnou rychlost aplikace.

2) Plné plochy jsou v mapě velmi potřebné. Ať už jde o vykreslení jezera, lesa či obydlené oblasti, vždy jde o významnou plochu, která velmi zpřehledňuje mapu. Protože J2ME nemá prostředky k vykreslování ploch, byla použita externí komponenta JMicroPolygon* jež umožňuje vykreslování plných ploch. Bohužel vykreslení plné plochy touto komponentou přináší zpomalení při překreslování scény. I přesto je však velice důležitá a potřebná. Určitou variantou by mohla být komponenta Mobile 3D Graphics**. Ta je ovšem implementována jen v malém množství mobilních telefonů a proto by mapa nebyla kompatibilní se všemi telefony.

   * JMicroPolygon je knihovna umožňující vykreslení plochy za pomocí jediné možné funkce, která je schopna vykreslit plný obrazec – vykreslení trojúhelníku. Vzhledem k tomu, že polygony v mapě tvoří i několik stovek bodů, je vykreslování touto metodou pomalé. Jiná možnost bez použití „standardní“ Mobile 3D Graphics knihovny však není.

   ** Mobile 3D Graphics (M3G) je standardizovanou grafickou knihovnou rozšiřující J2ME (JSR-184). Tato knihovna je graficky velice výkonná. Ač se očekává její masové nasazení, dnes tomu tak ještě zdaleka není.

3) Při zobrazování mapy většina aplikací obvykle vykresluje název křivky (řeka, ulice…) rovnoběžně s křivkou čímž se vytváří větší přehlednost mapy. Možnost zobrazit text pouze vodorovně se nezdá být až tak velkým problémem. Ve většině případů ani není třeba vykreslovat text rovnoběžně s linií. Problém nastává především v hustě prokreslených oblastech (především centra měst), kde rovnoběžné zobrazení textu a křivky dává jasně najevo, ke které linii nápis patří. Bohužel tento nedostatek nelze v J2ME nijak odstranit. Jedinou možností by byla vlastní grafická reprezentace alfanumerických znaků v několika různých úhlech (otáčení obrázků též není možné). Pak by bylo třeba tyto znaky (jednotlivé obrázky) poskládat do slov a následně vykreslit na scénu. Tato operace by ale

v některých případech mohla výrazně prodloužit dobu vykreslení scény a proto není metoda použita.

4) V některých případech je potřeba text zvýraznit podbarvením (např. číslo silnice). Na rozdíl od většiny standardních grafických knihoven nedává ta v J2ME možnost získat rozměr vykreslovaného textu. Možností vyřešení tohoto problému by mohlo být zjištění velikosti jednotlivých písmen a následné vypočítání délky (popř. výšky) plochy obsahující text. To však není zcela jednoduché a to hned ze dvou důvodů. Mezi dvěmi různými písmeny nemusí být stejná mezera, takže by bylo obtížné zjistit velikost mezer mezi všemi znaky v názvu linie. Druhým důvodem je to, že standard J2ME neudává přesnou velikost jednotlivých znaků a je tedy jen na výrobci zařízení, jaké znaky (font, velikost) přesně použije. Z těchto důvodů je tedy plocha odhadnuta z délky řetězce. Proto se může stát, že podbarvená plocha nebude přesně odpovídat zobrazenému textu.

5) Železnice má na mapách svůj specifický vzhled. Je to tlustá černo-bílá čára. Takovéhoto vzhledu však nelze za pomocí grafické knihovny v J2ME dosáhnout. Jediné možné řešení by bylo vypočítat pro každou úsečku ještě jakési mezibody, které by tvořily střídání mezi černou a bílou čarou. Navíc by bylo zapotřebí vykreslovat tyto krátké úsečky několikrát, aby vznikla tlustá čára (podobnou technikou jako je popsáno výše). Tato technika by však opět vyžadovala více času (místo jedné nebo několika málo úseček by se jich vykreslovaly stovky) a proto není použita. Místo ní je použita jiná, mnohem jednodušší technika, která také vytvoří čáru podobající se železniční trati.

   Snad jediné rozšíření pro linii je vykreslení čárkované čáry, ta sice ještě není podobná trati, ale pokud je pomocí ní vytvořena tlustá čára (vykreslení základní souřadnice, posun o 1px horizontálně a posun o 1px vertikálně), vznikne linie výrazně podobná železniční trati. Tato metoda používá tři linie namísto jedné. To téměř nijak neprodlouží metodu vykreslování.

4.2. Popis objektů mapy

Jak bylo popsáno již výše, mapová data se dělí na body zájmu, křivky a polygony. Ty jsou dále děleny podle typu (typy jednotlivých druhů objektů nemusí být nutně různé, i přesto však nevyjadřují podobný objekt). Každý jednotlivý typ má své grafické znázornění, pro body to jsou obvykle ikony, pro křivky specifická čára a pro polygon barva. Vše je v mapě zobrazeno tak, aby běžný uživatel intuitivně rozeznával jednotlivé objekty.

Ikony jednotlivých bodů jsou reprezentovány PNG obrázky o rozměrech cca 12x12 pixelů. Formát PNG byl zvolen především proto, že je to jediný obrazový formát, který je podporován normou. Dále také proto, že je možno vytvořit průhledné pozadí. Díky tomu lze ikony vykreslovat přes sebe nebo přes polygony aniž by zakrývaly větší plochu nežli je nezbytně nutné. Na barevném pozadí pak nevznikají bílé obdélníky s ikonou. Aplikace obsahuje přes padesát grafických ikon a všechny jsou mým vlastním dílem. Vytvořil jsem je sám především proto, abych neporušil autorská práva kopírováním, ale také proto, že takovýchto ikon je nedostatek a není jednoduché je obstarat.

Linie jsou tvořeny buď jednoduchou čarou různé barvy popř. čárkovanou čarou nebo technikou popsanou v 4.1. bodu prvním. Barvy linií a ploch jsou voleny z bezpečné palety barev (Browser-safe palette). Tu sestavila Lynda WEINMAN a zmiňuje se o ní ve své knize [1]. Paleta byla primárně sestavena pro webové prohlížeče, které nebyly schopny zobrazit všechny odstíny 24 bitové hloubky barev. Ale vzhledem k tomu, že 24 bitovou hloubku barev neumí zobrazit také naprostá většina dnešních mobilních telefonů, lze tuto paletu uplatnit i zde. Mobilní telefon umí sice zobrazit několik tisíc barev, ale pokud má vykreslit barvu, která je mimo jeho rozsah, přikloní se k barvě co nejvíce podobné a tu zobrazí. Může se ale stát, že dvě barvy lišící se v 24 bitové reprezentaci můžou být na display telefonu zobrazeny shodně. Tomu předchází paleta bezpečných barev, která obsahuje 216 odstínů, které by měl být telefon schopen zobrazit.

V následující části 4.3. je seznam všech typů bodů zájmu, linií a ploch. Body zájmu jsou pro přehlednost rozděleny do podskupin a je u nich vyobrazena ikona,

hexadecimální kód, název a popřípadě ještě upřesňující popisek. U křivek je znázorněna čára, dále popis pro který objekt je čára použita, uvedena barva (hexadecimálně a v RGB formátu) a ještě je stručně popsán vzhled čáry. Podobně je tomu tak i u seznamu ploch.

