FACHHOCHSCHULE STUTTGART HOCHSCHULE FÜR TECHNIK

STUTTGART UNIVERSITY OF APPLIED SCIENCES
Department of
Geomatics, Computer Science, and Mathematics

M.Sc Thesis

Design and Implement a Cartographic Client Application For Mobile Devices using SVG Tiny and J2ME

By
LI Hui

Stuttgart, March 2006

Design and Implement a Cartographic Client Application

For Mobile Devices using SVG Tiny and J2ME

A dissertation
presented in partial fulfillment of the requirements for
the degree of Master of Science in the
Department of Geomatics, Computer Science and Mathematics
of the University of Applied Sciences Stuttgart

By
LI Hui

****

Fachhochschule Stuttgart – Hochschule für Technik
University of Applied Sciences
March 2006

Supervisors:
Prof. Dr.-Ing. Franz-Josef Behr
Prof. Dr.-Ing. Dietrich Schroeder

Approved by:

____

27.03.2006, Supervisor

Master Course Photogrammetry and Geoinformatics

Acknowledgement

I would like to thank my supervisors Prof. Dr. Franz-Josef Behr and Prof. Dr.-Ing. Dietrich Schroeder who are the supervisors of my thesis work for their discussion and positive criticism. I would like to thank many unknown developers in the java forum who helped me a lot to solve the programming problems. I thank my parents and friends who are always supporting me.

Abstract

The fields of geoinformation technology and cartography have seen dramatic changes in the last decade. The dissemination of digital geospatial data is no longer bounded by the desktop platform. It is available now on mobile devices such as PDAs (personal digital assistants), smart-phones etc.

The mobile devices which support J2ME (Java 2 Micro Edition) offer the users and developers one open interface, the application developers and users can develop or download the software according their own demands. At present, J2ME has been one of the most popular mobile devices development platforms.

Now WMS (Web Map Service) can afford not only traditional raster image, but also the vector image. SVGT (Scalable Vector Graphics Tiny) is one subset of SVG (Scalable Vector Graphics) points to mobile devices. Because of its precise vector information, original styling and small file size, SVGT format is fitting well for the geographic mapping purpose, especially for the mobile devices which has bandwidth net connection limitation.

The aim of this research was to develop a cartographic client for the mobile devices, using SVGT and J2ME technology. Mobile device was simulated on the desktop computer for a series of testing with WMS, for example, send Getcapabilites and GetMap request , get the responding data from WMS and then display both vector and raster format image.

Analyzing and designing of System structure such as user interface and code structure were discussed, the limitation of mobile device should be taken into consideration for this applications. The parsing of XML document which was received after the GetCapabilites request and the visual realization of SVGT and PNG (Portable Network Graphics) image are important issues in codes' writing. At last the client was tested on Nokia S40/60 mobile phone successfully.

Keywords: Cartographic visualization, J2ME, MIDP, SVG, SVGT, PNG, WMS, Getcapabilites request, GetMap request, Mobile applications, Mobile cartography, Scalable Vector Graphics Tiny, Web Map Service, XML parser, Java 2 Micro Edition

Table of Content

Table of Figures

Table of Tables

Table of Listings

1 Introduction

1.1 Backgroud

In the nineties, Web maps and digital maps became popular due to the success of the Personal Computers and the Internet. These mapping technologies allow personalization and adaptation, are easily deliverable, revised in shorter times, can incorporate multimedia, and may be interactive. Now the popularity of mobile devices such as PDA's, mobile phones etc. and the availability of development environments such as J2ME (Java 2 Micro Edition) for such devices have made possible to design and develop of new kind of cartographic client software. No doubt that in the future cartographic data is not PC centric but would be available on mobile systems.

But the map display on a handheld device is a challenge to cartography due to the limiting factors such as screen size, colors, resolution, processing power, memory and power supply. And people will demand more advanced reliable applications. J2ME was born to help solving such problems. In this Java based development environment, the Connected Limited Device Configuration (CLDC) and the Mobile Information Device Profile (MIDP) are specifically designed for the wireless devices.

In the earlier years, the Web Map Service (WMS) mostly based on raster image format. Due to big size and limited bandwidth, the raster image format is also one obstacle to the data transforming from PC to mobile device. But after the birth of vector image format, every thing has changed. The Scalable Vector Graphics (SVG) is an open standard for encoding geographic information in a XML language defined by W3C and it becomes one part of WMS recommendations. As one subset of SVG, Scalable Vector Graphics Tiny (SVGT) is not related to any specific hardware or software platform: data encoded using it can be easily read and understood by any programming language and software system able to parse XML streams. SVGT encodes vector geographical information together with metadata on spatial and non-spatial resources.

In this application, SVGT and J2ME are collaborating to design and implement a cartographic WMS client for mobile devices.

1.2 Problem definition

1.2.1 Limitation of mobile devices

The popularity of handheld devices and mobile Internet gives a new platform to geoinformation. But cartographic visualization for small displays of mobile devices is restricted by several technical limitations such as the small display size and resolution, lack of processing power and memory, most critical the battery life, and the mobile network bandwidth. When writing J2ME codes, those limitations should be considered, especially for memory, screen size, and network bandwidth.

1.2.2 XML document parsing

Once the users send the GetCapabilities Request to the WMS, the client gets a XML format file as the responded data. The parameters of the GetMap Request URL in the content of this XML document should be picked up and be set in the GetMap Request URL. Because every WMS has its own Document Type Definitions (DTD) for GetCapabilities Request XML document, it is a big challenge to find one common solution to parse those different XML document.

1.2.3 SVGT/PNG Viewer using and designing

To display the result map file, SVGT *TinyLine* or PNG raster image format viewer will be loaded depending on which kind of image formats supported by the WMS.
How to implement the Tinyline SVGT viewer as one embedded part of the client? How to read and redraw the PNG image file?

1.2.4 User interface designing

This client should be designed as one complete WMS client with full functions that are same as the PC client. Thus it should include several basic interface for the users to select server address, input coordinate of the search point, select layers and corresponding Coordinate Reference System (CRS), and zoom and pan the result map, etc. It is not easy to let those interfaces run without mistakes in one clear workflow.

1.3 Motivation

The reason to choose this domain is that this is an upcoming field and has a lot of scope from a research point of view. Further, the client designing on the mobile devices seems interesting and challenging, and till now most of all the WMS clients for the mobile phone are based on raster image format, not vector image format, which was also a motivation to choose this as research area.
Moreover the coming times are going to large-scale development of commercial applications based on location based services. Such application can be extended in many aspects, such as navigation, mobile service, library service and so on.

1.4 Methodology

According to the objectives, the corresponding methodology is established.

1. **Pre-Analysis**: The current situation will be analyzed, including the study of the theory part of J2ME and SVGT, how to parse XML document in mobile devices etc.
2. **Selection of scenario**: The testing system will be selected, including the testing software and servers. Several Web Map servers which produce SVGT (SVG) map or raster image will be as the testing server.
3. **Design the code structure and user interface**: In this step, the code structure and user interface should be designed as one clear workflow in advance.
4. **Debug the codes**: The codes will be debugged carefully and using some virtual simulator to test it.
5. **Test of client**: The client will be tested both on mobile phone and on simulator, trying to connect different WMS to check its performance.

1.5 Thesis structure

The structure will try to follow a logical sequence to achieve the objective of the study. This chapter introduced the problems and challenges of visualizations for mobile devices and outlined the goals and contributions of this thesis. The remainder of the thesis is structured as follows:

Chapter 2: Background of cartographic client on mobile devices
This chapter reviews the existing mobile devices and Location Based Service. It explains the definitions of WMS, image format, SVGT and J2ME, XML parser which play important role in the following research. At last some similar earlier application will be introduced.

Chapter 3: Implementation of this application
This chapter describes the workflow of how to implement the client. At first chooses the testing WMS which support SVG image and PNG raster image, coordinate transformation is explained. After that, describes the code structure in both users' and developer's views, and user interface, J2ME Record Management Store and method to use kXML to parse the XML document, and method to use TinyLine SVGT SDK.

Chapter 4: Conclusions and outlook
This chapter discusses about the conclusions of client test on simulator and on Nokia S40/60 mobile phones. Further in conclusion the summary of whole work is given and scope for further research.

2 Background

2.1 Mobile application/Location based services

2.1.1 Location based services

Nowadays the telecom industry is eying at the marriage of geoinformation services and mobile devices in the form of Location Based Services (LBS). In general case, LBS can be defined as services utilizing the ability to dynamically determine and transmit the location of persons within a mobile network by the means of their terminals. LBS provide specific, relevant information based on the current position to the user. The Open Geospatial Consortium (OGC 2003) defined LBS as a wireless-IP service that uses geographic information to serve a mobile user or any application service that exploits the position of a mobile terminal.

