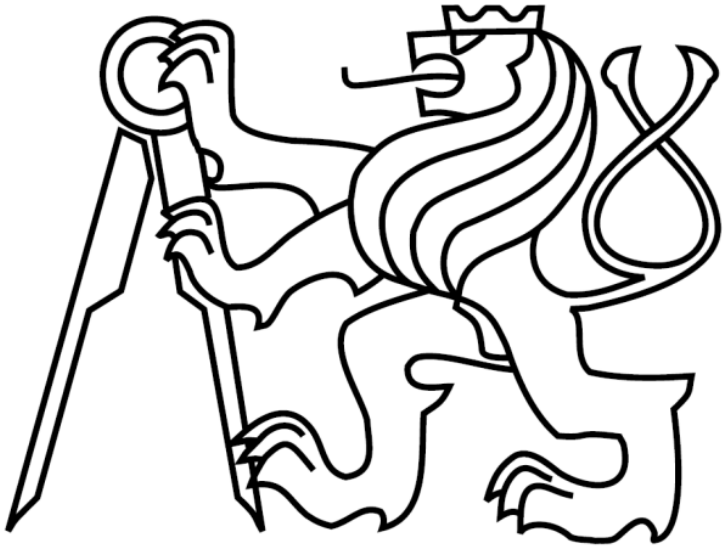
České vysoké učení technické v Praze
Fakulta elektrotechnická



Bakalářská práce

Java implementace mapy pro prostředí mobilních telefonů

Jiří Vaněk

Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

Studijní program: Elektrotechnika a informatika strukturovaný bakalářský

Obor: Informatika a výpočetní technika

Leden 2010

Poděkování

Rád bych poděkoval mému vedoucímu Ing. Pavlovi Kubalíkovi, Ph.D. za jeho cenné rady, náměty a zejména čas, který si na mne a mé dotazy při tvorbě této práce udělal. Dále bych poděkoval své rodině a přátelům za toleranci a podporu v průběhu vývoje bakalářské práce.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů. (autorský zákon).

V Praze dne 1. 1. 2010 …….

Abstrakt

Cílem práce bylo vytvořit aplikaci pro mobilní telefon, která bude schopna zobrazovat mapu formou statických obrázků. V této mapě má být možnost procházení, hledání měst a přibližování v několika úrovních.

Výsledkem je aplikace, ve které je možné danou mapu takto použít a nahrávat si mapy další podle požadavků uživatele.

Abstract

This work aims to develop a mobile application capable of displaying a map using static images. The map should enable user to view it, to search for a city or to zoom in multiple levels.

The result is an application which contains all mentioned features and is capable of loading extra maps according to user's requirements.

Obsah

1 Úvod ... 11
1.1 Důvody pro vytvoření této práce ... 11
1.2 Existující řešení ... 11
1.2.1 TrekBuddy ... 11
1.2.2 GoogleMaps ... 11
1.2.3 GMaps ... 11
1.2.4 Mapy ... 11
1.3 Specifikace cíle ... 12
2 Teorie ... 13
2.1 Mapa ... 13
2.1.1 Základní rozdělení digitálních map ... 13
2.2 Platforma JavaME ... 14
2.2.1 Konfigurace ... 15
2.2.2 Profily ... 16
3 Analýza a návrh implementace ... 18
3.1 Platforma Java ME ... 18
3.1.1 Hardwarové požadavky MIDP ... 18
3.1.2 Aplikace MIDP ... 19
3.2 Platforma Symbian ... 20
3.3 Platforma Android ... 20
3.4 Úzká místa návrhu ... 22
3.5 Výběr způsobu implementace ... 22
3.6 Návrh řešení ... 22
4 Realizace ... 23
4.1 Balík main ... 23
4.2 Třída MenuList ... 24
4.3 Třída Map ... 25
4.4 Třída World ... 26
4.5 Třída ExternMap ... 27
4.5.1 Struktura XML souborů ... 27
4.5.2 Zobrazení obrázků na displeji mobilního telefonu ... 29
4.5.3 Parsování XML souboru ... 31
5 Testování ... 33
5.1 Funkční testování ... 33
5.2 Testování použitelnosti ... 34
5.3 Uživatelské prostředí ... 35
5.4 Porovnání s konkurencí ... 37
6 Závěr ... 38
7 Seznam literatury ... 39
8 Obsah přiloženého CD ... 40

1 Úvod

1.1 Důvody pro vytvoření této práce

V dnešní době je velkým trendem používat GPS navigace nebo mapy v mobilních telefonech. Nevýhodou u mobilního telefonu je, že většina aplikací zobrazující mapu potřebuje pro svůj běh operační systém. Pokud takové mobilní zařízení s operačním systémem vlastníme, můžeme si vybrat z široké nabídky aplikací. Problém nastane, pokud používáme tzv.“hloupý telefon“. Takové zařízení operační systém nemá, a pokud má v sobě implementovanou podporu Javy musíme si vystačit jen s javovými aplikacemi. Na tyto mobilní telefony už velký výběr aplikací nenajdeme. Pokud se nám povede takovou nalézt, většinou pro svůj běh potřebuje připojení k internetu, aby mohla získávat potřebná data a mapy zobrazovat. Tento nedostatek byl motivem pro vytvoření MIDP aplikace zobrazující mapy offline s možností nahrávání svých vlastních map na paměťové úložiště přímo do mobilního zařízení a stahování dat z internetu se tak vyhnout.

1.2 Existující řešení

1.2.1 TrekBuddy

Aplikace, která pomocí připojeného GPS modulu dovede uživatele na zadané zeměpisné souřadnice. Zobrazí rychlost a kurz. Umožňuje odeslat SMS s aktuální polohou a hlavně, dovoluje vytvářet své vlastní mapy a ukládat je na paměťové úložiště přímo v mobilním telefonu.

1.2.2 GoogleMaps

Vytvořila společnost Google. Aplikace dovoluje zobrazovat mapy v různých režimech, vyhledávat ulice, firmy a plánovat trasy. Toto vše je možné, pokud je k dispozici internetové připojení.

1.2.3 GMaps

Aplikace, která zobrazuje mapy od Google, portálu Yahoo a virtuální mapy od MSN na obrazovce mobilního telefonu. V programu je možné přepnutí několika módů od klasického až po zobrazení satelitních snímků. Nevýhodou je nutnost datového tarifu pro získávání dat.

1.2.4 Mapy

Vývojářská společnost iCom Vision zvolila tento jednoduchý název pro aplikaci, která si dala za cíl při zobrazení map nepřenášet velké množství dat. Umožňuje nalézt požadovanou ulici a tu uložit pro pozdější použití. S dovolením uživatele je možné se dále posouvat v mapě a zobrazovat vedlejší segmenty. Jak bylo nepřímo zmíněno i u této aplikace je potřeba přenos dat z internetu.

1.3 Specifikace cíle

Cílem této práce je vytvořit aplikaci pomocí programovacího jazyku JAVA na platformě Java ME. Vstupními daty budou logicky rozdělené soubory XML a statické obrázky. Aplikace bude schopna tato data zobrazit a vytvořit dojem jedné velké mapy, ve které je možno se posouvat a v závislosti na uložených datech vyhledávat, přibližovat či oddalovat. Analýza existujících řešení bude nápovědou pro mou realizaci. Cílem je poskytnout jednoduché ovládání aplikace a možnost nahrávat vlastní mapy dle požadavků uživatele.

2 Teorie

2.1 Mapa

Zmenšené, zevšeobecněné zobrazení povrchu Země, ostatních nebeských těles nebo nebeské sféry, sestrojené podle matematického zákona na rovině a vyjadřující pomocí smluvených znaků rozmístění a vlastnosti objektů vázaných na jmenované povrchy[1]. Jedna z podmnožin map je mapa digitální, latentní, kterou je možno zviditelnit pomocí nějakého zařízení. Zařízením je myšleno PC[2], PDA[3] , GPS[4] a podobné.

2.1.1 Základní rozdělení digitálních map

Digitální mapy můžeme dělit podle několika hledisek – podle míry ovlivnění zobrazení uživatelem, podle typu zdrojových dat a také podle dynamických vlastností mapy.