4.3. Souhrn použitých objektů

Seznam bodů zájmu:

Stravování a ubytování

	0x2A00	restaurace	
	0x2A07	občerstvení	kiosek, fast food,
	0x2A0A	pizzerie	
	0x2A0C	gril	
	0x2A0D	pekárna, cukrárna	
	0x4500	restaurace	blíže nespecifikovaná
	0x2B00	hotel	
	0x2B01	motel	
	0x2B02	hostinec	
	0x2D02	bar, nightclub	
	0x2E00	nákupy	
	0x2E02	potraviny	
	0x2E04	nákupní centrum	
	0x2F0F	Garmin dealer	

Služby

	0x2F00	služby	bez upřesnění / upřesněno v názvu
	0x2F06	bankomat	banka uvedena obvykle v názvu
	0x2F05	pošta	
	0x640F	pošta	
	0x3002	nemocnice	

	0x6403	hřbitov	hřbitov / urnový háj / hrobka
	0x3001	policie	
	0x2E05	lékárna	
	0x4C00	informace	místní informační středisko
	0x5100	telefon	

Doprava

	0x2F0B	parkoviště	normální, placené, zastřešené, záchytné…
	0x2F01	benzinová pumpa	zahrnuje i LPG stanice
	0x2F08	hromadná doprava	bus / tramvaj / trolejbus / vlak
	0x2F02	auto půjčovna	
	0x2F07	autosalon	prodej nových automobilů
	0x5904	heliport	přistávací plocha pro vrtulníky
	0x2F04	letiště	letiště lokálního i mezinárodního využití
	0x2F03	autoopravna	
	0x6401	most	křížení dálnic, nad řekou, železniční
	0x4300	přístav	kotviště / půjčovna lodí

Kultura

	0x2D01	divadlo	
	0x2C02	muzeum	
	0x2C03	knihovna	

Stavby

	0x6404	kostel	kostel, mešita
	0x6411	věž	věž / rozhledna historického i moderního rázu
	0x6402	budova	historická nebo jinak významná budova
	0x6415	hrad	hrad / město s historickým centrem
	0x1602	radio vysílač	
	0x3003	radnice	historická radnice / sídlo městského úřadu
	0x2c05	škola	

Sport a turistika

	0x5400	koupaliště	
	0x2D09	koupaliště	
	0x6412	turistická stezka	značená i neznačená
	0x2D0A	fitnes	posilovna / aerobik / tělocvična
	0x2D06	vlek	vlek / lanovka / lyžařský areál
	0x2B03	kemp	
	0x6406	křižovatka	křížení značených turistických cest
	0x4100	rybaření	rybářská oblast / region
	0x6414	zdroj pitné vody	

Zábava

	0x6604	pláž	
	0x2C07	ZOO	Zoo / safari / obří akvárium
	0x2D03	kino	

Příroda

	0x6612	rezervace	přírodní rezervace, či jinak chráněná oblast
	0x6614	skála	
	0x640E	park	
	0x6511	pramen	
	0x2C0A	vinice	
	0x6616	vrch	vysoký kopec / hora , nadmořská výška (m)
	0x2C00	zajímavost	
	0x5200	vyhlídka	výhled na krajinu / zajímavý úkaz
	0x2C04	zajímavost	orientační bod / mezník / památka

Geografické pojmy

	0x1e00	část obce	(bez bodu)
	0x2800	městská část	(bez bodu)
	0x3006	hranice	hraniční přechod
	0x100	velkoměsto	město nad 1mil. obyvatel

0x200	velkoměsto		(500-1 mil)
0x300	velkoměsto		(100-500)
0x600	střední město		(10-100)
0x700	střední město		(5-10)
0x800	malé město		(2-5)
0x900	malé město		(1-2)
0xA00	vesnice		(500-1000)
0xB00	vesnice		(200-500)
0xC00	osada		(100-200)
0xD00	osada		(<100)

Křivky:

0x1	dálnice	#993300	RGB(153,51,0)	(3px, hnědá)
0x2	silnice 1.třídy	#339933	RGB(51,153,51)	(2px, zelená)
0x3	dopravní tepny	#339933	RGB(51,153,51)	(2px, zelená)
0x4	silnice 2. třídy	#000000	RGB(0,0,0)	(2px, černá)
0x5	místní komunikace	#000000	RGB(0,0,0)	(1px, černá)
0x6	ulice	#000000	RGB(0,0,0)	(1px, černá)
0x7	soukromá cesta	#666666	RGB(102,102,102)	(1px, tmavě šedá)
0x8	nájezd na dálnici	#000000	RGB(0,0,0)	(1px, černá)
0xA	nezpevněná cesta	#CCCCCC	RGB(204, 204, 204)	(1px, světle šedá)
0xC	kruhový objezd	#000000	RGB(0,0,0)	(1px, černá)
0x14	železniční trať	#000000	RGB(0,0,0)	(2px, černá, čárkovaná)
0x16	turistická stezka	#000000	RGB(0,0,0)	(1px, černá, čárkovaná)
0x1F	řeka / potok	#0000FF	RGB(0,0,255)	(1px, modrá)
0x27	runway	#993300	RGB(153,51,0)	(3px, hnědá)

Plochy:

0x1	město	#CC9999	RGB(204,153,153)	(hnědá)
0x3	zastavěná oblast	#999966	RGB(153,153,102)	(khaki)
0x4	vojenský prostor	#666666	RGB(102,102,102)	(tmavě šedá)
0x5	parkoviště	#CCCCCC	RGB(204,204,204)	(světle šedá)

	0x6	garáže	#CC9900	RGB(204,153,0)	(okrová)
	0x7	letiště	#FFCC99	RGB(255,204,153)	(světle oranžová)
	0x8	nákupní centrum	#FFCC33	RGB(255,204,51)	(oranžová)
	0xA	univerzita / škola	#FF9999	RGB(255,153,153)	(růžová)
	0xB	nemocnice	#FF9999	RGB(255,153,153)	(růžová)
	0xC	průmyslová zóna	#669999	RGB(102,153,153)	(modro zelená)
	0xE	runway	#CCCCCC	RGB(204,204,204)	(šedá)
	0x14	národní park	#99CC99	RGB(153,204,153)	(světle zelená)
	0x17	městský park	#339900	RGB(51,153,0)	(tmavě zelená)
	0x19	sportovní areál	#FF9900	RGB(255,153,0)	(tmavě oranžová)
	0x1A	hřbitov	#00FF00	RGB(0,255,0)	(zelená)
	0x1F	CHKO / les	#99CC99	RGB(153,204,153)	(světle zelená)
	0x3C	velká vodní plocha	#0066FF	RGB(0,102,255)	(modrá)
	0x40	malá vodní plocha	#0066FF	RGB(0,102,255)	(modrá)
	0x41	malá vodní plocha	#0066FF	RGB(0,102,255)	(modrá)
	0x46	velká řeka	#0066FF	RGB(0,102,255)	(modrá)
	0x4B	pozadí	#FFFFFF	RGB(255,255,255)	(bílá)
	0x53	bažina, mokřady	#66CCCC	RGB(102,204,204)	(světle modrá)

Kapitola 5

Programová část aplikace

5.1. Základní popis

Celý program je tvořen třemi základními třídami:

- Main – tato třída se stará pouze o rozběhnutí aplikace
- Mapa – zde běží hlavní smyčka programu, dále má tato třída metody na vykreslování načtených dat a k jejich dalšímu zpracování
- Vlakno – stará se o korektní a efektivní načtení potřebných dat a o jejich zpracování

a dále jednoduššími třídami, které pouze nesou informaci o některém z geografických objektů:

- crds – ukládá zeměpisnou šířku a délku
- poi – obsahuje všechny informace o bodu zájmu
- po – slouží k uložení křivky či plochy (všech jejich atributů)

5.2. Běh aplikace

Po spuštění se inicializují potřebné hodnoty, načtou se obrázky a začne běžet hlavní smyčka. Jeden cyklus proběhne po inicializaci, další vždy poté co nastane jedna z následujících událostí:

- uživatel stiskem klávesy zadá požadavek o překreslení scény
- aplikace dokončila načítání dat ze souboru a ta je teď potřeba vykreslit

Aby byla aplikace rychlá a uživatelsky přívětivá, je třeba soubory s potřebnými daty načítat v jiném vlákně nežli běží smyčka programu. To zajistí, aby aplikace reagovala

na požadavky i přesto, že načítá soubor. Reakce na požadavek a samotné vykreslení scény se tím sice zpomalí, ale i tak je pro uživatele lepší, když vidí, že mapa pracuje, než kdyby aplikace vždy „zamrzla“ do doby než načte potřebná data.