LBS/Mobile applications cover many aspects linked to human mobility such as:
Navigation,
Health/Safety/Security/Emergency,
Convenience,
Entertainment,
Travel Aids,
Productivity Aids,
Mobile Work Force Management etc

For an example a visitor want to find the nearest hotel, he needs nothing about names and addresses of those hotels which are within his reach, LBS will tell him the hotels in say 1 sq.km according to his position, selected out of the database of say 1000 hotels in the city spread over 2000 sq.km.

WMS can be chosen to use as server part of LBS, and one typical LBS system could be the followings: The user has one instrument (for example, GPS), it sends the position parameters of the user to the WMS though wireless net, and the WMS responses with the service according to the position, the data can be received from the

Database (for example, Oracle spatial data). At last the data requested was sent back to the Mobile Client as map in form of image. This application can be seen as one user part of the LBS system, see Figure 1.

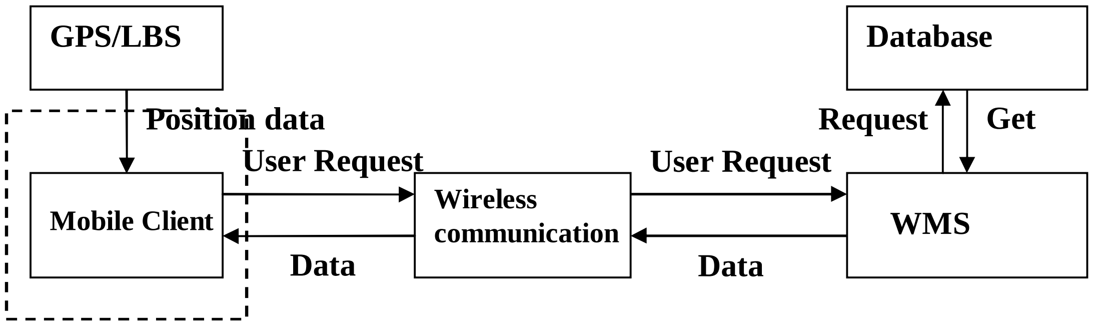


Fig. 1: This application can be seen as one user part of the LBS system

In this system, because of the limited ability of the mobile client, the most of collection and calculation of the data should be performed in WMS, and the Mobile client only sends request and receives the result data.

2.1.2 Mobile devices

Mobile device is a wide term covering information terminals, information appliances, Personal Digital Assistant (PDA), mobile phones etc. According to Gartner (2003), mobile devices can be broadly classified into two categories": Cellular phones with increasing computing capabilities, including the display of graphics and the enabling of interaction via graphics (Smartphones, Communicator) and Portable computers, which can be upgraded to voice communication capabilities (e.g. PDA)

Each mobile device has its own Operation System (OS), which is to give user interface and control the devices synchronous. Now there are four main OS in the market: Symbian, Smartphone, Palm OS, Linux. Symbian was built by Nokia, Motorala and Erison. In 2005, the mobilephones that have been installed Symbian OS occupy 51% of the smartphnes in the world (Internetnews.com[1], 2006). Though, for

[1] http://www.internetnews.com/wireless/article.php/3584431

this research the considered devices are the mobile phone model which has Symbian OS.

2.2 Web Map Service

Web Map Service (WMS) is an International Standard (ISO 19128: 2005 Geographic information - Web map server interface) which defines a "map" to be a portrayal of geographic information as a digital image file suitable for display on a computer screen. WMS is a specification that produces maps of spatially referenced data dynamically from geographic information. WMS-produced maps are generally rendered in a pictorial format such as PNG, GIF or JPEG, or occasionally as vector-based graphical elements in Scalable Vector Graphics (SVG) or Web Computer Graphics Metafile (WebCGM) formats.

An OGC compliant WMS applies to three different requests: GetCapabilities, GetMap and GetFeatureInfo. GetCapabilities and GetMap requests are mandatory, GetFeatureInfo is an optional request. Each server that follows the Web Map Service specification has to provide the interface to the users to input the standard parameters and if these parameters are valid the server must serve the data corresponding to the request. The methods to send the request from a client to a server are various. If the client is a browser, the parameters are sent inside the Uniform Resource Locator (URL); if the client is mobile device the request is sent also in URL though wireless net.

For GetCapabilities request, if the parameters are inputted correctly, the service-level metadata in form of XML document will be returned, including the information about the image formats and layers which are available in the server, the corresponding coordinates system names, the bounding box value for each coordinates system, the map formats and the contact person and so on.

The parameters of the GetCapabilities request are shown in Table 2:

Request Parameter	Mandatory Optional	Description
VERSION=version	O	Request version
SERVICE=WMS	M	Service type
REQUEST=GetCapabilities	M	Request name
FORMAT=MIME_type	O	Output format of service metadata
UPDATESEQUENCE=string	O	Sequence number or string for cache control

Tab. 1: The parameters are required by the GetCapabilities request (OPEN GEOSPATIAL CONSORTIUM, Web Map Service Specification, 2004)

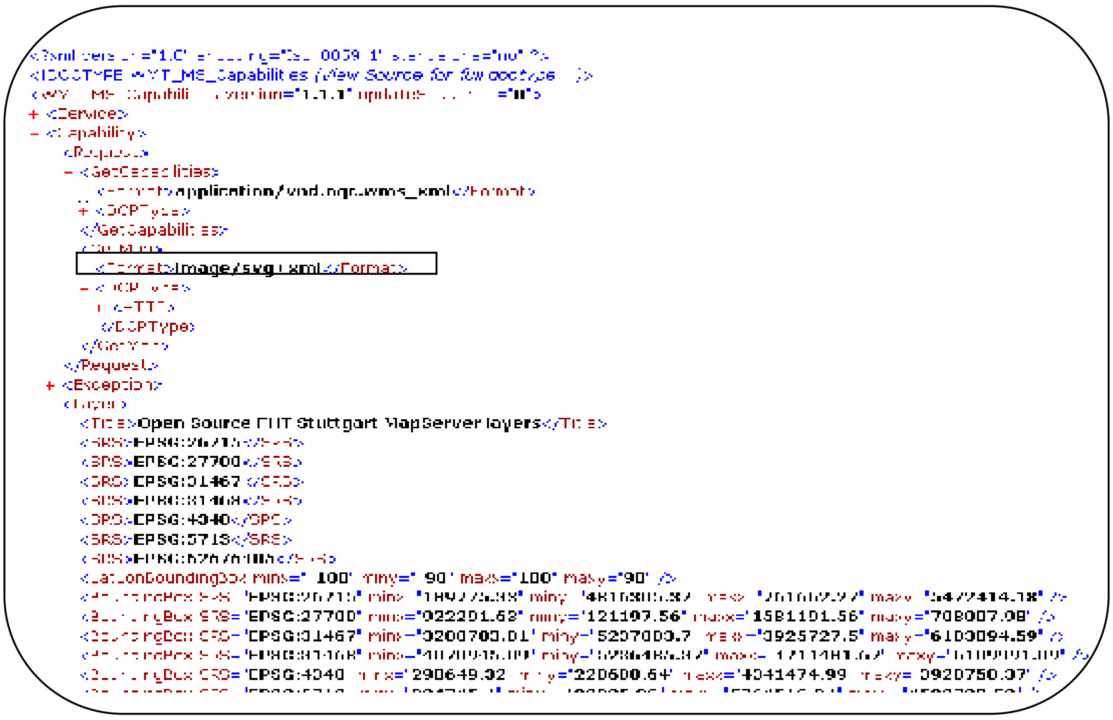
A valid GetCapabilities request should include all the parameters which specified as mandatory in Table 2. The format of GetCapabilities request in a URL likes this:

```
http://www.gis-news.de/wms/getmapcap.php?VERSION=1.1.1&SERVICE=WM
S&REQUEST=GetCapabilities
```

Listing 1: The format of GetCapabilities request in a URL

Inside the URL, "http" is the requested Internet protocol. "www.gis-news.de" is the server name; "/wms/getmapcap.php" shows the path of the document on the server. The "?" indicates the starting of the query string. "SERVICE=WMS" defines the service type, "VERSION=1.1.1" indicates the server should support OGC WMS version 1.1.1 and "REQUEST=GetCapabilities" tells the server the user asks for the GetCapabilities service.