Obrázek 2-1 Rozdělení digitálních map

2.1.1.1 Ovlivnění map uživatelem

Pro ovlivnění zobrazení mapy uživatelem jsou mapy interaktivní, kde může uživatel pomocí ovládacích prvků mapou pohybovat, zvětšovat, zmenšovat a tak podobně. Tuto interaktivitu zajišťují různé techniky a jsou vytvářeny pomocí různých programovacích jazyků, jako jsou Java, PHP[5], ASP[6], JSP[7]. Opakem interaktivních jsou mapy určené pouze pro čtení – view only.

---

[1] Definice podle Mezinárodní kartografické asociace (ICA)
[2] Personal Computer
[3] Personal Digital Assistant
[4] Global Positioning System
[5] Hypertext preprocessor
[6] Active Server Pages
[7] JavaServer Pages

2.1.1.2 Typ zdrojových dat

První skupinou, z níž mapa je generovaná, je původně analogová, která je převedená digitalizací nebo skenováním do digitální podoby. Kvalita takovéto mapy závisí na stavu originálu a také na parametrech digitalizace jako je použitý formát, barevná hloubka a použité rozlišení. Tímto druhem zpracování projdou zvláště staré mapy. Druhou skupinou jsou mapy generované přímo z dat digitálních.

2.1.1.3 Způsob vizualizace

Vizualizace může být pomocí rastrů nebo vektorů. Předností vektorového způsobu je dobrá interaktivita, protože vektorové mapy jsou složeny z objektů linií, křivek, polygonů a dalších prvků. Rastrové obrázky jsou tvořeny množinou (maticí) bodů, kde pro každý bod je udána barevná hloubka, přičemž jednotlivé formáty se liší používaným kompresním algoritmem. Základním rozdělením podle kompresního formátu jsou bezeztrátové algoritmy – formáty BMP[8], GIF[9], PNG[10]. Představitelem ztrátového formátu je JPEG[11]

2.2 Platforma JavaME

Na začátku devadesátých let společnost Sun Microsystems vytvořila programovací jazyk nazvaný Oak. Společnost vycházela z jazyka C++ a u nového jazyka odstranila některé rysy C++ jako například ukazatele a správu paměti pro zmenšení programátorských chyb. Po tomto zjednodušeném jazyku nebyla potřebná poptávka a tak společnost využila stále se zvyšující poptávky po prohlížečích webové sítě a svůj jazyk Oak přejmenovala na Java a použila ho pro vývoj přenositelného prohlížeče HotJava. Dnes známe především javové aplety.

Přenositelnost mezi platformami vytvořila zájem komerčních koncových uživatelů, a aby společnost Sun splnila potřeby vývojářů, rozšířila rozsah javové platformy. Tímto rozšířením jsou knihovny propracovanějšího uživatelského rozhraní a dalších funkcí pro výpočty a zdokonalené zabezpečení. J2ME je platformou pro malé přístroje a poskytuje funkcionalitu JDK[12] 1.1. Na rozdíl od stolních počítačů, kde dominuje J2SE[13] , a serverových řešení – J2EE[14], je „pocket“ svět mnohem rozmanitější. Každé zařízení z této oblasti má jiné schopnosti a vytvořit jediný produkt, který by zaštiťoval tuto rozsáhlost, je nemožné. J2ME proto není jedinou entitou, ale souborem (množinou) specifikací, jež definuje určitou podmnožinu a každá má za úkol řešit specifickou část. Dělení programovacího jazyka podle typu zařízení definují tzv. profily, které rozšiřují základní schopnosti konfigurace.

[8] Microsoft Windows Bitmap
[9] Graphics Interchange Format
[10] Portable Network Graphics
[11] Joint Photographic Experts Group
[12] Java Development Kit
[13] Java2 Standart Edition
[14] Java2 Enterprise Edition

2.2.1 Konfigurace

Specifikace definující softwarové prostředí pro určité zařízení, které má určené vlastnosti. Jedná se např. o velikost dostupné paměti, frekvenci procesoru či typ síťového připojení. Pro správnou funkčnost programového prostředí je povinností výrobců hardwaru danou specifikaci plně implementovat. J2ME v současnosti definuje dvě základní konfigurace. Konfiguraci CLDC[15] a CDC[16]. Každá se skládá z virtuálního stroje Javy a standardní kolekce javových tříd, které poskytují programovací prostředí pro aplikační software.

2.2.1.1 CLDC

Tato konfigurace je určena pro nízkoúrovňovou oblast spotřební elektroniky. V této oblasti jsou mobilní telefony či PDA zařízení s přibližně 512kB volnou pamětí. Z tohoto důvodu je CLDC spojena s tzv. bezdrátovou Javou (Wireless Java), jejímž cílem je umožnit uživatelům mobilních telefonů nákup a stažení javových aplikací na svá zařízení. Výchozí implementace CLDC obsahuje zdrojový kód pro platformy Windows, Linux a Solaris. Obsahuje také virtuální stroj KVM[17], jde o VM s omezenými funkcemi, který má malou spotřebu paměti. Existují však i jiné VM, které lze místo něj použít.

2.2.1.2 CDC

CDC se zaměřuje na ty zařízení, které nejsou ani počítači ani „hloupými“ mobilními telefony. Jedná se o zařízení, která mají více paměti a obvykle i výkonnější procesory a mohou podporovat více ze softwarového prostředí Javy. Příkladem jsou výkonnější PDA zařízení, chytré telefony (tzv. smartphone), webové telefony, Set-Top-Boxy apod.

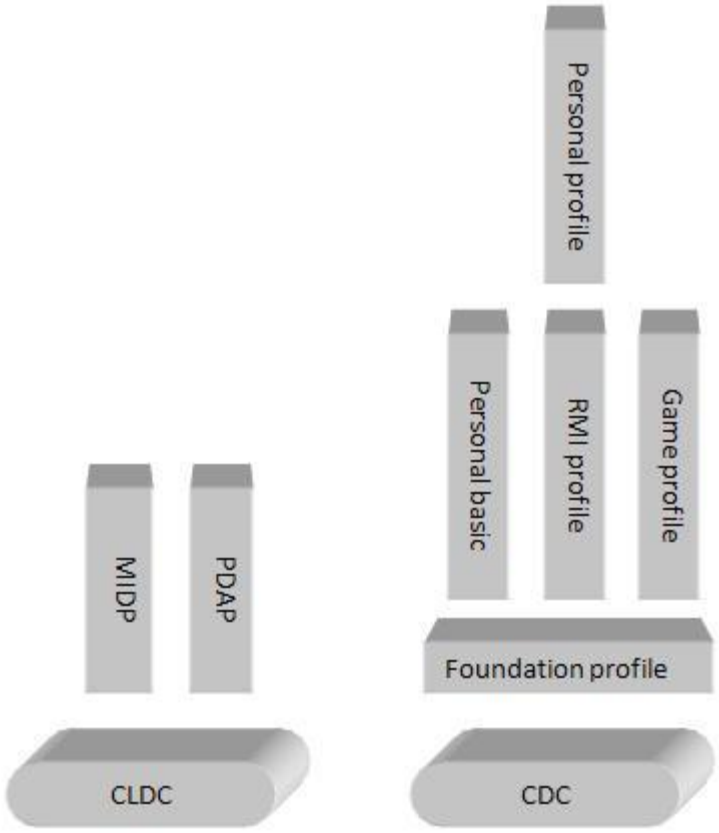
Konfigurace obsahuje standardní sadu tříd jazyka Java. Třídy musí být založeny na platformě Java 2. Tímto opatřením je zajištěna maximální podpora slučitelnosti mezi aplikacemi psanými pro různé platformy J2ME i aplikacemi psanými v J2SE. Obecně řečeno se můžeme spolehnout, že J2ME využívá třídy a balíčky J2SE, kdekoli je to možné. To znamená, že J2ME si nemůže vytvořit vlastní třídu, která řeší stejný problém jako třída v J2SE. Samozřejmě existují výjimky, například CLDC využívá vlastní balíček, který obsahuje odlehčenou sadu tříd pro síťové služby namísto balíčku java.net.