Pokud zadá uživatel požadavek na posun mapy, posune se střed souřadnic do požadované polohy a následně se zjistí, zda načtená data postačují k vykreslení celé scény. Pokud ano, scéna se vykreslí. Pokud by však některá data chyběla, je potřeba je načíst. Smyčka tedy spustí ve vedlejších vláknech načítání potřebných souborů a poté se vykreslí data získaná ze souboru.

I pokud se uživatel nad mapou pohybuje velmi rychle, jsou načítána pouze data, která jsou aktuálně potřebná. Jestliže se v době načítání dat změní poloha tak, že už data nebudou potřeba, načítání je ihned přerušeno, aby nebránilo rychlejšímu běhu aplikace.

Jestliže uživatel požádá aplikaci o přiblížení / oddálení mapy, můžou nastat tři situace. První dvě jsou shodné s popisem výše. Třetí situace znamená načtení jiné vrstvy mapy, než se kterou bylo doposud pracováno. Tato operace je vždy časově náročnější, neboť je potřeba načíst všechna data znovu z nové vrstvy. Předtím než je dokončeno načtení nových dat, je k vykreslení mapy použito dat starých.

V případě přiblížení jsou zprvu vykreslená data méně přesná, která neobsahují všechny objekty. Teprve po načtení dat z nové vrstvy dojde k překreslení scény a tím k přesnějšímu zobrazení mapy a vykreslení více objektů. V opačném případě, když chce uživatel mapu oddálit, jsou vykreslena všechna data doposud načtená. Obsahují tedy mnohem více objektů nežli aktuální vrstva. Data však nejsou schopna pokrýt celou scénu a proto do doby než jsou načtena nová data, zobrazuje se jen část mapy.

Opět zde platí, že se načítají jen právě potřebná data a všechna ostatní jsou z paměti odstraněna nebo je jejich načítání zrušeno.

5.3. Třída poi

Zde se ukládá načtený bod zájmu. Třída obsahuje všechny informace o bodu. Malou změnou oproti původním datům je, že souřadnice není uložena pomocí desetinného čísla, tedy pomocí typu double (40B), ale pomocí integeru (4B). To značně šetří paměť potřebnou na uložení dat. Tuto úpravu lze použít, neboť odstraněním desetinné čárky (úprava je ekvivalentní vynásobení souřadnice číslem 100000) vznikne číslo v řádu několika milionů. Integer svou kapacitou tuto hodnotu několikanásobně převyšuje, proto není problém uložit souřadnici takto a dále s ní pracovat jako s celým číslem a pouze v případě potřeby vydělit číslem 100000.

5.4. Třída po

Křivka nebo plocha je o něco složitější útvar než pouhý bod, základ je však naprosto stejný. Výrazným rozdílem je, že namísto jedné souřadnice je třeba uložit seznam bodů, kudy křivka probíhá. Ten je reprezentován vektorem souřadnic (crds).

I zde je využito ukládání souřadnic pomocí celého čísla namísto desetinného. Vzhledem k tomu, že křivky tvoří více než 80% všech dat mapy, je zde paměťová úspora velice vysoká.

Většina křivek i ploch má vlastní pojmenování, to je potřeba vykreslit společně s křivkou (plochou). Zde se křivka od plochy lehce liší. Výpočet pozice textu je pro oba objekty rozdílný. Uložení je však opět shodné, tedy pomocí jedné souřadnice (crds).

Aby bylo evidentní, ke které křivce nápis v mapě patří, je třeba ho vykreslit alespoň přibližně uprostřed křivky. Jestliže nechceme, aby se „střed“ počítal při každém překreslení, je dobré si ho vypočítat a uložit ihned při načítání dat. Každá instance třídy poi tedy nese i informaci o středu křivky.

5.5. Třída Vlakno

Jak už bylo výše naznačeno, tato třída slouží k načtení dat ze souboru a k jejich zpracování. Třída Vlakno je rozšířením třídy Thread a může tedy celá vystupovat jako vlákno. Má tyto důležité prvky:

Proměnné

- vektor bodů zájmu
- vektor křivek
- vektor ploch
- příznak ke zrušení vlákna
- ostatní proměnné viz. kód programu

Metody

- String getMtHeight(String) – převod nadmořské výšky ze stop do metrů
- crds getLineLabelPosition(Vector v) – výpočet pozice názvu křivky
- crds getPolygonLabelPosition(Vector v) – výpočet pozice názvu plochy
- Konstruktor – parametrem je název souboru a ukazatel na instanci třídy Mapa, jež spustila třídu Vlakno
- Metoda run

Pomocí konstruktoru je zadán název souboru, který má vlákno načíst a zpracovat. Po spuštění vlákna z hlavní smyčky začne třída načítat zadaný soubor. Vzhledem k tomu, že čtení znak po znaku ze souboru a jeho následné zpracování není příliš efektivní postup, je k načítání ze souboru použit zásobník (ByteBuffer). Jeho použití velice urychlí práci se souborem, neboť je schopen přečíst v krátkém čase velké množství bytů. Zásobník je možné zvolit libovolně velký, je ale třeba velikost volit s ohledem na běh celé aplikace a také na množství použitelné paměti. Abych dosáhl

optimální velikosti zásobníku, provedl jsem několik testů s různě dlouhým zásobníkem. Testy byly provedeny na různých typech mobilního telefonu a také na několika odlišných emulátorech.

Ideální se jevila velikost 4096 bytů. Až do této velikosti jevil program zrychlení načítání, při dalším zvětšování zásobníku se načítání zrychlovalo už pozvolna. V některých případech se dokonce projevovalo i prodloužení času načítání. Toto připomíná Beladyovu anomálii popsanou roku 1969 v knize [2].

4096 bytů, tedy 4kB je dost malé množství než aby zahltilo operační paměť telefonu. Vždy po zpracování celého zásobníku vlákno uvolní procesor ostatním vláknům.

Z ByteBufferu jsou data dále již zpracovávána po znacích. Lépe řečeno po celých řádcích. Ze zásobníku jsou totiž data kopírována znak po znaku a je z nich vytvářen řetězec. Ten je pak teprve zpracován. K urychlení kopírování dat mezi ByteBufferem a řetězcem nám poslouží další zásobník, tentokrát StringBuffer. Ten je totiž uzpůsoben k tomu, aby do něho šel přidávat vždy jeden znak. Klasický String, tedy řetězec znaků má tuto možnost také, není však na tuto činnost optimalizován. Využitím StringBufferu tedy ušetříme další čas při zpracovávání souboru.

Vždy, když je do StringBufferu načten jeden řádek (detekováno znakem ‘\n‘), je StringBuffer převeden na String a ten je pak podle obsahu zpracován. Z pohledu na formát dat (2.2.) je vidět, že je třeba rozlišit 6 řádků. První určuje druh objektu, druhý jeho typ, třetí je jeho názvem (ne vždy musí být obsažen), na dalším řádku jsou data potřebná pro vykreslení, 5. a 6. řádek neobsahuje žádnou informaci (5. řádek obsahuje pouze tag [END], 6. řádek je prázdný).

Metoda getMtHeight

public static String getMtHeight(String lab) – Pokud řetězec (obsahující název hory) neobsahuje formátovací podřetězec „~[0x1f]“ za nímž by měla následovat výška, je vrácen původní string (název vrchu neobsahuje údaj o nadmořské výšce). Pokud je formátovací podřetězec nalezen, je vstupní řetězec rozdělen na tři části. První část obsahuje název vrcholu, druhá formátovací řetězec a třetí výšku ve

stopách. Výška je tedy vydělena koeficientem 3.28 (převod na metry) a následně je název s výškou sloučen do jednoho řetězce. Ten je nakonec použit jako výstupní hodnota.