The response of the GetCapabilities request is a XML format document (see Figure 3).



```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE WMT_MS_Capabilities (View Source for full doctype...)>
<WMT_MS_Capabilities version="1.1.1" updateSequence="0">
+ <Service>
- <Capability>
    <Request>
    - <GetCapabilities>
        <Format>application/vnd.ogc.wms_xml</Format>
      + <DCPType>
      </GetCapabilities>
      <GetMap>
        <Format>image/svg+xml</Format>
      - <DCPType>
        + <HTTP>
        </DCPType>
      </GetMap>
    </Request>
  + <Exception>
    <Layer>
      <Title>Open Source FHT Stuttgart MapServer layers</Title>
      <SRS>EPSG:26715</SRS>
      <SRS>EPSG:27700</SRS>
      <SRS>EPSG:31467</SRS>
      <SRS>EPSG:31468</SRS>
      <SRS>EPSG:4040</SRS>
      <SRS>EPSG:5713</SRS>
      <SRS>EPSG:[illegible]</SRS>
      <LatLonBoundingBox minx="-180" miny="-90" maxx="180" maxy="90" />
      <BoundingBox SRS="EPSG:26715" minx="[illegible]" miny="[illegible]" maxx="[illegible]" maxy="[illegible]" />
      <BoundingBox SRS="EPSG:27700" minx="922201.62" miny="121197.56" maxx="1581191.56" maxy="708007.08" />
      <BoundingBox SRS="EPSG:31467" minx="3200700.01" miny="5207000.7" maxx="3925727.5" maxy="6100094.59" />
      <BoundingBox SRS="EPSG:31468" minx="[illegible]" miny="[illegible]" maxx="[illegible]" maxy="[illegible]" />
      <BoundingBox SRS="EPSG:4040" minx="290649.32" miny="220600.64" maxx="4041474.99" maxy="3920750.37" />
      <BoundingBox SRS="EPSG:5713" [illegible]
```

Fig. 2: The fragment of the result XML file of GetCapabilities request showed in browser

In the next step, the user selects the value of the parameters of GetMap request depending on the information from the GetCapabilities result. For example, the information about the map formats is provided in the GetCapabilities result document. In this case, *<Format>image/svg+xml<Format>* means that the available map format in this server is: SVG

The parameters of a GetMap request are shown in Table 3:

Request Parameter	Mandatory Optional	Description
VERSION=1.3.0	M	Request version
REQUEST=GetMap	M	Request name
LAYERS=layer_list	M	Comma-separated list of one or more map layers
STYLES=style_list	M	Comma-separated list of one rendering style per requested layer
CRS=namespace:identifier	M	Coordinate reference system
BBOX=minx,miny,maxx,maxy	M	Bounding box corners (lower left, upper right) in CRS units
WIDTH=output_width	M	Width in pixels of map picture
HEIGHT=output_height	M	Height in pixels of map picture
FORMAT=output_format	M	Output format of map
TRANSPARENT=TRUE|FALSE	O	Background transparency of map (default=FALSE)
BGCOLOR=color_value	O	red-green-blue color value for the background color (default=0xFFFFFF)
EXCEPTIONS=exception_format	O	The format in which exceptions are to be reported by the WMS (default=XML)
TIME=time	O	Time value of layer desired
ELEVATION=elevation	O	Elevation of layer desired

Tab. 2: The parameters are required by the GetMap request (OPEN GEOSPATIAL CONSORTIUM, Web Map Service Specification, 2004)

The format to send GetMap request in URL is like this:

```
http://www.gis-news.de/wms/getmapcap.php?VERSION=1.1.1&BBOX=18977
5.33,4816305.37,761662.27,5472414.18&LAYERS=airports,ctybdpy2,AUSSTA
TE1&STYLES=,,&REQUEST=GetMap&layer1=airports&layer2=ctybdpy2&layer3
=AUSSTATE1&style=&SRS=EPSG:26715&4340&minx=189775.33&miny=481
6305.37&maxx=190051.33&maxy=4816525.37&WIDTH=800&HEIGHT=600&
FORMAT=image/svg+xml&EXCEPTIONS=application/vnd.ogc.se_xml&transpar
ent=True
```

Listing 2: The format of GetMap request in a URL

Following the same principle, a valid request should include all the mandatory parameters. In the URL above, "http://www.gis-news.de/wms/getmapcap.php?" gives the server name and the path of the file resource on the server. "VERSION=1.1.1" indicates the server supports OGC WMS version 1.1.1 and "REQUEST=GetCapabilities" tells the request name. "BBOX=189775.33,4816305.37,761662.27,5472414.18" and "SRS[2]=EPSG[3]: 26715" define the bounding box which indicates the specific required region in the given coordinates system. "HEIGHT=800" and "WIDTH=600" defines the size of the image. "FORMAT=image/svg+xml" defines the output format of the image in this case it is SVG format. "LAYERS= airports,ctybdpy2,AUSSTATE1" defines the layers which requested by the user. The result of the GetMap request is a vector or raster image map to be displayed in the browser. All the parameters should be picked up from the GetCapabilities XML Document.

The GetFeatureInfo request is an option. It has not been involved in the implementation of this thesis, so it will not be discussed here.

2.3 Graphic data formats

As discussed above, the output image format is a very important parameter of the GetMap request. At the moment there are more than 44 different graphic format names, but there actually are only two basic methods for a computer to render, or store and display an image: raster image format and vector image format.

Raster image formats (RIFs) should be the most familiar to Internet users. A Raster format breaks the image into a series of colored dots called pixels. The number of ones and zeros (bits) used to create each pixel denotes the depth of color you can put into your images. There are various raster image formats available such as Tagged Image File Format (TIFF), Graphics Interchange Format (GIF), Joint Photographic Expert Group (JPEG) and Standard Windows Bitmap (BMP).

[2] Spatial Reference System
[3] European Petroleum Survey Group (http://www.epsg.org)

Vector formats are newly implemented into geographic mapping and becoming more and more popular. They are actual vectors of data stored in mathematical formats rather than bits of colored dots. This formatting falls into a lot of proprietary formats, formats made for specific programs. CorelDraw (CDR), Hewlett-Packard Graphics Language (HGL), Windows Metafiles (EMF), SVG (Scalable Vector Graphic) and WebCGM (Web Computer Graphics Metafile).

2.3.1 Raster image formats

A raster image file is generally defined to be a rectangular array of regularly sampled values, known as pixels. Each pixel (picture element) has one or more numbers associated with it, generally specifying a color which the pixel should be displayed in.

The most common file format that used for mobile device is PNG (Portable Network Graphics). PNG is a bitmap format that was developed by the World Wide Web Consortium especially for the web; it is fully streamable with a progressive display option.

The PNG format has several advantages over other graphics formats; for example, it is license free and supports true color images, including a full transparency (alpha) channel. PNG images are always compressed with a loss-less algorithm. The algorithm is identical to the algorithm used for JAR files, so the mobile implementation can save space by using the same algorithm for both purposes.

2.3.2 Vector image formats

Vector image files record images descriptively, in terms of geometric shapes. These shapes are converted to bitmaps for display on the monitor. Vector images are easier to modify, because the components can be moved, resized, rotated, or deleted independently. The W3C has produced an XML-based format called Scalable Vector Graphics (SVG). SVG has been widely used in GIS Internet mapping purpose, and it is more optimal compared to other vector image formats.

SVG Basic (SVGB) is a subset of SVG, and SVG Tiny (SVGT) is a subset of SVG Basic. These light versions of SVG are designed to accommodate cellphones and other handheld devices with limited screen real estate, memory and bandwidth. SVGT is the format involved in the implementation of this thesis. The theoretical issues of SVGT are discussed as following.

2.4 Introduction of Scalable Vector Graphics Tiny (SVGT)

2.4.1 What is SVGT?

Because of industry demand, two mobile profiles were introduced with SVG 1.1: *SVG Tiny* (SVGT) and *SVG Basic* (SVGB). These are subsets of the full SVG standard, mainly intended for user agents with limited capabilities. In particular, SVG Tiny was defined for highly restricted mobile devices such as cellphones, and SVG Basic was defined for higher level mobile devices, such as PDAs. Hence, it was decided that SVG Tiny would be a strict subset of SVG Basic, itself a strict subset of SVG Full. See Figure 3:

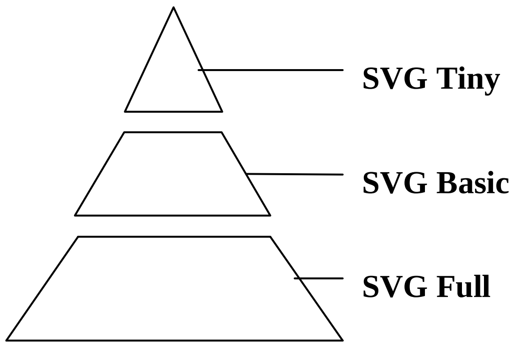


Fig. 3: Pyramid of SVG, SVGB and SVGT

2.4.2 First impression of SVGT

Since SVGT is an XML format and follows the XML grammar. SVGT grammar is based on an XML DTD. It starts and ends with an "SVG" tag. Objects are placed within the SVGT object to form graphics. Some basic drawing components are provided, such as rectangle, circle, ellipse, line, polyline and polygon; the corresponding style parameters are also defined. Furthermore, another command "`path`" is provided to describe lines and curves between points. Additionally transformations, masking, linking, temporal effects and animation are also possible.