---

[15] Connected Limited Device Configuration
[16] Connected Device Configuration
[17] Kilobyte Virtual Machine (K se dnes už jako kilobyte nepřekládá, zkratka nic neznamená)

2.2.2 Profily

Profil doplňuje konfiguraci přidáním dalších tříd, které poskytují vhodné funkce pro určitý druh zařízení. Obě konfigurace J2ME mají jeden či více profilů a některé z nich mohou spoléhat na jiné profily. Obrázek 2-2 ukazuje závislost profilu na dané konfiguraci.



Obrázek 2-2 Konfigurace a profily

MIDP (Mobile Information Device Profile)

Tento profil přidává do CLDC síťové služby, správu trvalých dat a třídy pro tvorbu uživatelského prostředí. Profil je zaměřen převážně na menší displej a úložné prostředky mobilního telefonu. Aplikacím této kategorie se říká midlet podle základní třídy profilu javax.microedition.midlet.

PDAP

Profil určený pro organizéry PDA, které mají lepší displeje a více paměti než mobilní telefony.

Základový profil (Foundation profile)

Přidává většinu základních tříd, které CDC chybí oproti standardní edici. Neobsahuje žádné uživatelské rozhraní.

Personal Basic (Osobní základ) a Osobní (Personal)

Tyto profily přidávají k základu uživatelské rozhraní, jehož záměrem je použití na zařízení, jež mají jednoduché schopnosti. Pro schopnost zpracování složitějšího uživatelského rozhraní je použito profilu Personal.

RMI profil
Přidá k základovému profilu knihovny pro vzdálené vyvolání metod J2SE.

Herní profil (Game profile)
Poskytuje platformu pro psaní herního softwaru na zařízeních CDC.

3 Analýza a návrh implementace

V dnešní době se na trhu objevují nové technologie a také platformy, na kterých aplikace může být spuštěna. Při implementaci mobilní aplikace se musíme nejdříve rozmyslet, jakou platformu vybrat. Ve druhé kapitole jsem se zaměřil na Javu, platformu Java ME jsem si vybral pro vytvoření své aplikace. Zde bych chtěl zmínit požadavky Javy a srovnání s dalšími existujícími platformami.

3.1 Platforma Java ME

Mé řešení aplikace je postaveno na konfiguraci CLDC pro mobilní telefony využívající profil MIDP. Jak už bylo řečeno, MIDP profil je určen pro malá zařízení s omezenými prostředky paměti a displeje.

3.1.1 Hardwarové požadavky MIDP

Profil MIDP definuje řadu tříd, které nejsou běžnou součástí CLDC konfigurace, proto vyžaduje více paměti než základ, a to minimálně 128 kB paměti RAM pro uložení vlastní implementace a kromě toho je potřeba 32 kB pro javovou haldu. Při vypnutí telefonu je vhodné zachovat některá data, např. nastavení v aplikaci. V MIDP je tato hodnota alespoň 8 kB avšak není zajištěno, že data uložená v trvalém úložišti budou k dispozici i při vyjmutí baterie.

Charakteristikou MIDP jsou malé displeje, specifikace vyžaduje, aby obrazovka byla 96 pixelů široká, 54 pixelů vysoká a pixel měl (přibližně) tvar čtverce. Počet barev obrazovky musí být alespoň dvě – to splňuje většina dnešních mobilních telefonů, kde se tato hodnota pohybuje v tisících barev.

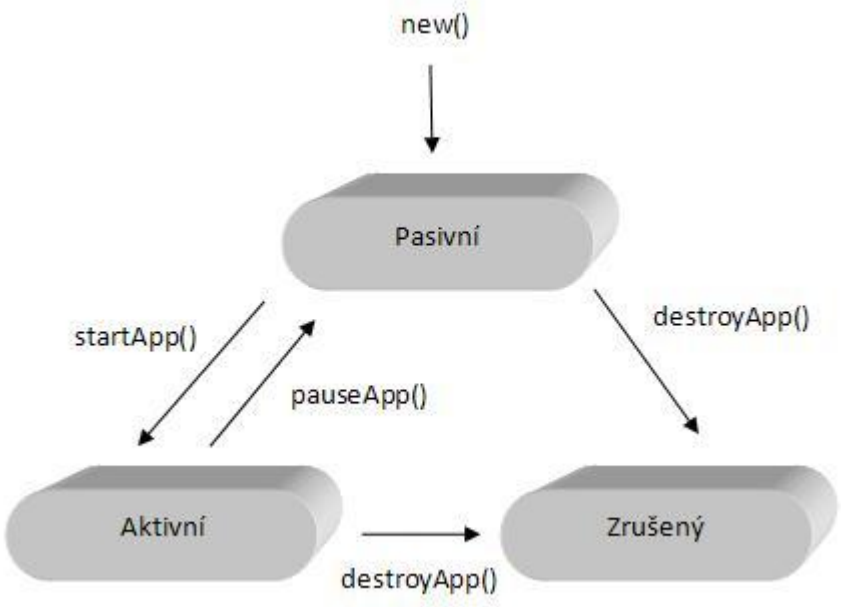
Podobně jako u displejů, tak i vstupních zařízení, existuje více typů. Od plně alfanumerické, přes dotykovou až po běžnou klávesnici mobilního telefonu. Zmiňovaná specifikace předpokládá, že zařízení obsahuje minimálně klávesnici s čísly 0 – 9, popřípadě ekvivalenty kláves.

MIDP nepředpokládá trvalé připojení do sítě nebo přímou podporu TCP/IP protokolu, ale je vyžadováno, aby výrobce zařízení poskytl alespoň iluzi, že zařízení podporuje HTTP 1.1 ať už přímo přes zásobník internetového protokolu nebo pomocí brány WAP.

3.1.2 Aplikace MIDP

Aplikaci opírající se o základ profilu MIDP se říká midlet. Midlet obsahuje alespoň jednu třídu, která musí být zděděná od abstraktní třídy javax.microedition.midlet. MIDlet. Midlet běží v běhovém prostředí v rámci javového VM, který poskytuje tzv. životní cyklus řízený metodami třídy MIDlet.

Při spuštění aplikace je třída MIDlet vytvořena veřejným konstruktorem bez parametrů. Následující obrázek 3-1 zobrazuje stavy, ve kterých se může midlet nacházet. Běh aplikace a přechod mezi stavy řídí aplikační manažer.



Obrázek 3-1 Životní cyklus midletu

Po zavolání konstruktoru se aplikace nachází v pasivním stavu. V tomto stavu by neměly být použity žádné sdílené zdroje. Z pasivního stavu se do aktivního dostaneme zavoláním metody startApp(). V této metodě se inicializují zdroje typu Thread, Connection apod. Metoda pauseApp() způsobí přechod zpět do pasivního režimu a inicializované zdroje by měly být uvolněny. Při ukončení aplikace aplikační manažer zavolá metodu destroyApp(). Pokud je parametr true aplikace bude ukončena v každém případě, ale v případě false se může stát, že se vyhodí výjimka MIDletStateChangeException a tím dá aplikace najevo, že ještě nemá dojít k ukončení.

Před používáním midletu v mobilním telefonu je nutné vhodné zabalení. Zabalení je do jediného souboru JAR[18], který obsahuje nezbytné třídy, obrázky a další soubory pro správný běh aplikace. Balící informace, které zařízení řeknou, co vše je v souboru JAR

[18] Java Archive

musí být uloženy v souboru MANIFEST.MF. Podobné balící informace jsou uloženy v jiném souboru JAD[19], který je uložen odděleně od souboru JAR.

3.2 Platforma Symbian

Firmy Nokia, Motorola, Ericsson a Psion se v roce 1998 spojily a vytvořily společnost Symbian Ltd. Cílem tohoto spojení bylo vytvořit operační systém na základě EPOCu pro mobilní zařízení. Symbian OS podporuje preemptivní[20] multitasking a ochranu paměti. Díky jazyku C++ je dosaženo veliké optimalizace aplikací, ale zato jsou kladeny větší nároky na programátora především, co se týče správy paměti a problematických oblastí charakteristickým pro tento jazyk. Dalším neduhem je, že systém dokáže „běžet" jen na procesorech ARM[21]. Do systému může uživatel přidávat aplikace, které závisí na verzi systému. V současné době vlastní největší podíl firma Nokia, která se rozhodla postupné zveřejnění systému jako open-source pro zvýšení počtu programátorů tohoto systému. Je nutno říci, že Symbian má největší zastoupení na trhu s chytrými telefony – smartphony.