Metoda getLineLabelPosition

public static crds getLineLabelPosition(Vector v) – vstupem je seznam bodů (souřadnic), kterými prochází křivka, výstupem souřadnice místa, kde má být zobrazen název křivky. Pro výpočet se použije následující algoritmus: nejdříve je zjištěno jestli krajní body křivky mají vzdálenost větší horizontálně či vertikálně. Uvažujme například, že horizontální vzdálenost je větší (tedy křivka je více vodorovná). Najdeme střed této vzdálenosti a následně ze seznamu bodů vybereme dva sousední takové, kde každý bod leží na jiné straně od středu. Úsečka tvořená těmito body je střední úsečkou celé křivky. Nakonec je nalezen její střed a ten je vrácen jako souřadnice (crds).

Metoda getPolygonLabelPosition

public static crds getPolygonLabelPosition(Vector v) – vstupním parametrem je seznam souřadnic, jež tvoří plochu, výstupem souřadnice místa, kde bude zobrazen název plochy. Jednotlivé souřadnice středu vypočteme tak, že spočítáme průměr všech souřadnic tvořících plochu. Tedy provedeme výpočet zvlášť pro x-ovou a y-ovou souřadnici. Průměr pro každou souřadnici pak tvoří pozici na níž bude vykreslen název objektu.

5.6. Zpracování jednotlivých řádků třídou Vlakno

Po předání řetězce ke zpracování je podle jeho prefixu rozhodnuto, jak bude dál upraven a jaké další postupy budou provedeny.

- Jestliže je řádek prázdný, nebo příliš krátký, je ignorován.

- Pokud je některé slovo z {„[POI“, „[POLYL“, „[POLYG“} prefixem řádku, je pro následující data nastaven příslušný druh tak, aby byla korektně zpracována.
- Když je prefixem „Typ“, je String oříznut a postfix nesoucí informaci převeden z hexadecimálního tvaru do integeru. Tím je nastaven typ objektu.*
- Řádek s prefixem „Lab“ určuje název objektu. Jeho zpracování je jednoduché, pouze se odřízne prefix a zbytek tvoří název objektu. Výjimku tvoří pouze bod zájmu vrch (dle tabulky 4.3. typ 0x6616). Zde totiž řádek neobsahuje pouze název objektu, ale také jeho nadmořskou výšku. Ta je bohužel udána ve stopách, což by pro většinu uživatelů byla nedostatečná informace. Celý název je tedy předán metodě getMtHeight(String) a ta vrátí název hory již s upravenou nadmořskou výškou.
- Posledním řádkem nesoucím informaci je řádek s prefixem „Data“. V původním formátu byla data ještě rozdělena na Data0, Data1, Data2. O toto se však nemusíme starat, neboť máme již data rozdělena a předpřipravena (viz. Kapitola 3). Další zpracování se liší podle toho zda jde o bod zájmu, křivku či plochu.

* Typ objektu by bylo možno uchovávat též v řetězci znaků. Datový objem by nebyl příliš rozdílný, někdy dokonce menší. S typem short se však lépe pracuje. Dvě instance lze rychleji porovnat na shodu, je možné ho použít ve switch příkazu. Navíc lze i do kódu vkládat celé číslo v hexadecimální podobě, takže není problém s opačnou konverzí při testování na shodu.

Zpracování bodu zájmu (POI)

Tento objekt je nejjednodušším (obsahuje pouze jedinou souřadnici) a proto i jeho zpracování a uložení bude jednodušší oproti ostatním dvěma. Nejdříve je potřeba

z řetězce s prefixem „Data“ extrahovat zeměpisnou souřadnici bodu. Vzhledem k tomu, že formát dat je přesně určen, vytvoříme dva substringy, které obsahují zeměpisnou šířku a délku. Nyní je potřeba z řetězce odstranit desetinnou čárku, čímž se nám změní číslo z reálného na celé (důvod, užitečnost a korektnost popsána v 5.3.).

Zpracování křivky (POLYLINE)

Jedná-li se o křivku, je třeba přečíst celý seznam souřadnic. Postup je podobný jako v předchozím případě, je však potřeba načítat souřadnice do doby než nastane konec řetězce. Souřadnice jsou pak vkládány do vektoru „list_crds“ (seznam souřadnic) třídy „po“.

Došlo zde také k jednomu vylepšení. Protože jednu křivku lze uložit pomocí více menších křivek, lze tyto křivky také nazpět „slepit“. Pokud tedy v souboru ihned po sobě následují křivky, kde druhá přímo navazuje na první, jsou slepeny a je z nich vytvořena jedna křivka. Toto lze uplatnit až v 20%. Paměť je pak ušetřena vynecháním redundantních dat. Tedy především název křivky a typ křivky bude uložen jednou namísto dvakrát. Dále poslední a první body jsou ekvivalentní a tedy je stačí uložit pouze jedenkrát.

Nakonec je ještě potřeba vypočítat bod pro pozdější vypisování názvu křivky. To je umožněno pomocí metody getLineLabelPosition(). Střed je však vypočítáván jen pro křivky, které mají nějaký název, pro ostatní by to nemělo smysl a zbytečně by se spouštěla funkce, jejíž výsledek by do aplikace nepřinesl žádné informace.

Zpracování plochy (POLYGON)

Protože plocha je uložením dat shodná s křivkou, je použit i obdobný algoritmus na její načtení. Rozdílem je to, že plochy nelze spojovat stejně jako křivky a výpočet souřadnic pro výpis názvu plochy probíhá jinak. Je tedy namísto metody

getLineLabelPosition() spuštěna metoda getPolygonLabelPosition(). I ta je však spouštěna jen tehdy pokud má plocha nějaký název.

Výstup třídy Vlakno

Každý z objektů je vždy po načtení vložen do příslušného vektoru, čímž jsou vytvořeny tři vektory nesoucí informaci celého jednoho souboru. Aby instance třídy byla schopna po načtení celého souboru zobrazit načtený obsah na display, dostala v konstruktoru ukazatel na třídu, jež tuto instanci spustila. Jakmile tedy třída načte soubor, může ho vykreslit na display.

Pokud chce někdo z vnějšku přinutit Vlakno, aby zanechalo své činnosti a přestalo načítat soubor, stačí když vnitřní veřejnou proměnnou „destroy“ nastaví na true. To zapříčiní, že je přerušen cyklus v němž probíhá načítání souboru. Soubor je však uzavřen a všechna práce ve vlákně vyjma načítání je dokončena. Obdobným řešením by mohlo být využití standardní funkce Thread.interrupt(). Ač by tuto metodu měly telefony podporovat, není tomu tak. Metoda se mi jevila jako nefunkční. Stejný názor měly i přispěvatelé do fora o vývoji v jazyce J2ME.

5.7. Třída Mapa

Struktura třídy

Mapa je potomkem třídy Canvas, to nám dovoluje vykreslovat data na display. Třída je rozšířena tak, aby v sobě mohla mít smyčku (Runnable), jenž bude tvořit hlavní program a obsluhovat události. Dále je ještě rozšířena o CommandListener. To proto, aby bylo možné reagovat na stisknuté klávesy.

Proměnné:

- bits0, bits1, bits2, bits3 – bitové mapy pro každou vrstvu dat, obsahují informaci zda je soubor načten v paměti či nikoliv.