SVGT is XML-based graphics, the source code for SVGT graphic is not held in any arbitrary text format, but an SVGT image or an SVGT document is a "well-formed" XML document. A well-formed SVGT document might have a structure which could look like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
    "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg:svg width="4cm" height="8cm" version="1.1" baseProfile="tiny" >
    <svg:ellipse cx="2" cy="4" rx="2" ry="1" />
</svg:svg>
```

Listing 3: The format of GetMap request in a URL

The first line of the SVGT file header is the standard XML processing instruction that efficiently states that this document conforms to the XML 1.0 specification, uses UTF-8 character encoding and depends on a Document Type Definition (DTD) external to the document to parse correctly. Following that might be a DOCTYPE declaration that provides information to the SVGT rendering engine (also called SVG viewer) about what structure to expect in the SVGT document. The "`DOCTYPE`" states where the DTD is located and the name of the document element it will be applied to. In this case, the DTD is applied to a document element name "`svg`". In the DOCTYPE declaration,

```
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd"
```

Listing 4: DTD in the DOCTYPE declaration

Listing 4 is the URL for the file that defines the allowable structure of an SVGT document, for example the grammar and rules for the document. More specifically, an SVGT document also must be "valid". In XML terminology, that means the SVGT image must be structured in the way demanded by the Document Type Definition

(DTD) for the SVGT language. The SVGT rendering engine needs to process an SVGT document with an appropriate, predictable structure, and the DTD helps that this process goes smoothly.

The "`svg`" element contains all other elements. It is the document element of the document. Inside this element, the developers can add their own codes which describe their graphic by applying the standard shape elements and attribute which are provided by SVGT.

2.4.3 Basic Capabilities of SVGT

The use of the word "scalable" in SVG has two meanings. First, vector graphic images can easily be made scalable; for example, they are not limited to a single and fixed pixel size. This means SVG format can be displayed on any device of any size and any resolution without changing the image clarity. This contrasts with raster image files, which are difficult to modify without loss of information. When the raster images are zoomed in to large scales, the content of the image becomes blurred. SVGT graphics don't have the blur problem because of their vector property. Figure 4 shows a comparison of the quality of a SVGT graphic and a JPEG graphic. The difference can be seen easily. When the graphics are zoomed into a big scale, the raster image becomes blurred and the borderline becomes unclear.

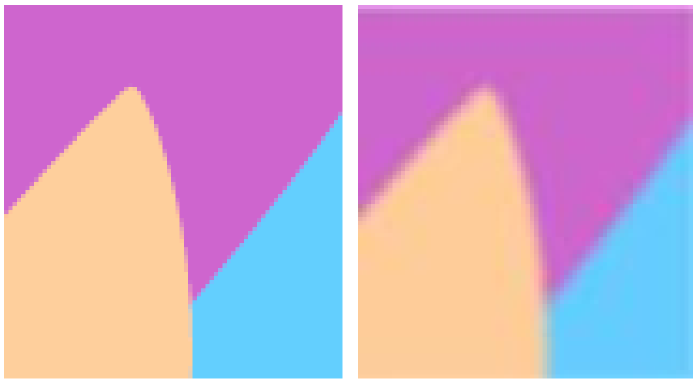


Fig. 4: Two images were created to compare the quality of SVG image (left) and raster image (right) in high scale. SVG graphic kept higher quality; raster image became blurry.

On top of basic shapes like rectangles, ellipses, circles, lines, polygons, etc, SVGT offers full support for Bezier type curves, with only the elliptical arc commands not available. The second most basic set of features that you'll be looking at in SVG Tiny will probably be text.

SVG Tiny delivers by allowing for most text operations from SVG Full, apart from text on a path and text fragments. Still in the text area, SVG Tiny supports simple SVG fonts definitions so that you can ensure your types design will be respected across all devices and most importantly scale well (as opposed to system fonts). Raster images are also supported, with the two mandated formats being PNG and JPEG.

SVG Tiny supports solid fills and strokes on paths and basic shapes. On the other hand, gradients and pattern fills are not available; they were thought to be too computing-intensive for the SVG Tiny 1.1 timeframe. Opacity, masks and clips are not there either for the same reasons.

One of the main design goals of SVG Tiny was to provide a solution for the new generation of messaging services. In order to cater to this requirement, SVG Tiny was designed to offer the full spectrum of what SVG Full offers in terms of animation capabilities. All the elements and attributes supported by SVG Tiny can be animated, using either discrete, paced, or finely-tuned interpolations, or even motion paths. All of this allows for the creation of SVG cartoons and gimmicky animations.

2.4.4 SVGT Viewer

There are quite a few available implementations for consuming SVG Tiny content on a mobile device.

The first one to mention is *TinyLine*[4], because it's easy to obtain, it's free, and it can be run on a phone that doesn't ship with a built-in SVG implementation. *ZOOMON*

[4] http://www.tinyline.com

Mobile Solutions[5] has been in the SVG game from day one, as the first mobile vendors were joining the SVG Working Group and getting work underway toward creating the SVG Mobile specification. Canadian company *BitFlash*[6] was also a very early contributor to the SVG Working Group. In that capacity, *BitFlash* has acted as a tremendous community evangelizer for SVG Mobile and committed very early to providing high-quality SVG Tiny and SVG Basic implementations. And there are many other commercial viewer softwares.
In this application, *TinyLine* viewer is chosen because it is open source and free.

2.4.5 Shipping and Announced SVG Phones

Because of the many advantages of SVGT, more and more mobile phones begin to support this format. Here is an updated list of phones that come fully equipped with a compliant SVG Tiny 1.1 implementation.

Motorola	C975, C980, E770V, E1000, i870, V3X, V975, V980, V1050
NEC	V703N, V802N
Nokia	3250, 6265, 6280, 6282, 7370, 7710, E60, E61, E70, N70, N71, N80, N90, N91, N92
Panasonic	MX6, MX7, SA6, SA7, VS3, VS7
Sagem	my-X8, my-V76, my-V85
Samsung	D600, E350, Z300, Z500, ZV10, ZV30
Sanyo	S750
Sharp	V501SH, V601SH, V602SH, V603SH, V604SH, V703SH, V703SHf, 802, V804SH, 902, V903SH
Siemens	C65, C70, C75, CF65, CFX65, CL75, CX65, CX70, CX70 Emoty, CX75, M65, M75, S65, S75, SF65, SL65, SL75, SK65, SP65
Sony Ericsson	D750, F500, K300, K500, K508, K600, K608, K610, K700, K750, M600, P990, S600, S700, S710, V600, V800, W550, W600, W800, W810, W900, W950, Z500, Z520, Z800
Toshiba	TS 803, TS 921, V902T, V903T

Tab. 3: Shipping and Announced SVG Phones

One SVGT example was offered in Appendix E.

[5] http://www.zooomon.com
[6] http://www.bitflash.com

2.5 J2ME

2.5.1 What is J2ME?

J2ME is a lean Java platform targeted specifically at applications running on small devices such as mobile phones, PDAs, Internet screenphones, digital television set-top boxes, as well as a broad range of embedded devices. Like its counterparts for the enterprise (J2EE), desktop (J2SE) and smart card (Java Card) environments, J2ME includes Java virtual machines and a set of standard Java APIs defined through the Java Community Process (JCP) by an expert group of more than 50 companies, including leading device manufacturers, wireless carriers, and vendors of mobile software.

J2ME delivers the power and benefits of Java technology to consumer and embedded devices. It includes flexible user interfaces, a robust security model, a broad range of built-in network protocols, and extern support for networked and offline applications that can be downloaded dynamically. Applications based on J2ME specifications are written once for a wide range of devices, also exploit each device's native capabilities. The J2ME platform is deployed on millions of devices, supported by leading tool vendors, and used by companies worldwide. In short, it is the platform of choice for today's consumer and embedded devices.

2.5.2 The J2ME Architecture

The J2ME architecture comprises a variety of configurations, profiles, and optional packages that implementers and developers can choose from, and combine to construct a complete Java runtime environment that closely fits the requirements of a particular range of devices and a target market. Each combination is optimized for the memory, processing power, and I/O capabilities of a related category of devices. The result is a common Java platform that takes full advantage of each type of device to deliver a rich user experience.

2.5.2.1 Configurations

Configurations comprise a virtual machine and a minimal set of class libraries. They provide the base functionality for a particular range of devices that share similar characteristics, such as network connectivity and memory footprint. Currently, there are two J2ME configurations: the Connected Limited Device Configuration (CLDC) and the Connected Device Configuration (CDC).

2.5.2.2 Profiles

To provide a complete runtime environment for a specific device category a configuration must be combined with a profile, a set of higher-level APIs that further define the application life-cycle model, the user interface, and access to device-specific properties. A profile supports a narrower category of devices within the framework of a chosen configuration. A widely adopted example is to combine CLDC with the Mobile Information Device Profile (MIDP) to provide a complete Java application environment for cell phones and other devices with similar capabilities.

2.5.2.3 Optional Packages

The J2ME platform can be extended by adding various optional packages to a technology stack that includes either CLDC or CDC and an associated profile. Created to address very specific application requirements, optional packages offer standard APIs for using both existing and emerging technologies such as database connectivity, wireless messaging, multimedia, Bluetooth, and web services. Because optional packages are modular, developers can avoid carrying the overhead of unnecessary functionality by including only the packages of an application actually needs.