3.3 Platforma Android

Relativně nová softwarová platforma založená na Linuxu, která je přednostně určená pro mobilní zařízení (smartphone, PDA, navigace). Tuto platformu vyvinula společnost Google, která celou platformu i se zdrojovými kódy předala sdružení Open Handset Alliance[22], jíž je také členem. Pro tento softwarový balík je dostupný zatím jen zkušební SDK[23], který poskytuje nástroje a API[24] potřebný pro vývoj základních aplikací pro tuto platformu. Sada knihoven je odvozena z C/C++. Každá aplikace běží jako samostatný proces s vlastní instancí virtuálního stroje Dalvik. Virtuální stroj je optimalizovaný tak, aby na zařízení mohlo běžet těchto strojů více a přitom efektivně. Na rozdíl od jiných virtuálních strojů je Dalvik registrově orientovaná architektura. Dalvik je často označován jako Java Virtual Machine, ale to není úplně přesné, protože bytecode není Java bytecode. Namísto toho, nástroj s názvem dx, který je součástí Android SDK, proměňuje soubory Javy, které jsou zkompilovány kompilátorem Javy do jiného formátu souboru třídy (Dex format). Dalvik executable soubory, které jsou zazipované do jednoho souboru APK – android package na zařízení. Na obrázku 3-2 níže je možné si udělat představu o architektuře Android.

---

[19] Java application descriptor

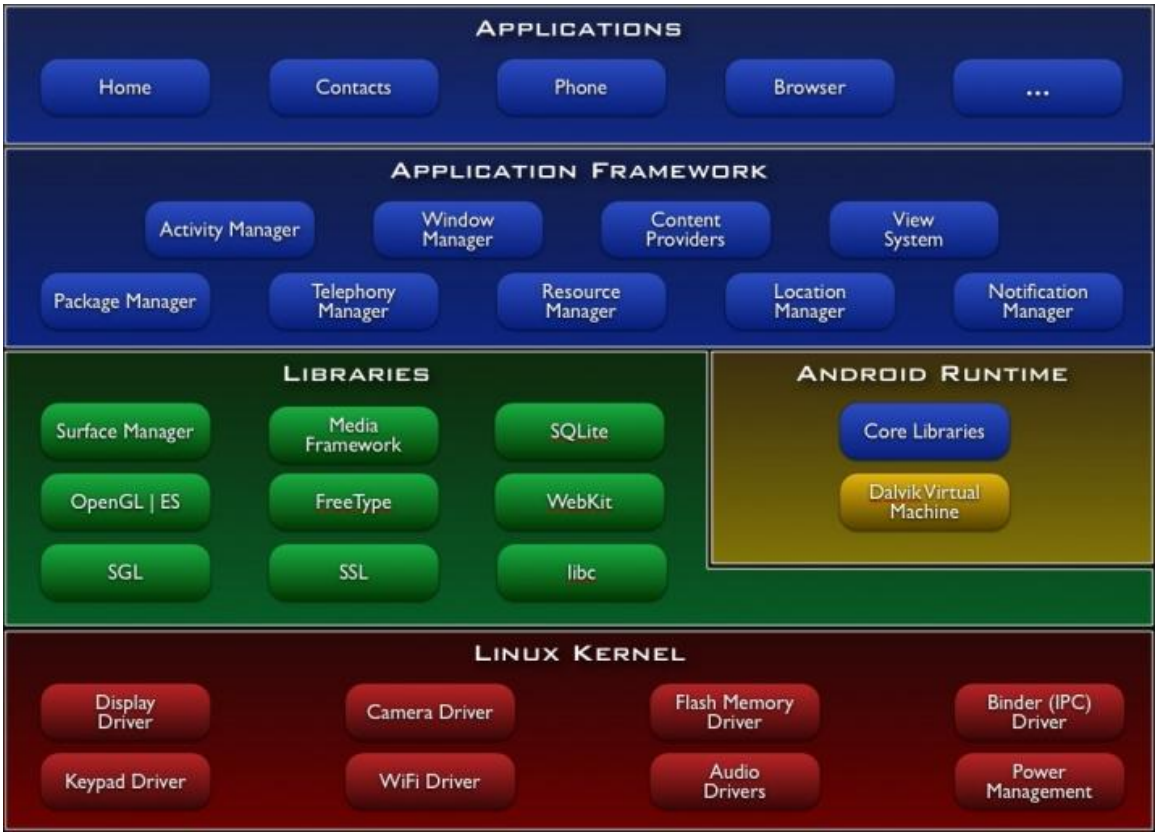
[20] U tohoto druhu multitaskingu je přidělování a odebírání procesoru plně pod kontrolou OS. Ten v pravidelných intervalech přeruší běžící program a předá jinému nebo vrátí zpět tomu samému procesu. Nevýhodou je větší hardwarová náročnost.

[21] Architektura typu RISC vyvinutá firmou AMR limited.

[22] Je uskupení výrobců mobilních telefonů, telekomunikačních operátorů a technologických firem

[23] Software development kit

[24] Application Programming Interface



Obrázek 3-2 architektura Android

Tímto přehledem platforem můžeme shrnout schopnosti všech zmiňovaných platforem. Prvním aspektem srovnání je GUI. Symbian, Java ME i Android jsou schopni zobrazit 2D i 3D grafiku a je k dispozici GUI editor pro snadnější práci vytvoření uživatelského rozhraní. Dále můžeme srovnat přístup k datům přístroje. Symbian a Android mají k těmto datům plný přístup, u Javy ME záleží na dostupnosti rozšiřujících profilů JSR. Funkcionalita je u systému Symbian bez omezeni, Java ME je odkázána na rozšiřující profily JSR avšak je omezen přístup k souborům. Android má funkcionalitu částečnou skrze API. Dobrým měřítkem platforem je i rychlost jakou běží aplikace. Symbian se pyšní nejlepšími výsledky, Java ME a Android je průměrně rychlý vzhledem k běhu na VM. Posledním aspektem srovnání je přenositelnost. Symbian potřebuje provést kompilaci pro určité zařízení, Android je přenositelný v rámci platformy a Java ME je plně přenositelná a závislá jen na VM.

3.4 Úzká místa návrhu

Vzhledem k velkému množství souborů (obrázků) se kterým aplikace pracuje, bylo nutné je oddělit od aplikace do vlastní složky, která se může uložit do paměti telefonu nebo paměťové karty. To sebou nese dva problémy. Prvním je nutnost větší kapacity datového úložiště, druhým je podpora přístupu k lokálním souborům mobilního zařízení definovaných ve standardu JSR 75. Tyto limitující faktory zužují okruh mobilních telefonů, na kterém bude aplikace fungovat, ale s jistotou mohu říct, že tento okruh je většina mobilních telefonů, které se dnes na trhu vyskytují. Jediným možným řešením je tyto požadavky obejít, stahovat data z internetu, a tím bych nesplňoval podmínku offline aplikace, zároveň bych se dostal do sféry ostatních aplikací, které jsou již dostupné na našem trhu.

3.5 Výběr způsobu implementace

Jak jsem již zmínil, aplikace je psána v Java ME pro zajištění kompatibility na většině mobilních telefonů s podporou Javy. Dle nabyté zkušenosti s vývojem Java SE aplikací jsem si vybral vývojové prostředí Netbeans s podporou mobility a SDK Sun Java Wireless 2.5.2.

3.6 Návrh řešení

Návrh řešení je inspirován z existující aplikace webového portálu Atlas. Tato inspirace se týká organizace obrázků, výpočtu vzdálenosti a přiblížení či oddálení mapy. Od internetové aplikace je mobilní řešení značně omezeno, protože všechna data musí být dostupná offline.