- t0,t1,t2,t3 – vlákna, která načítají části mapy
- pols0, pols1, pols2, pols3 – vektory, v nichž jsou uloženy plochy načtené ve vláknech
- pos0, pos1, pos2, pos3 – obdobně pro křivky
- pois0, pois1, pois2, pois3 – obdobně pro body zájmu
- nula – zde je uložen počátek souřadnic vzhledem k zeměpisné délce a šířce (vynásobeno číslem 100000, aby odpovídalo uloženým souřadnicím)
- krok – kolik úhlových stupňů odpovídá jednomu bodu na displayi (vynásobeno číslem 100000, aby odpovídalo uloženým souřadnicím)
- další proměnné v kódu

Metody:

- konstruktor – zde probíhá inicializace a načtení ikon
- run() – tvoří hlavní smyčku programu
- draw_coord(Graphics) – zobrazení poledníků, rovnoběžek a jejich souřadnic
- getStringDegrees(double, boolean) – převod reálného čísla na stupně a minuty
- draw_poi(Graphics, Vector) – vykreslení bodů zájmu na display
- draw_lines(Graphics,Vector) - vykreslení křivek
- draw_poly(Graphics, Vector) – metoda k vykreslení ploch
- getDegreeLength(double) – počítá délku jednoho rovnoběžkového stupně
- paint(Graphics) – obstarává vykreslování na display
- keyPressed(int) – reaguje na stisk klávesy
- keyRepeated(int) – reaguje na držení klávesy
- commandAction(Command, Displayable) – zpracovává příkazy kontextových kláves

Po spuštění třídy Mapa se provedou instrukce zapsané v konstruktoru. Nejprve se aplikace přepne do režimu, kde je celá plocha displaye využita pro vykreslovanou scénu. Dále se všechny bitové mapy nastaví na false, vytvoří se instance všech vláken, nastaví se hranice pro přechod mezi jednotlivými vrstvami mapy a je zadán úhel který odpovídá jednomu pixelu. Nakonec je ještě přidán příkaz, díky kterému je možno ukončit aplikace kontextovým tlačítkem.

Po inicializaci proměnných pomocí konstruktoru je spuštěna metoda run(), ta se stará o překreslování scény. Metoda čeká na stisk klávesy a pak zavolá metodu paint(), která vykreslí nová data. Metoda paint() je také nepřímo volána ze třídy Vlakno při načtení souboru. Jindy už překreslení nenastane.

Metoda getStringDegrees

static public String getStringDegrees(double, boolean) – metoda slouží k úpravě čísla do podoby úhlů, minut úhlů a v případě potřeby ještě i tisícin minut. Protože je metoda využívána pro úpravu hodnot před zobrazením, je návratovou hodnotou String. Metoda má dvě možnosti výstupu, v případě zadání hodnoty true do druhého parametru vrátí metoda i tři desetinná místa minut. V opačném případě pouze celé minuty.

Metoda drawClock

static private void drawClock(Graphics, int, int) – metoda vykreslí přesýpací hodniny na danou scénu. Vstupní hodnoty x a y jsou středem hodin, měly by tedy odpovídat středu displaye.

Metoda draw_coord

private void draw_coord(Graphics) – parametrem metody je ukazatel na scénu, která bude nakonec vykreslena na display. Úkolem metody je na display vykreslit korektně souřadnice a jejich přesnou hodnotu. Souřadnice jsou vykreslovány tak, aby mezi dvěma sousedními bylo alespoň 50 pixelů rozestup. Vždy je nalezena nejkrajnější zobrazitelná souřadnice a k ní je vždy připočtena hodnota taková, aby odpovídala

kroku nejméně 50px. Toto pokračuje až do doby, než hodnota dosáhne konce scény. Protože jsou souřadnice vypočítávány přímo ze zeměpisných souřadnicích (vynásobených 100000), není problém zjistit jejich přesnou hodnotu. Tu je pak ještě třeba upravit do správného formátu (úhly a minuty úhlu). K tomu nám slouží funkce getStringDegrees(double, boolean). Protože nechceme zobrazovat desetinná místa, jako druhý vstupní parametr zadáme false.

Metoda draw_poi

private void draw_poi(Graphics g, Vector v) – jako parametr dostane scénu na kterou bude vykreslovat a seznam bodů, které je třeba vykreslit. Pro každý bod provede následující operace:

- Transformace souřadnic ze zeměpisných do grafických – to lze za pomocí proměnných krok (určuje kolik stupňů je jeden pixel), nula (počátek souřadného systému vzhledem k zeměpisným souřadnicím) a pomocí zeměpisné souřadnice bodu. Nyní už víme kam se bude vykreslovat, nevíme však co.
- Většina bodů zájmu jsou lokální objekty a proto není třeba je vykreslovat již při malém měřítku. Výjimku tvoří obce a vrcholy. Obce jsou vykreslovány v závislosti na měřítku a velikosti obce. Vrcholy jsou zobrazovány již při menším měřítku než ostatní objekty, protože je možné je spatřit již z dálky. Ostatní body zájmu jsou zobrazovány až při dostatečně velkém měřítku. Navíc jsou nejdříve zobrazovány bez názvu a až následně (po přiblížení mapy) je zobrazen jejich název. To proto, aby v oblasti, kde je hodně bodů byla mapa přehledná.
- Nakonec, pokud je třeba, je vykreslen název objektu. Pozice textu je určena souřadnicí bodu.

Metoda draw_lines

private void draw_lines(Graphics, Vector) – vykreslení křivky znamená vykreslení úseček, které jsou tvořeny seznamem bodů, jimiž křivka prochází. Protože metoda

dostane jako parametr seznam křivek, je třeba pro vykreslení všech úseček použít dva cykly. Vnitřní přes seznam bodů a vnější přes seznam křivek.

Ve vnitřním cyklu je prováděno vykreslování úseček podle jejich typu. Nejdříve je nastavena barva, v případě potřeby i čárkovaná čára, a následně je provedeno vykreslení úsečky (v některých případech více úseček vedle sebe).

Po skončení vnitřního cyklu je ještě potřeba vykreslit název křivky (za předpokladu, že nějaký má). I v případě, že křivka název má, nemusí být ještě vykreslen. Pokud je totiž křivka zobrazovaná na scénu příliš krátká, její název se pro přehlednost nezobrazí. Pro určení zda název vykreslit se spočítá horizontální a vertikální vzdálenost krajních bodů. Pokud by na scéně nepřesáhla ani jedna ze vzdáleností velikost 50 pixelů, název vykreslen nebude. Pokud má být název zobrazen, transformují se souřadnice středu křivky na souřadnice displaye a zde je název vykreslen. Pokud se jedná o některou z významných komunikací (dálnice, rychlostní silnice, silnice 1. a 2. třídy), je text podbarven žlutým obdélníkem. V ostatních případech je vykreslen pouze text.

Metoda draw_poly

private void draw_poly(Graphics ,Vector) – tato metoda se nestará přímo o vykreslení polygonu, neboť J2ME nemá prostředky na jeho vykreslení. (problém je popsán v 4.2. bod 2). Je potřeba použít k vykreslení funkci PolygonGraphics.fillPolygon(Graphics, int[], int[]) z knihovny JMicroPolygon. Tato funkce má tři parametry, prvním je komponenta, na kterou lze vykreslovat, druhým a třetím parametrem jsou pole souřadnic. V prvním je horizontální, v druhém vertikální složka.

Protože jsou všechna data uložena ve vektoru a navíc jsou reprezentována pomocí upravených zeměpisných souřadnic, je třeba data před spuštěním funkce upravit a vložit do dvou různých polí. Také je podle typu potřeba nastavit barvu výsledného polygonu. Pak už je možno funkci spustit. Po skončení funkce je ještě potřeba vykreslit název polygonu (jestliže nějaký má). To se provede transformací již spočítaných souřadnic do souřadnic displaye.

Metoda getDegreeLength

public static double getDegreeLength(double) – protože s měnící se zeměpisnou šířkou se mění i vzdálenost mezi poledníky, je třeba tuto vzdálenost korektně vypočítat. Vzdálenost je pak použita k výpočtu a zobrazení měřítka mapy. K výpočtu slouží matematický vzorec, který z úhlu zeměpisné šířky a poloměru Země vypočte délku rovnoběžky. Tuto hodnotu pak už stačí jen vydělit 360 a získáme délku jednoho stupně.

Metoda paint

public void paint(Graphics) – tato metoda zajišťuje veškerou práci při vykreslování v jednom cyklu. Volá funkce na vykreslení objektů a zadává požadavky na načítání souborů.