2.5.3 Mobile Information Device Profile (MIDP)

The Mobile Information Device Profile (MIDP) is a key element of J2ME. CLDC and MIDP provide the core application functionality required by mobile applications, in the form of a standardized Java runtime environment and a rich set of Java APIs. The MIDP specification was also defined through the Java Community Process (JCP) and it defines the J2ME platform for dynamically and securely deploying optimized, graphical, networked applications.

MIDP has been widely adopted as the platform of choice for mobile applications. Companies around the world have already taken advantage of MIDP to write a broad range of consumer and enterprise mobile applications.

2.5.3.1 Specifications

MIDP 2.0 is a revised version of the MIDP 1.0 specification. New features include an enhanced user interface, multimedia and game functionality, more extensive connectivity, over-the-air provisioning (OTA), and end-to-end security. MIDP 2.0 is backward-compatible with MIDP 1.0, and continues to target mobile information devices like mobile phones and PDAs.

MIDP 1.0 is the original specification, which provides core application functionality required by mobile applications, including basic user interface and network security.

2.5.3.2 Benefits

Rich User Interface Capabilities: MIDP applications provide the foundation for highly graphical and intuitive applications.

Extensive Connectivity: MIDP enables developers to exploit the native data network and messaging capabilities of mobile information devices.

Multimedia and Game Functionality: MIDP is ideal for building portable games and multimedia applications.

Over-the-Air-Provisioning: A major benefit of MIDP is its capability to deploy and update applications dynamically and securely, over the air.

End-to-End Security: MIDP provides a robust security model that complies with open standards and protects the network, applications, and mobile information devices.

2.5.4 MIDlets

All applications for the MID Profile must be derived from a special class, MIDlet. The MIDlet class manages the life cycle of the application. It is located in the package *javax. microedition.midlet.*

MIDlets can be compared to J2SE applets, except that their state is more independent from the display state. A MIDlet can exist in four different states: loaded, active, paused, and destroyed. Figure 5 gives an overview of the MIDlet lifecycle. When a MIDlet is loaded into the device and the constructor is called, it is in the loaded state. This can happen at any time before the program manager starts the application by calling the *startApp()* method. After *startApp()* is called, the MIDlet is in the active state until the program manager calls *pauseApp()* or *destroyApp()*; *pauseApp()* pauses the MIDlet, and *desroyApp()* terminates the MIDlet. All state change callback methods should terminate quickly, because the state is not changed completely before the method returns.

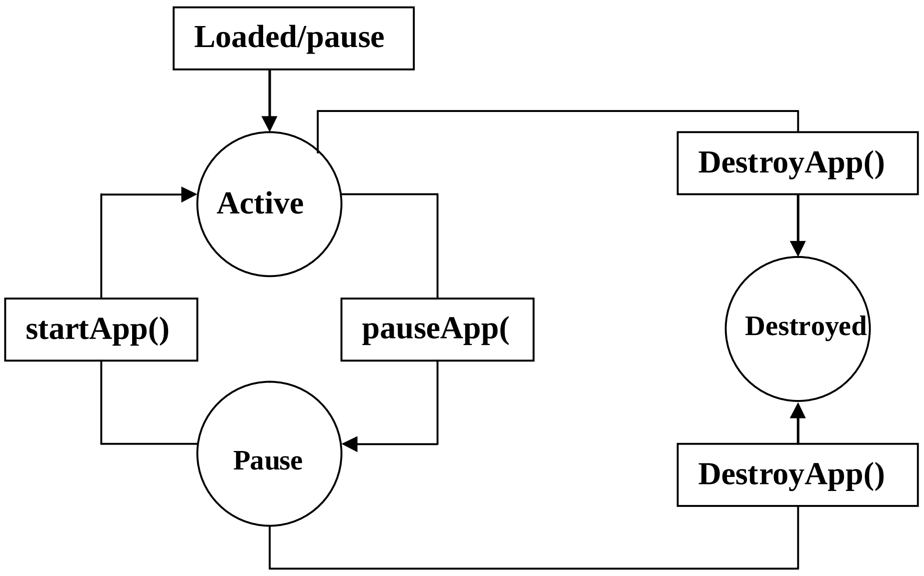


Fig. 5: The life cycle of a MIDlet

2.6 XML Document Parsing

Extensible Markup Language (XML) is a simple, very flexible text format derived from Standard Generalized Markup Language (SGML) (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

More and more enterprise and Java technology projects are making use of XML as a medium to store data in a portable fashion. But due to the increased processing power demanded by XML parsers, J2ME applications have largely been left out of this trend. Now, however, small-footprint XML parsers for the Java language are emerging that will allow MIDP programmers to take advantage of the power of XML.

2.6.1 XML parser overview

An XML processing model describes the steps an application should take to process XML; an application that implements such a model is called an XML parser. You can integrate an XML parser into your Java applications with the Java API for XML Processing (JAXP). JAXP allows applications to parse and transform XML documents using an API that is independent of any particular XML processor implementation. Through a plug-in scheme, developers can change XML processor implementations without altering their applications.

The XML parsing process operates in three phases:

1. XML input processing. In this stage, the application parses and validates the source document; recognizes and searches for relevant information based on its location or its tagging in the source document; extracts the relevant information when it is located; and, optionally, maps and binds the retrieved information to business objects.
2. Business logic handling. This is the stage in which the actual processing of the input information takes place. It might result in the generation of output information.

3. XML output processing. In this stage, the application constructs a model of the document to be generated with the Document Object Model (DOM). It then either applies XSLT style sheets or directly serializes to XML.

2.6.2 XML parser in MIDP

Parsers are traditionally bulky, featuring lots of code and hefty runtime memory requirements. In MIDP devices, the memory available for code is usually small and individual applications may have a maximum code size.
There are three fundamental parser types. Which type you choose depends on how you want your application to behave and what types of documents you're expecting to parse.

1. A *model* parser reads an entire document and creates a representation of the document in memory. Model parsers use significantly more memory than other types of parsers.

2. A *push* parser reads through an entire document. As it encounters various parts of the document, it notifies a listener object.

3. A *pull* parser reads a little bit of a document at once. The application drives the parser through the document by repeatedly requesting the next piece.

Open source parsers are attractive because they give you lots of control. You can customize a parser if you need additional features, and you can fix the parser if it has bugs.

The following table summarizes the current offering of small XML parsers that are appropriate for MIDP.

Name	**License**	**Size**	**MIDP**	**Type**
ASXMLP 020308	Modified BSD	6 kB	yes	push, model
kXML 2.0	EPL	10 kB	yes	pull
kXML 1.2	EPL	16 kB	yes	pull
MinML 1.7	BSD	14 kB	no	push
NanoXML 1.6.4	zlib/libpng	10 kB	patch	model
TinyXML 0.7	GPL	12 kB	no	model
Xparse-J 1.1	GPL	6 kB	yes	model

Tab. 4: The current offering of small XML parsers that are appropriate for MIDP

In this application, kXML 2.0 was used as XML parser.

2.7 Earlier work on cartographic visualization for mobile applications

Here several WMS clients developed by other developers for the mobile devices will be introduced. This application has extracted some good ideas from those work when designs the codes structure and user interfaces.

n J2ME OGC WMS Client 1.10

This client was developed by Skylab mobilesystems[7], implemented in J2ME (MIDP1.0/CLDC1.0), to communicate with OGC conform WMS servers (OGC WMS 1.1.0/1.1.1 specifications) and display the requested maps. It is designed to be run on ultramobile devices like PDAs and mobile phones. It has such features:

- Navigation inside the map (zoom, scroll)
- Free scalable zoom and scroll level
- Unlimited layer selection and presentation
- Server bookmark management
- Manual server input possibility

Figure shows the user Interfaces:

[7] http://www.skylab-mobilesystems.com

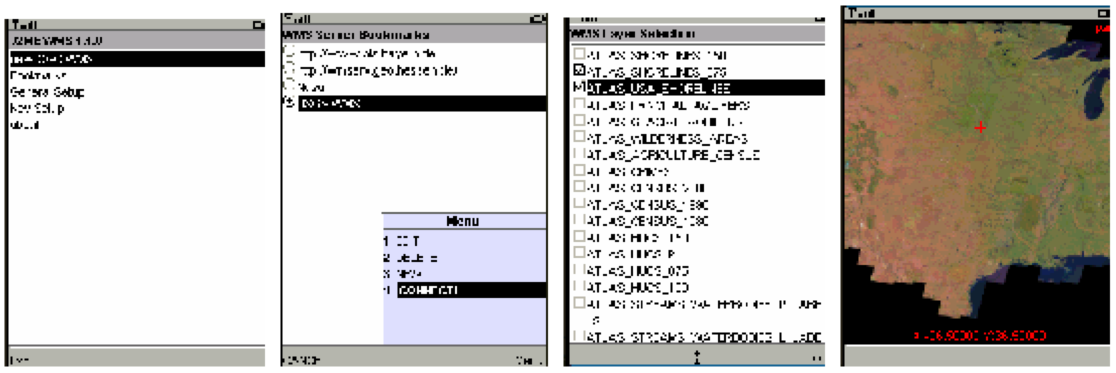


Fig. 6: The user interfaces of J2ME OGC WMS Client 1.10

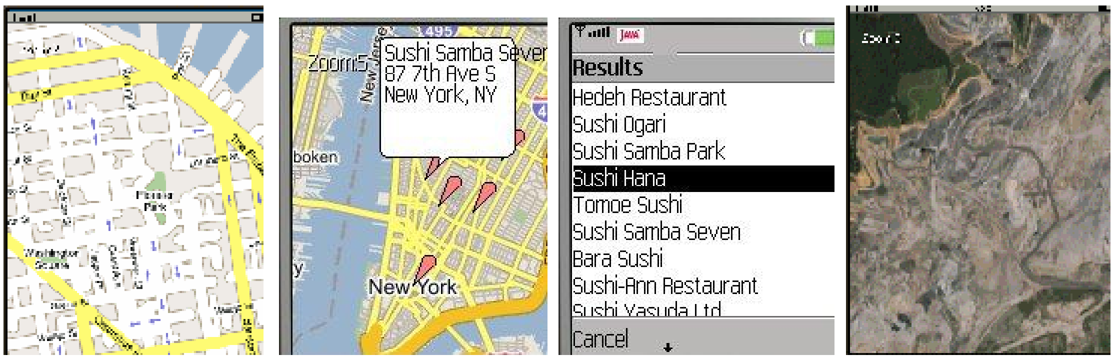
J2ME Map 8.03

This client was developed by TomSoft[8], it's a small interface to GoogleMap that allows you to do the following things:

- Browse the entire GoogleMap database
- Swith from GoogleMap/Satelitte and MSN Virtual Earth Maps and Satellite
- Switch between satellite or map view
- Zoom in/zoom out
- Do request to google maps, and show results on screen
- Save your favorite locations
- Have access to some RSS feed to discover some new locations
- Can be extended with your own data
- Automatic painless saving of your preference...
- Use an embeeded GPS if present, to be automatically located
- Use an external GPS connected with Bluetooth, if present
- Support of GPX file format
- Support of touch screen enabeld handsets

[8] http://j2memap.landspurg.net/

Figure 7 shows the user Interfaces:



Results
Hedeh Restaurant
Sushi Ogari
Sushi Samba Park
Sushi Hana
Tomoe Sushi
Sushi Samba Seven
Bara Sushi
Sushi-Ann Restaurant
Sushi Yasuda Ltd
Cancel

Fig. 7: The user interfaces of J2ME Map 8.03

3 The implementation

3.1 Three testing server environments

For testing the client, WMS server based on vector and raster image formats should be chosen. GetCapabilities and GetMap request could be sent to WMS and got the responding data from WMS.

3.1.1 Apache/PHP/MySQL SVG based map WMS in localhost

- Data selection

  The borderline of Germany

  Several cities

- Software used

  Apache[9] 2.0.54

  PHP[10] 5.0.5.5

  MySQL[11] 4.1

  phpMyAdmin[12] 2.39.2.2

  PHPMyWMS[13]

.
An PHP based compliant web map server is set up in my laptop as localhost server. Apache, PHP, MySQL, phpMyAdmin and PHPMyWMS are the server side software.

Apache was used as web server, MySQL was used as database and phpMyAdmin is used to manager the database. PHPMyWMS was developed by Prof. Behr and Mr. Filmon, is an open source based WMS compliant web map server, using PHP technology.

9 http://www.apache.org
10 http://www.php.net
11 http://www.mysql.com
12 http:// www.phpmyadmin.net
13 http://www.gis-news.de/wms/

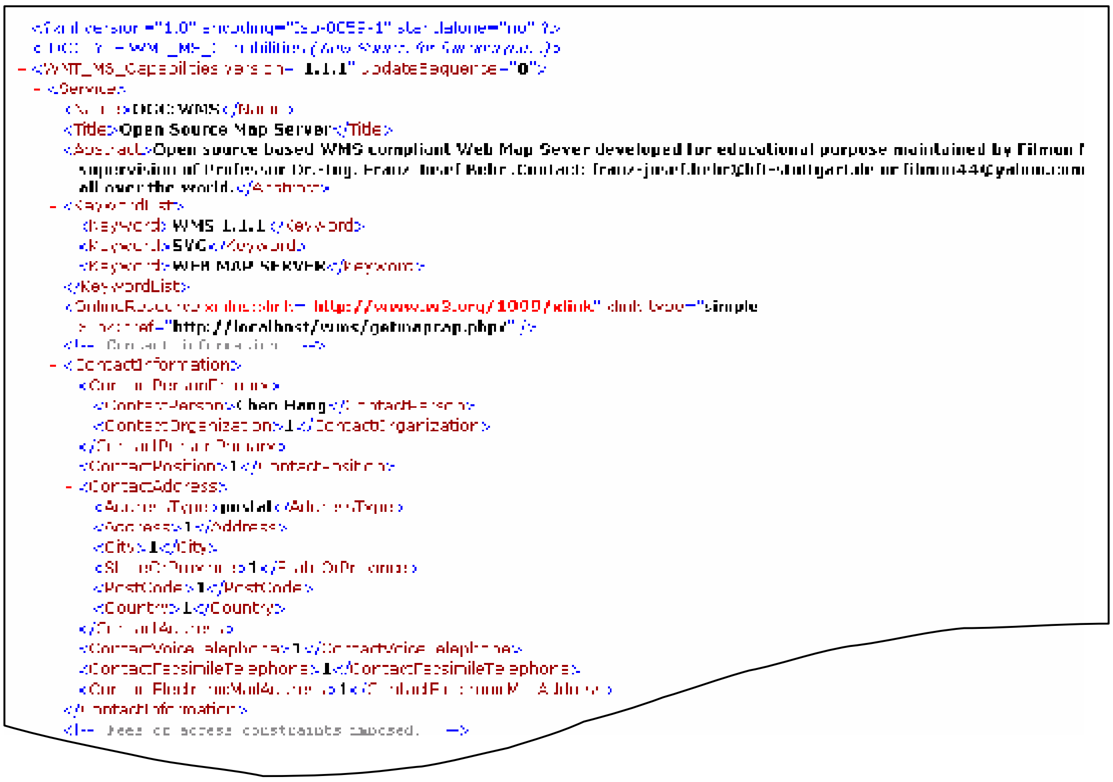
In this WMS server, the "getmapcap.php" is the main entrance process page. Anytime when the user sends the requests, this page will be called firstly and the parameters inside the URL will be analyzed. The corresponding page or function will be executed.

The "GetCapabilities" request can be called by using the URL