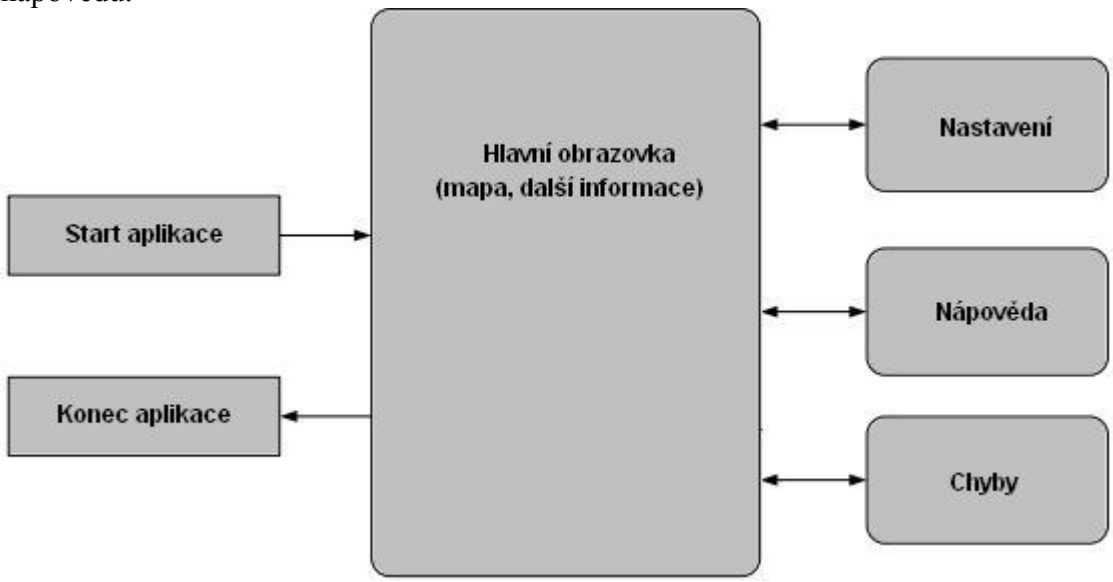
4 Realizace

Realizaci aplikace vytvořenou v programovacím prostředí Netbeans bych rád nastínil v této kapitole a podrobně rozebral určité prvky, které zabraly ve vývoji největší čas.

Celá aplikace je rozdělena do tří balíků. První balík obsahuje hlavní třídu MIDlet dle specifikace a další třídy a metody starající se o správné zpracování dostupných dat, v druhém balíku jsou třídy a metody kXML parseru volně dostupným pro MIDP zařízení, aplety apod, který zpracovává vstupní XML soubory. Poslední balík obsahuje ikony a obrázky aplikace.

4.1 Balík main

Snahou bylo použít co nejméně tříd a implementovat model (Model), zobrazeni (View) a řízení (Control) do jedné třídy. Vznikla třída Map zděděná od MIDlet, kterou musíme zavést podle specifikace, třída MenuList, která zobrazuje menu aplikace, dále třída ExternMap pro zobrazování mapy, pro připojení k souboru třída FileBrowser a poslední třída mMath rozšiřující matematickou knihovnu Javy java.lang.Math. Dialogový model aplikace je vidět na obrázku 4-1. Po spuštění aplikace je zobrazena hlavní obrazovka, kde je vidět mapa. V případě potřeby se dají zobrazit doplňkové informace jako např. GPS souřadnice či volná dostupná paměť telefonu. Pomocí menu se pak uživatel může dostat na další obrazovky, na chyby, které z nečekaného důvodu nastaly, nastavení nebo nápovědu.



Obrázek 4-1 Model uživatelského rozhraní

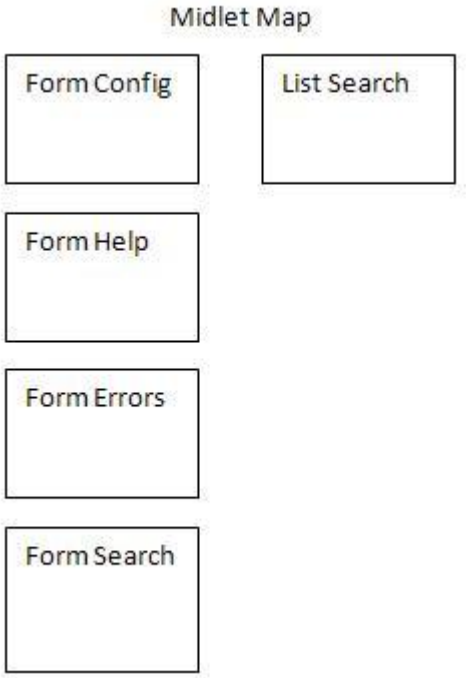
4.2 Třída MenuList

Tato třída nedělá nic jiného než že, zobrazuje menu aplikace. V podstatě se vypíše seznam možností, co si může uživatel vybrat, od načtení mapy až po ukončení celé aplikace. Jak menu aplikace vypadá, přibližuje následující obrázek 4-2 emulátoru Netbeans.

Obrázek 4-2 Menu aplikace

4.3 Třída Map

V této třídě jsou obsaženy všechny formuláře a obsluhující rutiny aplikace kromě vykreslování samotné mapy. Najdeme tu např. formulář nastavení, nápovědu a chyby kam se zapisují všechny výjimky, které mohly během spuštěné aplikace nastat. Pokud existuje instance třídy ExternMap pak se v menu zobrazí položka „Hledej“, která vyhledává města a jejich pozici podle zadaných prvních znaků nebo GPS souřadnic. Obsah objektů třídy je zobrazen na obrázku 4-3.



Obrázek 4-3 Obsah třídy Map

4.4 Třída World

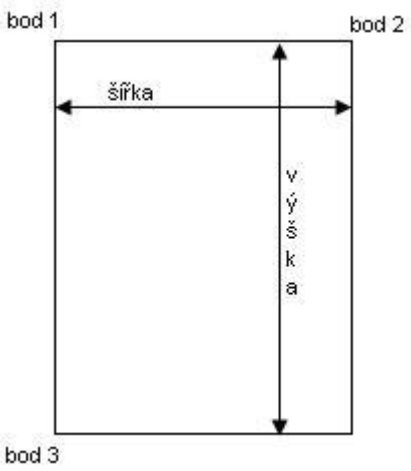
Při spuštění aplikace se inicializuje tato třída a vykreslí mapu Světa. Tato mapa je uložena přímo v JAR souboru aplikace. Na této mapě je možné jen procházet, funkce přiblížení, oddálení či hledání nejsou k dispozici. Výpočet GPS souřadnic je vypočten následujícím způsobem. V této třídě jsou napevno uloženy hodnoty GPS souřadnic bodů. Které body to jsou je vidět na obrázku 4-4. Pro výpočet GPS šíře a délky je využito následujících vzorců.

$$rozdilSire = \frac{(bod1sire - bod3sire)}{vyskaObrazku}$$

$$rozdilDelka = \frac{(bod2delka - bod1delka)}{sirkaObrazku}$$

$$poziceSire = bod1sire - (rozdilSire * pozicevY)$$

$$poziceDelka = bod1delka + (rozdilDelka * posicevX)$$



Obrázek 4-4 body pro výpočet GPS

Dle uvedených vzorců výše, každý bod obsahuje dvě hodnoty, hodnotu šíře a délky, která je uvedena ve stupních. Proměnné *pozicevY, pozicevX* obsahují počet pixelů, kde se aktuálně nachází pozice kurzoru. Výsledek proměnných *poziceSire* a *poziceDelka* je ve stupních a převod na stupně, minuty vteřiny je podle následujícího pravidla.

Převod Stupně° (D.ddd°) na Stupně° Minuty' Vteřiny (D° M' S")[25]:

```
D° M' S" = D° (0.ddd * 60)' (0.ddd * 3600)"
```

[25] Zdroj http://wiki.geocaching.cz

4.5 Třída ExternMap

Tato třída obsahuje vše potřebné pro vykreslení samotné mapy. Je zapotřebí ji předat cestu rootovského souboru XML ve kterém je obsažena cesta k dalšímu souboru XML, který obsahuje x, y souřadnice a název daného obrázku.