Výpočet čísla stránky

Na začátku je vždy pro aktuální vrstvu map, kterou bude potřeba vykreslit spočteno, jaké stránky jsou potřeba (data z jakých souborů). Vždy jsou to čtyři sousední stránky v obdélníku. Každou stránku pak načteme do jednoho ze čtyř oken. (Toto je jen imaginární označení, v programu se pracuje vždy se třemi vektory, které zastupují jedno okno a je v nich uložen jeden soubor).

0	1
2	3

Do nultého okna bude načtena stránka, jejíž číslo vypočteme za pomocí počátku souřadnic (proměnná nula) a ze znalosti toho, jak je daná vrstva dělena. Ostatní čísla stránek jsou odvozena od čísla stránky v nultém okně. Číslo stránky v prvním okně je o jedna vyšší, druhé okno sousedí s nultým zespodu (číslo stránky je tedy posunuto o počet stránek na jednom řádku vrstvy), číslo stránky ve třetím okně je opět o jedna vyšší než ve druhém okně. Je jen třeba si dát pozor na požadavek na číslo stránky mimo rozsah. Potom je počítáno se stránkami které jsou požadavku nejblíže, ale máme je k dispozici.

Obsluha načítání stránek – posun oken

Když víme jaké stránky jsou potřeba, musíme také zjistit, zda je máme v paměti, nebo je bude třeba načíst. To se děje pomocí bitové mapy, která nám řekne, které stránky jsou v paměti již uloženy. Ostatní pak musíme načíst.

V paměti jsou vždy uloženy čtyři stránky, kde každá je uložena v jiném okně (tedy v jiných třech vektorech). Bitová mapa nám však pouze říká, jestli je stránka načtena a už ne to, ve kterém z oken leží. Při pohybu nad mapou je tedy potřeba zajistit správné kopírování vektorů mezi okny a zároveň udržování korektně vyplněné bitové mapy.

Mějme například v okně „0“ načtenou první stránku, v okně „1“ druhou stránku, v okně „2“ dvacátou první stránku a v okně „3“ dvacátou druhou stránku (vrstva by tedy měla na jednom řádku 20 stránek). Navíc mějme korektně vyplněnou bitmapu. Pokud dojde nad mapou k pohybu doprava a nastane situace, kdy je třeba načíst další stránky, budou to jistě stránky č.3 a č. 23, ty by měly spadat do oken „1“ a „3“, tam už ale máme stránky č.2 a č.22 a ty nechceme přepsat, protože je budeme potřebovat. Jednoduše by to tedy šlo vyřešit tak, že bychom je přepsali a načetli do oken „0“ a „2“ a následně přepsali bitovou mapu nově načtenými soubory. To by však znamenalo odstranění dvou stránek a jejich následné načtení. Efektivnější tedy bude, pokud již načtené, ale ještě stále potřebné stránky zkopírujeme směrem doleva (tedy stránky č.2 a č.22 do oken „0“ a „2“) a do pravých oken načteme nové stránky. Bitmapu pak upravíme tak, že z ní odstraníme záznam o stránkách již přepsaných (tedy č.1 a č.21) a přidáme stránky nově načtené (3 a 23).

Obsluha načítání stránek – přepojování vláken

Horší situace však nastane, pokud se uživatel pohybuje nad mapou rychle a stránky se nestíhají načítat. Pokud by byl program řešen pouze pomocí předchozího algoritmu, stalo by se to, že by se načítalo mnoho souborů a některé dokonce i vícekrát. Malým vylepšením by bylo rušení vláken u kterých jsme si jisti, že nebudou nadále potřeba.

Daleko chytřejším řešením je však přepojování vláken. Funguje podobně jako okna, ale využívají se i data která nebyla ještě kompletně načtena. Jednotlivá vlákna

totiž nejdříve načtou celý soubor a pak data předají k překreslení. Pokud by byl pohyb rychlejší než doba na načtení souboru, nebylo by co kopírovat z jednoho okna do druhého a tak by načítání muselo vždy začínat od znova.

Přepojování vláken funguje asi takto: vezměme podobný příklad jako byl uveden výše., jen s tím rozdílem, že uživatel udělal rychle za sebou více posunů mapy a tedy se rychle dostal ze stránky č.1 až na stránku č.3 (tedy místo stránek 1,2,21,22 by měly být načteny stránky 3,4,23,24) . Poté co se uživatel přesunul ze stránky č.1 na stránku č.2 byly dvě stránky překopírovány směrem doleva a další dvě se začaly načítat (3 a 23), ještě než se však stihly načíst, uživatel udělal další krok a dostal se na stránku 3. Nyní by bylo potřeba znovu kopírovat data směrem doleva v oknech, to však není možné, protože data ještě nejsou načtena. Jediné co je nyní možné, je vzít vlákno z okna „1“ a přiřadit ho do proměnné okna „0“ (a obdobně pro „2“a“3“). Ještě předtím je ale třeba zrušit činnost v původních vláknech oken „0“ a „2“, jinak by totiž vlákna dále žila a načítala by soubory, o které už nemáme zájem. Nakonec ještě spustíme načítání nových souborů v oknech „1“ a „3“. Tímto je problém pohybu nad mapou vyřešen.

Pokud měníme měřítko mapy (zoom), je výpočet velice podobný. Občas se s měřítkem mapy musí změnit i vrstva. Potom jsou stránky do všech oken načteny znovu a původní bitová mapa nastavena na nulu.

Vykreslení dat

Na display je na konci každého cyklu smyčky vykreslováno několik různých komponent, které jsou na sobě nezávislé, je jen potřeba je vykreslit ve správném pořadí.

Nejdříve jsou vykresleny polygony, protože by jinak překrývaly ostatní objekty. Následují křivky a body. Přes ně jsou překresleny zeměpisné souřadnice. Nakonec je vykreslena spodní informační lišta. Tam lze sledovat načítání stránek (souborů). Prázdný obdélník znamená, že vlákno načítající soubor ještě běží, plný obdélník znamená, že již doběhlo a data jsou načtena. Pokud některé z vláken ještě běží, jsou na scéně zobrazeny také přesýpací hodiny pomocí metody drawClock(Graphics, int, int). Po doběhnutí všech vláken zmizí.

Dále je na liště informace o zoomování a přesné měřítko, díky němuž lze z mapy číst přesnou vzdálenost. Jelikož máme v programu uloženou informaci o tom kolika stupňům odpovídá jeden obrazový bod a pomocí funkce getDegreeLength(double) umíme spočítat délku jednoho stupně, lze pak už jednoduše dopočítat délku 30 pixelové úsečky v mapě. Hodnota se může pro různou zeměpisnou šířku měnit a proto je přesnější používat větší měřítko mapy. Hodnota je vždy vypočtena pro zeměpisnou šířku uprostřed scény.

Komponenta pro určování přesné pozice

Aby bylo možné díky mapě určit přesnou zeměpisnou pozici, je základní zobrazení mapy rozšířeno o komponentu, jež toto umožňuje. Po zadání volby se zobrazí nad mapou kurzor, jímž můžeme pohybovat. Ve vrchní části scény se zobrazí poloha ve stupních, minutách a desetinách minut. Tato hodnota je vypočítána z polohy levého horního rohu displaye (kterou známe) a přičtení pozice kurzoru nad scénou. K zobrazení je pak použita metoda getStringDegrees(double, boolean). Protože je potřeba určit pozici co možná nejpřesněji, je třeba, aby se kurzor pohyboval po pixelech. V tomto případě by se ale musela při každém pohybu překreslit celá scéna, což by velice zdržovalo určení pozice.

Proto je pro tuto komponentu zvolen algoritmus, který uloží zobrazenou scénu jako obrázek a nadále se překresluje jen kurzor a uložený obrázek, což je několikanásobně rychlejší. Má to však tu nevýhodu, že nelze kurzorem posouvat zobrazenou scénu. K tomu je zapotřebí kurzor vypnout, provést potřebný pohyb (přesun, přiblížení), a znovu kurzor spustit.