```
http://localhost/phpmywms/getmapcap.php?VERSION=1.1.1&SERVICE=WMS&REQUEST=GetCapabilities
```

Listing 5: The GetCapabilities request in a URL

The result XML file can be seen in Figure 8:



```
  <?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
  <!DOCTYPE WMT_MS_Capabilities (View Source for full doctype...)>
- <WMT_MS_Capabilities version="1.1.1" updateSequence="0">
  - <Service>
      <Name>OGC:WMS</Name>
      <Title>Open Source Map Server</Title>
      <Abstract>Open source based WMS compliant Web Map Sever developed for educational purpose maintained by Filmon F
        supervision of Professor Dr.-Ing. Franz Josef Behr. Contact: franz-josef.behr@hft-stuttgart.de or filmon44@yahoo.com
        all over the world.</Abstract>
    - <KeywordList>
        <Keyword>WMS 1.1.1</Keyword>
        <Keyword>SVG</Keyword>
        <Keyword>WEB MAP SERVER</Keyword>
      </KeywordList>
      <OnlineResource xmlns:xlink="http://www.w3.org/1999/xlink" xlink:type="simple"
        xlink:href="http://localhost/wms/getmapcap.php?" />
      <!-- Contact information -->
    - <ContactInformation>
        <ContactPersonPrimary>
          <ContactPerson>Chen Hang</ContactPerson>
          <ContactOrganization>1</ContactOrganization>
        </ContactPersonPrimary>
        <ContactPosition>1</ContactPosition>
      - <ContactAddress>
          <AddressType>postal</AddressType>
          <Address>1</Address>
          <City>1</City>
          <StateOrProvince>1</StateOrProvince>
          <PostCode>1</PostCode>
          <Country>1</Country>
        </ContactAddress>
        <ContactVoiceTelephone>1</ContactVoiceTelephone>
        <ContactFacsimileTelephone>1</ContactFacsimileTelephone>
        <ContactElectronicMailAddress>1</ContactElectronicMailAddress>
      </ContactInformation>
      <!-- Fees or access constraints imposed. -->
```

Fig. 8: XML file fragment of GetCapabilites request from localhost WebMap server in GIS lab

The GetMap request can be called by using URL