4.5.1 Struktura XML souborů

```
<?xml version="1.0"?>
<mapyZoom>
  <zoom id="1" src="/mapy/zoom1/mapa.xml"></zoom>
  <zoom id="2" src="/mapy/zoom2/mapa.xml"></zoom>
  <zoom id="3" src="/mapy/zoom3/mapa.xml"></zoom>
  <zoom id="4" src="/mapy/zoom4/mapa.xml"></zoom>
```

Obrázek 4-5 Ukázka struktury root.xml

```
<?xml version="1.0"?>
<mapy koeficient="1" gpsLH="50*55',11*37'" gpsPH="50*57',11*51'" gpsLD="50*47',11*4
<img x="0" y="0" src="0_0.png"></img>
<img x="0" y="1" src="0_1.png"></img>
<img x="0" y="2" src="0_2.png"></img>
<img x="0" y="3" src="0_3.png"></img>
<img x="0" y="4" src="0_4.png"></img>
<img x="0" y="5" src="0_5.png"></img>
```

Obrázek 4-6 Ukázka struktury mapa.xml

Na obrázku 4-6 je ukázka root.xml na kterém jsou vidět údaje potřebné pro pozdější správný chod aplikace. Atribut id elementu zoom obsahuje číselnou hodnotu, která nám říká, v jakém zoomu jsou obrázky na cestě udané v atributu src. Tato cesta odkazuje na další XML soubor, jehož ukázku vidíme na obrázku 4-7. V elementu img jsou atributy x, y a src. Ze souřadnic x a y se skládá celá mapa, obrázkem 4-8 nastíním filozofii rozmístění obrázků.

X=0 Y =0	X =1 Y = 0	....	X=n-1 Y=0	X=n Y=0
X=0 Y=1	X=1 Y=1	....	X=n-1 Y=1	X=n Y=1
⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮
X=0 Y m-1	X=1 Y=m-1	....	X=n-1 Y=m-1	X=n Y=m-1
X=0 Y=m	X=1 Y=m	....	X=n-1 Y=m	X=n Y= m

Obrázek 4-7 Rozmístění obrázku do mapy

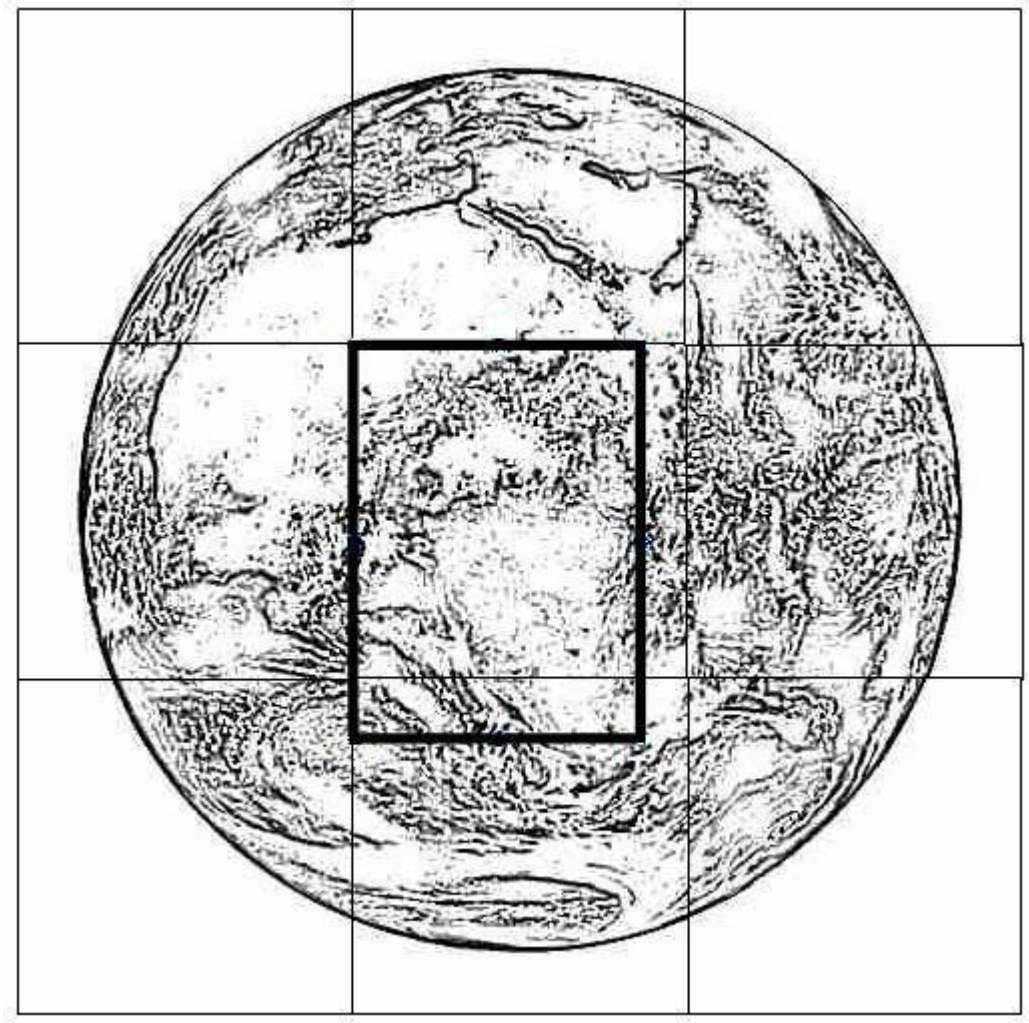
Někdo může namítnout, proč není použit kartézský souřadný systém, který inkrementuje y směrem nahoru a je nám známější z matematiky. Důvod je prostý, zdrojem dat jsou mapy ze serveru Atlas a ten používá souřadný systém uvedený na obrázku. To ovšem není vše, myšlenka vývojářů byla přenesena i na pojmenování souborů. V praxi to vypadá následovně. Mapa, která je zobrazena na x-ové souřadnici hodnoty 10 a y-ové hodnoty 7 má název souboru 10_7.png. Díky této skutečnosti se dá jednodušeji provázat XML soubor s daty získanými z portálu Atlas a vše lépe automatizovat. Dalšími údaji, které jsou na obrázku 4-7, a nebyly dosud zmíněny, jsou atributy koeficient a GPS** obsažené v kořenovém elementu „mapy“. Využití koeficientu vysvětlím v následujících řádkách. Pokud budeme využívat zdroj dat map Atlasu a zachovávat jejich filozofii pojmenování souřadnic, zjistíme, že např. soubor zoomu 1, kde je zobrazena Praha je na souřadnicích x=11 a y=7. Jeden zoom přeskočíme, potom jméno soubor následujícího zoomu Prahy najdeme na souřadnicích x=44 ,y=28.

To je 4 násobek původních souřadnic. Abychom jsme se „strefili“ do podobného místa, nové x a y je vypočteno podle rovnice:

$$noveX = stareX * koeficient + (\frac{koeficient}{2} - 1)$$

To znamená, že pro koeficient = 4 je k novému x připočtena 1, pro koeficient =8 připočtena 3 atd.