Kapitola 6

Ovládání

6.1. Základní ovládací prvky

Ovládání je uzpůsobeno tak, aby bylo co možná nejvíce intuitivní pro uživatele a zároveň aby pracovalo se všemi standardními mobilními telefony. K pohybu nad mapou lze použít jak joystick (pokud je jím telefon vybaven), tak i klávesy určené k pohybu v menu. Aby byla aplikace uzpůsobena i ostatním telefonům je možno pohyb uskutečnit i křížem na numerická klávesnici (‘2‘-nahoru, ‘4‘-vlevo, ‘6‘-vpravo, ‘8‘-dolů). Tyto klávesy jde samozřejmě používat i u telefonu s vestavěným joystickem nebo šipkami. Například pro větší pohodlnost.

Ke změně měřítka mapy jsou použity klávesy ‘*‘ a ‘#‘, kterými je opět vybaven každý mobilní telefon. Klávesa ‘*‘ je použita k zvětšení měřítka a tedy k přiblížení mapy. Klávesa ‘#‘ má přesně opačný význam.

Spustit a vypnout zobrazení kurzoru a souřadnic lze pomocí klávesy ‘5‘ nebo pomocí potvrzovacího kontextového tlačítka. K pohybu kurzoru nad mapou se používá stejných tlačítek jako při posuvu mapy. Mapa však nelze posunout, přiblížit ani oddálit. Aplikaci lze vypnout pomocí kontextového tlačítka menu a výběrem volby “EXIT“ (u některých výrobců není třeba).

Bohužel nemám možnost aplikaciotestovat na přístrojích různých výrobců, takže se může stát, že ovládání kontextovými klávesami nebude plně funkční. Ovládání pomocí numerických kláves by však mělo fungovat na všech telefonech bez ohledu na výrobce.

6.2. Urychlení obsluhy aplikace

Protože se mapa skládá z několika vrstev a každou vrstvu je potřeba znovu načíst, je pro uživatele výhodnější, pokud již v nejvyšší vrstvě vybere polohu, kterou chce zobrazit a následně dle potřeby zvětšuje měřítko. Pokud je ještě třeba korigovat polohu, je opět nejlépe to udělat předtím, než se začne načítat nižší vrstva. Hranice pro jednotlivé vrstvy jsou pro koeficient přiblížení mezi 9-10, 13-14, 18-19.

Pokud se chce uživatel přemístit na mapě o delší vzdálenost, ale nechce přitom měnit měřítko, je lepší několikrát stisknou klávesu ve směru kterým se chce přemístit a poté vyčkat až se načtou data. Pokud by vždy po stisku tlačítka čekal, až se mapy načtou, zdržoval by se načítáním map o které nemá zájem. Stejná věc pak platí i pro změnu měřítka mapy. Pokud se chce uživatel přiblížit z nejvyšší vrstvy do nejnižší, je lepší několikrát zmáčknout tlačítko a kontrolovat koeficient přiblížení. Je to opět rychlejší než při vyčkávání na načtení map jednotlivých vrstev.

Kapitola 7

Porovnání metod

V této kapitole se snažím nastínit vývoj map pro mobilní telefon. Vždy jsou u každého zpracování uvedeny výhody a nevýhody

7.1. Mapa jako obrázek

Protože již první grafické telefony podporovaly zobrazování různých obrázků, hned jak to bylo možné (na počátku rozvoje barevných telefonů), objevily se v mobilním telefonu různé skenované mapy, které sloužily k základní orientaci. Protože byly zdroje těchto telefonů velice omezené, nebylo možné používat příliš velké mapy. Stejně nebylo možno se nad vykresleným obrázkem pohybovat a proto byl uživatel omezen jen na mapy velikosti displaye nebo menší. Při zobrazení jiného listu mapy bylo potřeba přepínat mezi obrázky.

7.2. Mapa ve webovém prohlížeči

Předchozí varianta nebyla příliš použitelná jako plnohodnotná mapa. Proto vznikl nápad využít vestavěného webového prohlížeče jako lepší prvek pro prohlížení rastrových map. Prohlížeč má totiž možnost zobrazovat obrázky vedle sebe i pod sebou. Je možno s celou webovou stránkou pohybovat a tím i měnit pohled na mapu. Pak už jen stačilo vytvořit obrázky reprezentující mapu, správně napsat html stránku a nahrát ji do paměti telefonu. Bylo též možné vytvořit přibližování mapy. A to tak, že každý mapový list byl zároveň hypertextovým odkazem na nižší vrstvu mapy. Tato metoda nepoužívala připojení k internetu, pouze využívala webový prohlížeč k zobrazování map.

7.3. J2ME prohlížeč rastrových map

Prvním opravdovým programem na vizualizaci map v telefonu byl prohlížeč vytvořený v jazyce J2ME. Prohlížeč vytvořený především k zobrazování map měl samozřejmě již mnohem širší využití. V jedné mapě nebyl problém měnit měřítko, nahrávat do jedné aplikace více různých map a přepínat mezi nimi. Uživatel laik byl schopen si vytvořit v grafickém editoru mapu a následně ji importovat do aplikace, čímž se aplikace stala ještě mnohem užitečnější. Díky tomu také vzniklo mnoho variant jedné aplikace, kdy bylo možné nahrát do paměti telefonu jen tu mapu, kterou uživatel chtěl.

Nevýhody ale stále zůstaly. Rastrová data poskytují vzhledově krásné mapy, jsou přehledné a není problém je získat. Problémem je však jejich velikost. Ať už se jedná o formát GIF či JPEG*, vždy je pro rozsáhlejší mapy objem dat rastrových obrázků příliš velký. V poslední době, kdy mají mobilní telefony výměnné karty a tím pádem téměř neomezenou kapacitu však již tento problém upadá.

Limitující je však pro tento druh map jakákoliv indexace. Uživatel dnešní výpočetní techniky a internetu si zvykl na vyhledávání míst v mapě a i v mobilním telefonu by chtěl podobnou službu. Rastrová data neposkytují jakoukoliv informaci o tom co obsahují (ve smyslu geografie), proto jedinou možností je dodat ručně vytvořená data a s těmi spojit rastrovou mapu. I takovéto aplikace existují, ale vzhledem k náročnosti zpracování jednotlivých bodů a jejich omezenému využití obsahují jen základní informace.

*JPEG není formát souboru, nýbrž ztrátová komprese obrazu. Bohužel se ale toto označení již natolik zažilo, že jej zde tak označuji. Původní název je JFIF (JPEG File Interchange Format).

7.3. Online mapy

Online mapa je tvořena dvěma částmi – aplikací na straně uživatele a databází na straně serveru. Komunikace probíhá tak, že uživatel pošle serveru požadavek, server ho vyhodnotí a vrátí odpověď.

Nejdříve uživatel zadá město, které by chtěl vyhledat, odešle požadavek a server zašle zpět mapu s okolím zvoleného města (či jiného objektu). Poté se uživatel může pohybovat nad mapou a vždy když opustí oblast mapy, kterou má načtenou, odešle se opět serveru požadavek na další mapu. Takto funguje i změna měřítka mapy.

Tento druh map řeší oba problémy předchozích typů. Požadavky jsou vyřizovány a zpracovávány na straně serveru a data jsou většinou čerpána z volně přístupných zdrojů. Ty jsou dnes na velmi vysoké úrovni, protože mají komerční zázemí. Přenos dat nakonec probíhá opět v podobě rastrových map.

Velkou nevýhodou je však u těchto map vysoký přenos dat, který velice snižuje jejich využití. Málokterý uživatel totiž chce platit za vyhledávání v mapě (ač neplatí vlastníkovi map, ale operátorovi). Cena se totiž může vyšplhat do desítek či stovek korun. Výjimku tvoří lidé, kteří využívají neomezený internet. Pro takovouto skupinu je toto řešení zajisté nejlepší.

7.4. Vektorové mapy

V této oblasti dochází v posledních letech k velkému rozmachu. Je to především díky rozšíření GPS přijímačů mezi amatéry, kteří si nechtějí nebo nemohou pořídit drahé vektorové mapy a proto se snaží je vytvářet sami. To by nešlo bez spolupráce a tak se obvykle spojí více různých zdrojů, které dají dohromady komplexnější mapu. To je také případ map využitých v tomto projektu.

Takovéto mapy je potom možné využívat v různých odvětvích a pro různé účely. Protože na rozdíl od rastrových map v sobě mají uloženou veškerou informační hodnotu, lze jednoduše za pomocí těchto dat vyhledávat objekty. Ze stejného důvodu mohou být vektorové mapy dosahující stejné přesnosti jako rastrová data daleko

úsporněji uložena. Nevýhodou je naopak to, že volně přístupná vektorová data jsou jen v omezené míře. I když v poslední době se i to značně změnilo.

Poslední nevýhodou je daleko složitější zpracování vektorových dat nežli rastrových obrázků, proto se také objevuje daleko méně programů s nimi pracujících.

7.5. GPS navigace

Mobilní telefon lze dnes rozšířit o GPS modul, jenž nám dovolí využívat GPS navigace pomocí displaye. K tomuto účelu slouží především právě výše zmiňované vektorové mapy, se kterými většina GPS přijímačů (ať už samostatných nebo k mobilnímu telefonu) spolupracuje. GPS modul propojený s mapou v mobilním telefonu pak posouvá zpracování map v přístroji o úroveň výš.

Kapitola 8

Možná vylepšení

8.1. Binární data

Nynější aplikace obsahuje jako zdroj dat textové soubory obsahující části mapy. Tato reprezentace je výhodná především pro uživatele. Ten je schopen upravovat si sám části mapy a přidávat například body zájmu.

Co se velikosti týče, jsou data uložená v textové podobě jen o málo větší nežli by byla binární data. J2ME používá totiž při kompilaci kompresi ZIP, která je na tento druh dat velice kvalitní. Znatelný rozdíl by v případě binárních souborů však byl při načítání dat. Tato rychlost je omezena především přístupem do paměti, ten by se několikanásobně snížil. Se zavedením binárních souborů by ale aplikace přišla o možnost úpravy vnitřních dat, nebo by bylo potřeba řešit dalším softwarem na konverzi.

8.2. Vyhledávání objektů

Protože vektorová data umožňují implementaci vyhledávání objektů, lze toto rozšíření aplikovat i na tento program. Bylo by však potřeba vytvořit rozhraní pro vyhledávání a slovník s indexy na soubory. Pro vyhledávání ulic by bylo ještě třeba ošetřit vkládání města. Pro duplicitní výsledky by bylo nutné doplnit název okresu města (jiného objektu).

8.3. Vyhledávání pomocí pozice

Pro vyhledávání pomocí pozice by bylo třeba implementovat jen rozhraní, jež by umožnilo zadání souřadnic. Samotné provedení je již díky způsobu fungování aplikace implementováno.

8.4. Vkládání bodů zájmu, ukládání tras

Pro případ, že by chtěl uživatel vložit bod, který v mapě ještě není, by bylo možné vytvořit rozhraní, jež by toto dovolovalo. Kurzorem by uživatel zvolil pozici a následně by ve formuláři zvolil typ a název. Tyto body by se však musely uchovávat ve vnitřní databázi, neboť J2ME nemá možnost měnit externí soubory.

Obdobnou funkcí by bylo zaznamenávání (nebo také vytyčování) trasy. Uživatel by zadal kurzorem trasu, kterou prošel (projde) a trasa by se opět uložila do telefonu. Pak by ji bylo v případě potřeby možno zobrazit na mapě.

8.5. Filtrace dat

Tato funkce by umožnila uživateli zobrazit jen data, která potřebuje, čímž by zpřehlednila mapu. Např. pokud by chtěl uživatel vyhledat nemocnici, dal by zobrazit jen nemocnice a na mapě by se zbytečně nezobrazovaly i kostely, obchodní domy a další. Stejnou funkci by šlo použít i pro křivky a plochy. U ploch by to mělo ještě jeden zásadní vliv na aplikaci. Protože překreslování ploch je celkem pomalé, a pohyb nad mapou zdržuje, vše by se urychlilo.

8.6. Uložení poslední pozice

Pokud někdo využívá mapu stále pro stejný účel a na stejném území, měl by být program též vybaven volbou pro uložení poslední zobrazené mapy a při opětovném spuštění zobrazit shodnou pozici.

8.7. Nápověda

V případě, kdy by uživatel nevěděl jak přesně má aplikaci ovládat, mohl by si z menu vyvolat nápovědu. Další nápověda by sloužila ke stručnému popisu ikon, křivek a ploch.

8.8. Inverzní mapa

Protože mapu lze využívat také za snížených světelných podmínek, bylo by možno aplikaci přepnout do inverzního (nočního) režimu. V tomto režimu display nevyzařuje takové množství světla a proto by jím uživatele neoslňoval. Stejnou funkci najdeme např. u GPS navigací.

8.9. Načítání libovolných map

Aby se stala aplikace mnohem více univerzální, bylo by potřeba ji rozšířit tak, aby byla schopna načítat různé mapy alespoň v jednom standardním formátu. To je však velice limitující, protože data je pro aplikaci potřeba nejdříve předpřipravit procesy popsanými v Kapitole 3. Za určitých podmínek však lze vytvořit program, který data připraví a do aplikace se už jen zkopírují.

8.10. GPS navigace

Propojením mobilního telefonu s GPS modulem by bylo možné vytvořit soustavu shodnou s GPS přístroji. Program by sám určil přesnou pozici uživatele a zobrazil ji na mapě. Bylo by snazší vytvářet prošlé trasy a ukládat je. Vzhledem k tomu, že GPS modul poskytuje i některé rozšiřující informace jako je rychlost pohybu, šlo by aplikaci dovybavit i dalšími funkcemi, jako například průměrná rychlost, graf pohybu, měřič vzdálenosti…

Další vylepšení se už týkají spíše zpřesnění používaných map a jejich doplnění.

Kapitola 9

Závěr

Cílem této bakalářské práce bylo vytvořit uživatelsky přívětivý prohlížeč vektorové mapy České republiky. Osobně si myslím, že se toto podařilo. Mapu testovalo několik uživatelů a všichni se shodli na tom, že je tato aplikace velice užitečná a bude jim nápomocná. Díky volbě map a způsobu zpracování bude možno tuto mapu používat zdarma čímž se její použitelnost ještě zvýší.

Doufám, že tento projekt nezanikne a bude pokračovat i po zakončení studia. Předpokládám doplnění aplikace o většinu výše uvedených rozšíření. Již jsem začal jednat o rozšíření o GPS navigaci. To ale už nejspíše nebude samostatná práce jako doposud, ale sloučí se tento projekt s projektem, který vytvořil aplikaci spolupracující s modulem GPS. Jednou by tak mohlo vzniknout dílo plně konkurenceschopné přijímačům GPS.

Díky této bakalářské práci, jsem vytvořil aplikaci, jež doposud v oblasti mobilních telefonů chyběla. Doufám, že bude tato aplikace přínosem nejen pro obyčejného uživatele, ale že se díky ní začnou vektorové mapy rozšiřovat a počet aplikací využívajících vektorová data se zvýší.

Literatura

[1] Weinman Lynda (2004): Velká kniha webdesignu : Zoner Press, Brno

[2] Belady L. A., Nelson R. A.,Shedler G. S.(1969): Communications of the ACM ACM Press, New York, USA

[3] Popis standardů J2ME: http://java.sun.com/javame/

Přílohy

Obsah přiloženého CD

CzechMap.jad	aplikace
CzechMap.jar	aplikace
Main.java	zdrojový soubor
Mapa.java	zdrojový soubor
Vlakno.java	zdrojový soubor
jmicropolygon_midp2-1.1.2.jar	použitá knihovna
Bakalářská_práce_David_Stach.pdf	bakalářská práce