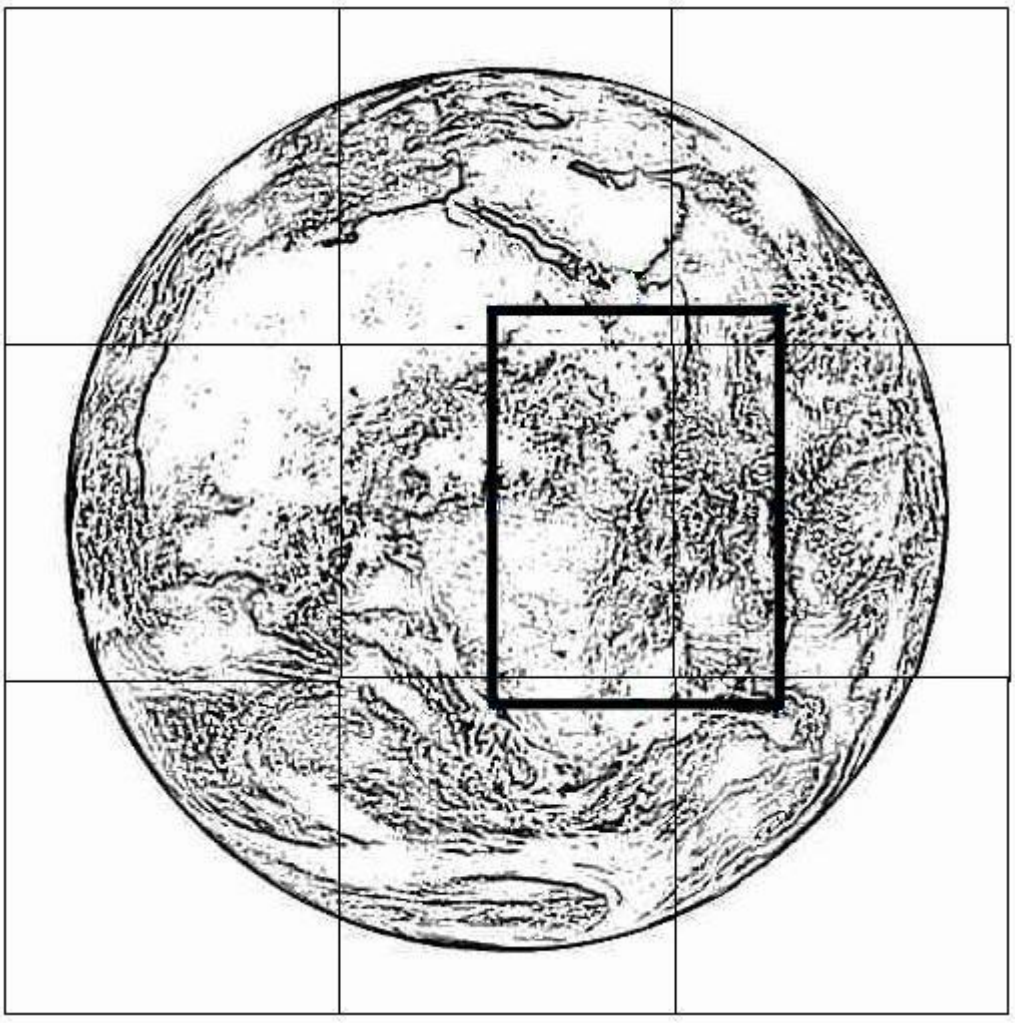
4.5.2 Zobrazení obrázků na displeji mobilního telefonu



Obrázek 4-8 Zobrazení obrázků

Třída Canvas (plátno) poskytuje přímý přístup k obrazovce zařízení MIDP. Mimo jiných metod obsahuje metody getWidth() a getHeight(), které nám vrací šířku a výšku v pixelech, kterou můžeme využít pro kreslení objektů, v našem případě převážně obrázků. Při vykreslení obrázku může dojít k tomu, že nebude využit celý displej mobilního telefonu. Pro lepší vysvětlení přejdu k příkladu. Máme obrázky tvořící mapu skládající se z 256x256 pixelů a displej mobilního zařízení, který má plochu použitelnou pro vykreslení rovnou 240x320 pixelů. Jak je vidět, výška displeje je vyšší než obrázek.

Abychom zaplnili volné místo displeje, musíme načíst ještě jeden obrázek. Toto je vidět na obrázku 4-9, silně vybarvený rámeček představuje displej a je potřeba načíst všechny obrázky, které jsou uvnitř rámečku, v tomto případě dva obrázky. Pro procházení po mapě jsem využil metody translate() třídy Graphics. Tato metoda posouvá počáteční bod. Pokud se posuneme v x-ové i y-ové ose o určitý počet pixelů, nastane situace vyobrazená na obrázku 4-10. Abychom zamezili zobrazení prázdného místa na displeji, musíme načíst 6 obrázků.



Obrázek 4-9 Zobrazení obrázků - posun

Ukázka posunu souřadného systému o počet pixelů v x a y a vykreslení obrázku imC:

```
g.translate(translatex, translatey);
try {
 g.drawImage(imC, 0, 0, Graphics.LEFT | Graphics.TOP);
    } catch (Exception e) {
                midlet.errors.append(e.toString);
                            }
```

4.5.3 Parsování XML souboru

XML neboli eXtended Markup Language je jazyk pro distribuci dat. XML je obyčejný dokument, který je nosičem dat a lze ho editovat pomocí běžných textových editorů. Základním požadavkem na správně strukturovaný XML dokument je, aby byl uzavřen v jednom elementu. Asi takto

<dokument>
nějaký text nebo další element
</dokument>

Dále můžeme do elementu vložit atributy a s takto strukturovaným dokumentem pracovat dle libosti. V našem případě využíváme takovouto strukturu:

```
          <mapy>
<img x="0" y="0" src="0_0.png"></img>
          </mapy>
```

Abychom mohli pracovat s XML dokumenty, potřebujeme příslušný parser. Pro zjednodušení práce jsem využil knihovny kXML volně dostupné na internetu. Tato knihovna patří do tzv. Pull parserů. Pull parser má tu vlastnost, že čte dokument po částech. Výhodou takových parserů jsou menší paměťové nároky. Naopak nevýhoda je, že zabírají více místa ve výsledném balíku JAR. Zde byla dána přednost menším paměťovým nárokům před větší velikostí výsledného JAR balíku.
Příklad parsování XML dokumentu:

```
parser = new XmlParser(inStreamReader);
            if (!endDocXml) {
                    ParseEvent event = parser.read();
                    System.out.println("counter: " + counter);
                    switch (event.getType()) {
                        case Xml.START_TAG:
                            try {
                                if (event.getName().equals("img")) {
                                    counter++;

                        str[0]=event.getAttribute("x").getValue();
                        str[1] = event.getAttribute("y").getValue();
                        str[2] = event.getAttribute("src").getValue();
                                    try {

                                    hashPicture.put(retVectorsVal(str[0]
                                    )  +  "_"  +  retVectorsVal(str[1]),
                                    retVectorsVal(str[2]));
                                    } catch (Exception e) {

                                    midlet.errors.append(e.toString());
                                    }

                                }
```

Tato ukázka kódu zobrazuje co se děje pokud v XML souboru dojdeme k tagu img. V jeho „těle“ jsou obsaženy potřebné informace daného obrázku. Do pole *str* se uloží atributy x, y a src obsažené v tagu. K provázání souřadnic x a y se src, který obsahuje název příslušného obrázku, je využito hashovaní tabulky, kde parametrem klíče je x, y a hodnotou je název objektu, který se má na daných souřadnicích zobrazit.

Pro zobrazení obrázku na daných souřadnicích je využito funkce *getObr()* s parametry souřadnic x, y, která vrací, pokud existuje, obrázek na těchto souřadnicích.

```
private Image getObr(int x,int y){
String namePict = (String) hashPicture.get(x + "_" + y);
            axisPict = "x=" + x + " " + "y=" + y;
            if (namePict != null) {
                cestaObrazku = cestaKSoub.concat(namePict);
                    setInputStream(cestaObrazku);

If(inStream!=null)
retValue = Image.createImage(Image.createImage(inStream));
else
retValue=null;
}
```

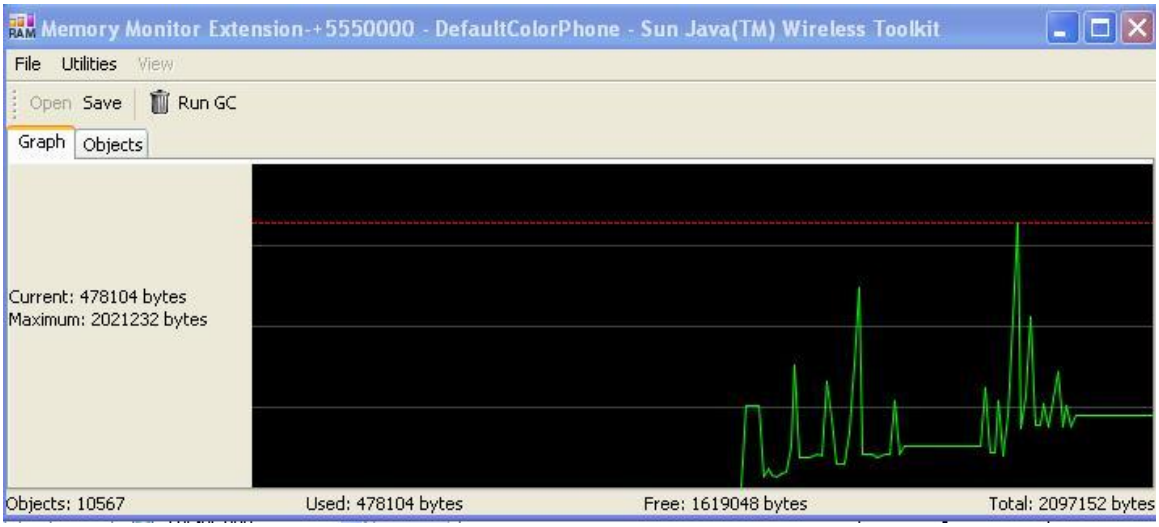
Z hashovaní tabulky podle klíče, kterým jsou souřadnice obrázku, zjistíme cestu k souboru. Pokud obrázek na dané cestě existuje, vytvoří se stream obrázku a z něj se vytvoří objekt obrázku, v opačném případě se žádný obrázek nevytvoří a ve vykreslovací funkci se načte defaultní obrázek, představující chybějící část.

5 Testování

Testování je nedílnou součástí vývoje aplikace. Pro nalezení co možná nejvíce chyb je potřeba použít vhodné nástroje a postupy. Typů testů je více, ale pro takovýto projekt postačí, pokud se budeme soustřeďovat na funkční testování a na testování použitelnosti. Testovat mobilní aplikaci je záludné, nevíme totiž, jaké prostředky cílené mobilní zařízení má. Použití emulátorů také není ideální, protože jak se již několikrát ukázalo, aplikace se někdy může na mobilním telefonu chovat jinak než na emulátoru.

5.1 Funkční testování

Pro testování funkčnosti aplikace se vytvořila turistická mapa České republiky v první úrovni a v dalších třech úrovních přiblížení hlavního města Prahy. Testování funkčnosti aplikace byla laděna v emulátoru Netbeans a fyzicky na mobilním telefonu Nokia 6500. K ladění bylo využito také nástroje Memory Monitor, který umí monitorovat alokaci paměti a také spustit Garbare Colector, který nepotřebnou paměť uvolní. Tento nástroj byl velmi používán v začátcích vývoje, kdy aplikace „padala“ kvůli nedostatku paměti. Nedostatek paměti byl hlavně z důvodu nedostatečného uvolňování již nepotřebných prvků. Na obrázku 5-1 vidíme průběh spouštěné aplikace, zřetelné výkyvy v průběhu znamenají načítaní a přiblížení mapy, kde se načítá nový obsah dat a využití paměti je závislý na počtu položek, který se musí načíst.



Obrázek 5-1 Memory Monitor

5.2 Testování použitelnosti

Toto testování se používá proto, abychom zjistili, zda je uživatelské prostředí lehce ovladatelné a pochopitelné. Typicky koncovému uživateli zadáme několik úkolů a sledujeme, jak si poradí s jejich vyřešením. Většinou sledujeme čas, jak dlouho vyřešení úlohy trvalo, počet chyb, kterých se uživatel dopustil, a po dokončení testu se ptáme na jeho pocit.

V našem případě tento typ testování probíhal u dvou osob, které mají běžné zkušenosti s mobilním telefonem. Měli za úkol načíst mapu, vyhledat město v mapě, nastavit jiný posun a přiblížit či oddálit se v mapě. Když nastala situace, při které si nevěděli rady, požádali o radu přísedícího, který znal kompletní rozhraní hry.

Během testu bylo získáno mnoho podnětů, na jejichž základě se změnila nápověda a popisky v nastavení. Další vylepšení mohou být budoucím rozšířením aplikace.

5.3 Uživatelské prostředí

V této kapitole zobrazím screenshot emulátoru představující 5. edici systému S40 mobilního telefonu Nokia. Při spuštění aplikace se zobrazí mapa Světa jak je tomu na obrázku 5-2. Pokud chceme mapu, která byla uložena do mobilního telefonu uživatelem, přejdeme do menu a najdeme soubor root.xml. Tento postup je vidět na obrázku 5-3.

Obrázek 5-2 Spuštění aplikace – „world“

Obrázek 5-3 Načtení root.xml představující mapu

Nevýhodou některých mobilních telefonů je, že při přístupu na paměťové úložiště se telefon zeptá, zda chceme tuto akci povolit. Tento dotaz je vidět na obrázku 5-3. Na obrázku 5-4 je zobrazeno, načtení a následné zobrazení mapy.

Obrázek 5-4 Načtení a zobrazení mapy

5.4 Porovnání s konkurencí

V kapitole 1.2 kde jsem zmiňoval existující řešení, mohu svou aplikaci srovnat s programem TrekBuddy, který jako jediný z uvedených, využívá paměťového úložiště pro uložení zdrojových dat. TrekBuddy využívá místo XML souborů dva textové, v jednom je uložen seznam obrázků a v druhém je uvedeno jak tyto obrázky „poskládat“. Tyto dva soubory spolu se samotnými obrázky map jsou uloženy v jednom komprimovaném TAR[26] souboru. Výhodou takového řešení je, že se soubor načetl jednou a v mapě se dá plynule posouvat. Obešlo se také dotazování aplikace, která se ptá na přístup do souborového systému mobilního zařízení. Pokud telefon nepodporuje vypnutí tohoto dotazování, aplikace TrekBuddy má v tomto ohledu navrch. Stejně tak jako v mé aplikaci existuje možnost vytváření svých vlastních map i v TrekBuddy. Tato možnost se jeví jako silná zbraň obou aplikací. Díky delšímu vývoji má TrekBuddy větší podporu mapových podkladů než má aplikace, ale to nevidím jako závažný nedostatek. Srovnáním obou aplikací docházím k názoru, že u obou aplikací je pořád co k vylepšení a aplikace mají jiné cíle vývoje. TrekBuddy se snaží navigovat dotyčného na určené místo díky připojenému GPS modulu, kdežto má aplikace má za úkol podat přibližnou představu kde se dotyčný nachází.

[26] Zkratka z anglického tape archiver neboli páskový archivovač

6 Závěr

Během práce se podařilo vytvořit aplikaci zobrazující mapy, obsažené na paměťovém úložišti mobilního telefonu. Na vytvořené mapě je možnost přibližování, oddalování v několika úrovních a vyhledávání měst. Data jsou uložena v datové struktuře XML.

Druhotným výsledkem je možnost zobrazení mapy světa zobrazující GPS souřadnice. Zobrazení GPS je možně i v externí mapě, ale musí být vloženy správná data do XML dokumentu, který danou mapu zobrazuje.

Tato práce byla velkým přínosem, která přinesla nové zkušenosti s programováním v Java ME a vývojem na mobilních zařízeních.

7 Seznam literatury

[1] K. Topley J2ME V KOSTCE – Pohotovostní referenční příručka, volume 1. Grada Publishing.
[2] B. Mobilmania Odkud se vzal symbian ? http://unleaded.blog.mobilmania.cz.
[3] Petr Dytrych Programování Java aplikací pro mobilní telefony – hra Piškvorky. 2008. ČVUT. Bakalářská práce.
[4] Wikipedie Java ME http://cs.wikipedia.org/wiki/Java_ME.
[5] Otakar Čerba Mapy na internetu Zapadočeská univerzita v Plzni. 2006
[6] Michal Krátký Mapy na internetu Univerzita Karlova v Praze 2004. Ročníková práce
[7] The motorola developer network http://developer.motorola.com/
[8] Nokia http://www.nokia.com/developers
[9] Sony Ericsson http://developer.sonyericsson.com/site/global/home/p_home.jsp
[10] Sun Developer Network http://java.sun.com/
[11] Seriál programování aplikací J2ME http://interval.cz/vyvoj-aplikaci/j2me/

Všechny webové zdroje byly k 28.12.09 v provozu.

8 Obsah přiloženého CD

- **examples**/ screenshoty aplikace
- **src/** zdrojové soubory
- **text/** text této práce