```
http://localhost/phpmywms/getmapcap.php?VERSION=1.1.1&BBOX=3538000.0,5444000.0,3592000.0,5556000.0&LAYERS=city&Bundeslaender&STYLES=%2C%2C&REQUEST=GetMap&layer2=city&layer1=Bundeslaender&style=&SRS=EPSG:31467&WIDTH=800&HEIGHT=600&FORMAT=image/svg+xml&EXCEPTIONS=application/vnd.ogc.se_xml&transparent=TRUE&button=GetMapRequest
```

Listing 6: The GetMap request in a URL

The result is shown as below (see Figure 9):

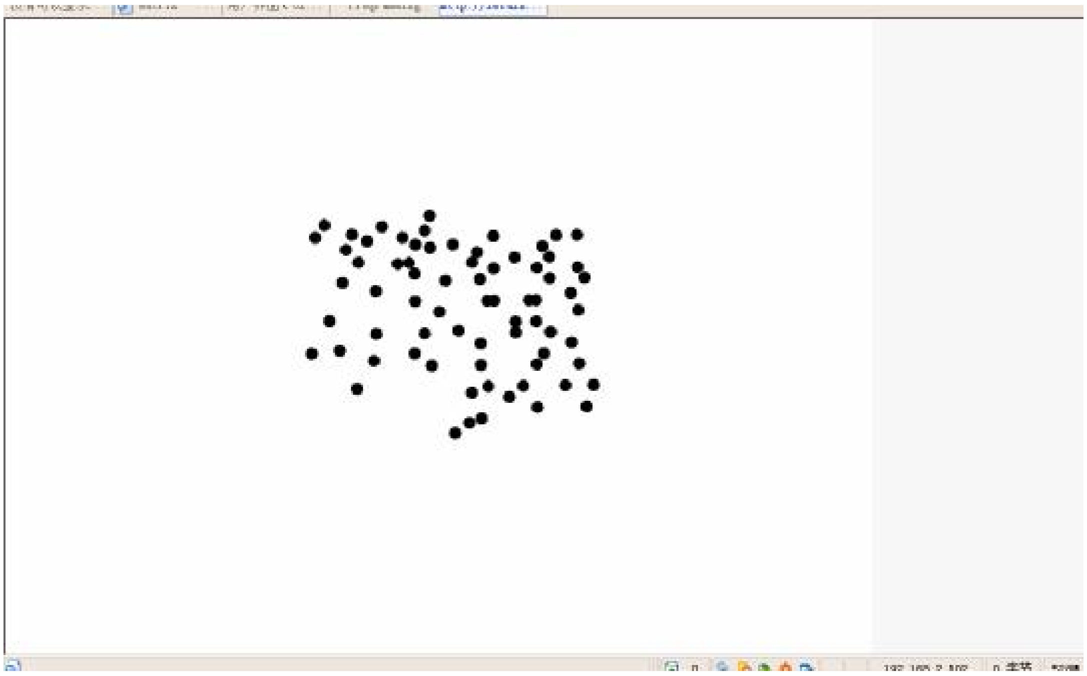


Fig. 9: GetMap request result and URL of the http://localhost/phpmywms web map server

The description how to install and configure this localhost SVG map server can be found in Appendix B.

3.1.2 Apache/PHP/MySQL SVG based map WMS from gis-news

- Data selection
  The data from all over the world including city, airport, river etc.

- Software used
  The same as what were used in localhost WMS

The "GeMap" request can be called by using the URL

```
http://www.gis-news.de/wms/getmapcap.php?VERSION=1.1.1&BBOX=189775.33,4816305.37,
761662.27,5472414.18&LAYERS=airports,ctybdpy2,lakespy2&STYLES=,,&REQUEST=GetM
ap&layer1=airports&layer2=ctybdpy2&layer3=lakespy2&style=&SRS=EPSG:26715&minx=18
9775.33&miny=4816305.37&maxx=761662.27&maxy=5472414.18&WIDTH=700&HEIGHT=
700&FORMAT=image/svgxml&EXCEPTIONS=application/vnd.ogc.se_xml&transparent=True
&button=GetMapRequest
```

Listing 7: The GetMap request in a URL

The result is shown as below (see Figure 7):

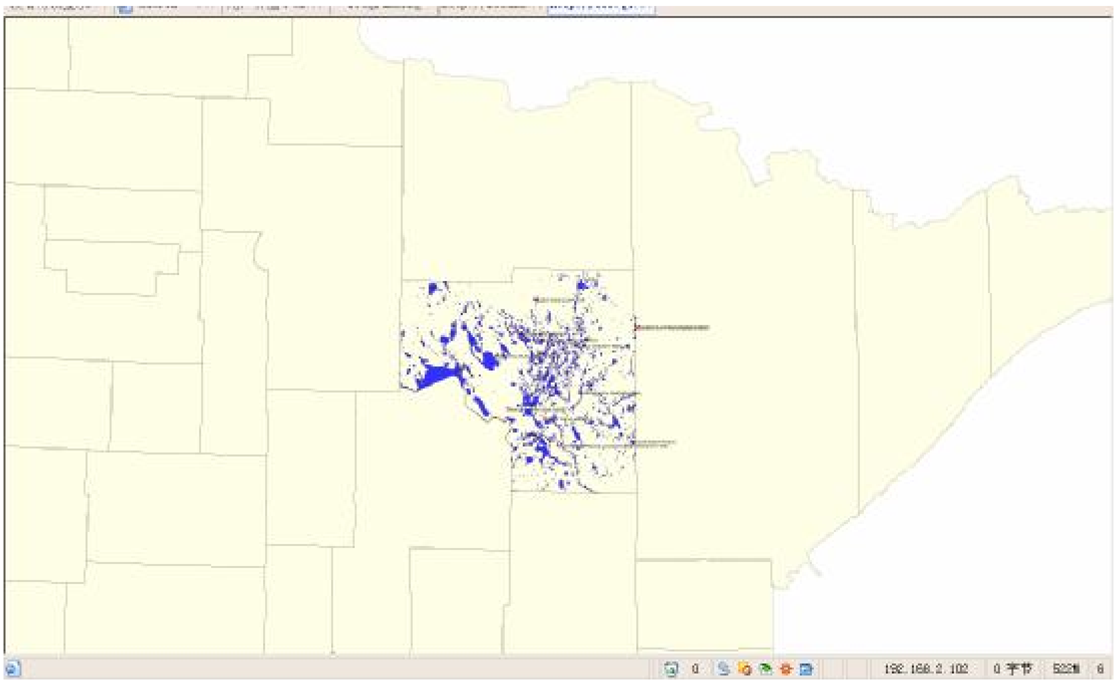


Fig. 10: GetMap request result and URL of www.gis-news.de web map server

3.1.3 Raster image based map WMS from NASA

It is a well-known web map server. Map in raster format is dynamically generated when the GetMap page is called.

The URL of the web map server is:

```
http://onearth.jpl.nasa.gov/browse.cgi?wms_server=wms.cgi&layers=global_mosaic&srs=EPSG:4
326&width=1000&height=500&bbox=-180,-90,180,90&format=image/jpeg&styles=&zoom=
```

Listing 8: The GetMap request in a URL

Logical zoom is available in this server. Figure 8 show the application.

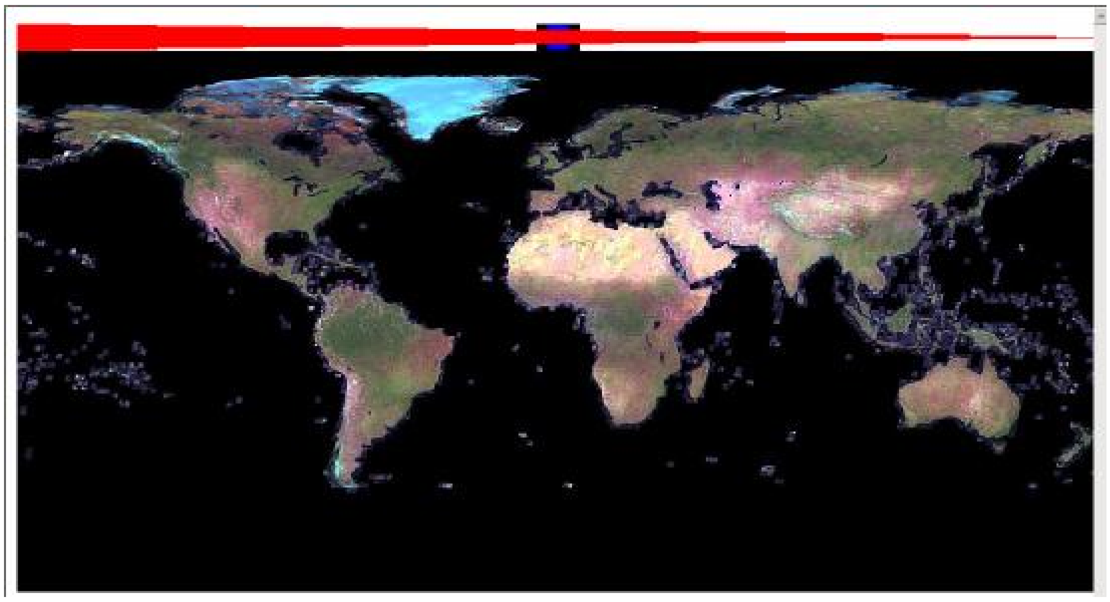


Fig. 11: The raster map in a small scale from NASA

3.2 Coordinates transformation

3.2.1 Dot pitch

The dot specified by the dot pitch is the smallest physical visual component on the display. Also called pixel scale, a measurement that indicates the diagonal distance between like-colored phosphor dots on a display screen (Figure 12). Measured in millimeters, the dot pitch is one of the principal characteristics that determine the quality of display monitors. The lower the number, the crisper the image. The dot pitch of color monitors for personal computers ranges from about 0.15 mm to 0.30 mm.

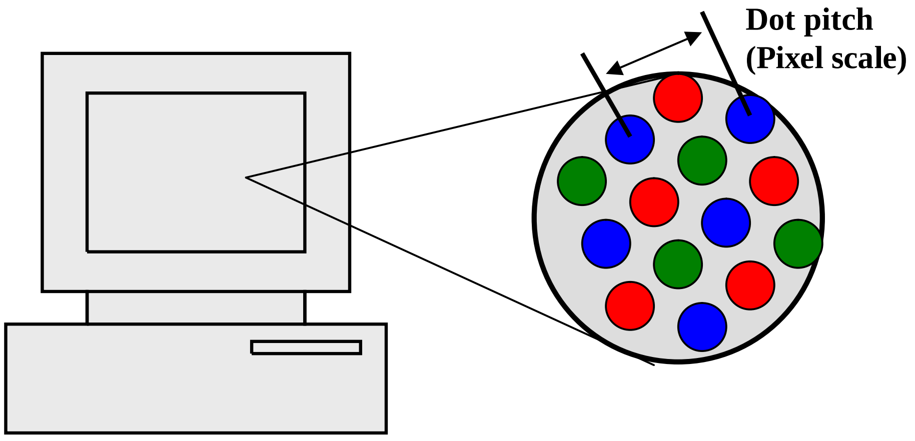


Fig. 12: Zoomed view of screen pixels

For this research, the display size of a simulator i.e. 240 x 320 pixels on 3.94 inches' screen is considered. One inch equal to 25.4 mm, which means the pixel scale is 0.25 mm. So the real width and height of the screen i.e. 6 x 8 cms. This is the formulary how to calculate the pixel scale.

1 inch = 25.4 mm

Screen.width(mm)^2+Screen.height(mm)^2 =(Screen.size(inch) x 25.4(mm))^2

Dot pitch(Pixel scale) = Screen.width(mm)/ Screen.width(px) or Screen.height(mm)/ Screen.height(px)

Listing 9: Formulary to calculate the pixel scale

3.2.2 View Box

In the SVG map, the viewing coordinates information is described by the view box coordinates and four transformation parameters which are defined in the *svg* section (see List 10).

```
<svg   width="240px" height="320px" viewBox="427581 -5260617 24000 32000"
preserveAspectRatio="xMidYMid meet">

viewBox = "leftupX -rightdownY viewBox.width viewbox.height"

viewBox.width = rightdownX-leftupX
viewbox.height = rightdownY-leftupY

map scale = Screen.width(mm)/ viewBox.width(m)
```

Listing 10: Formulary to calculate the map scale

The principle of this transformation is the coordinates system transformation from the view box coordinates system to a geocoded coordinates system. It is a 2-dimensional similarity transformation (2D Helmert). There are four transformation parameters in this transformation. X shift, Y shift, scale factor and one rotation parameter (see Figure 13).

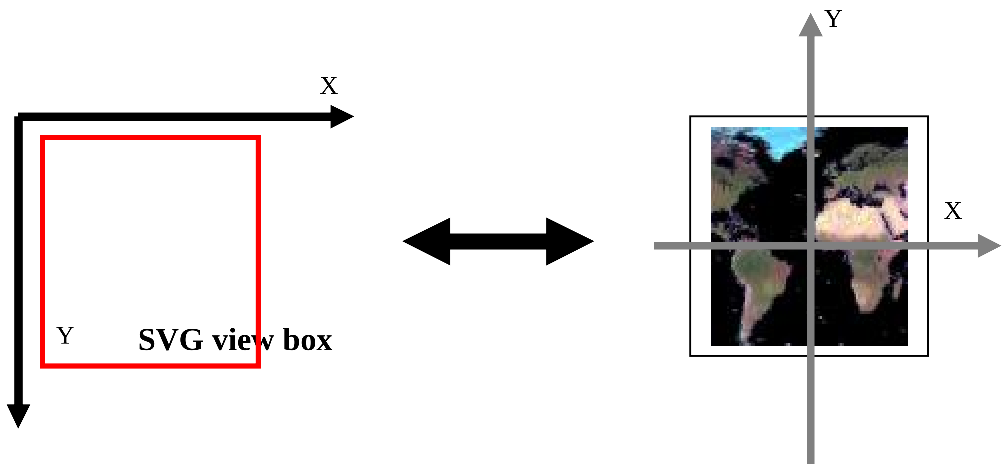


Fig. 13: Coordinates transformed from SVG view box coordinate system to a geocoded coordinate system.

Figure 14 and 15 show example SVG documents with same viewBox but with different width or height.

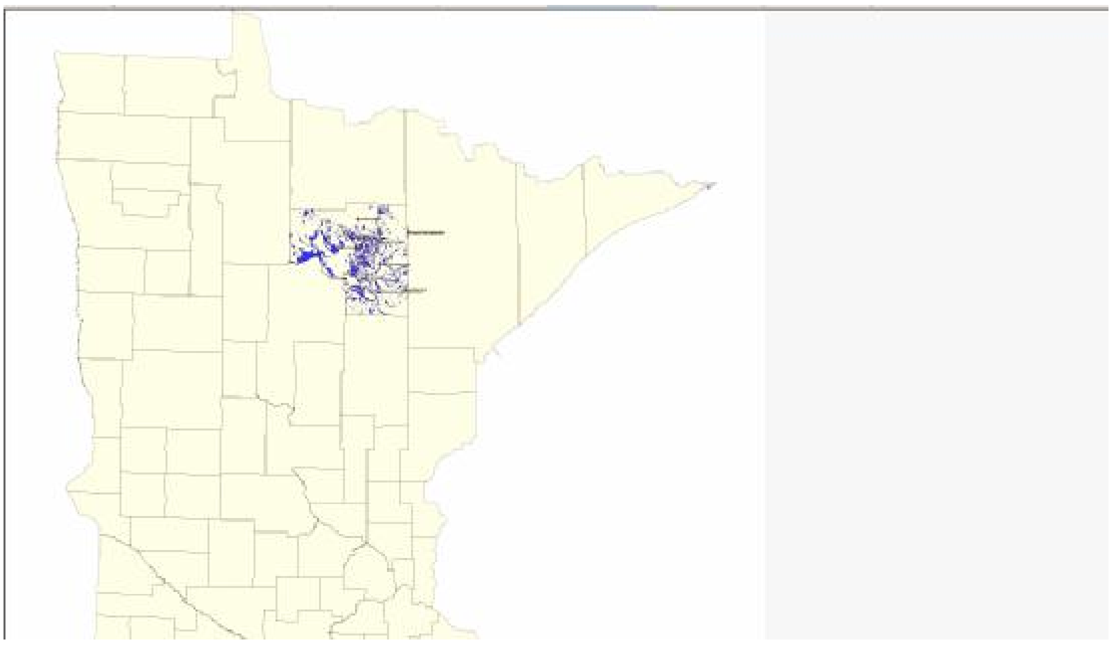


Fig. 14: SVG document with 800 (width) x 600 (height) pixels and viewBox with 571887 (width) x 656109 (height) m.

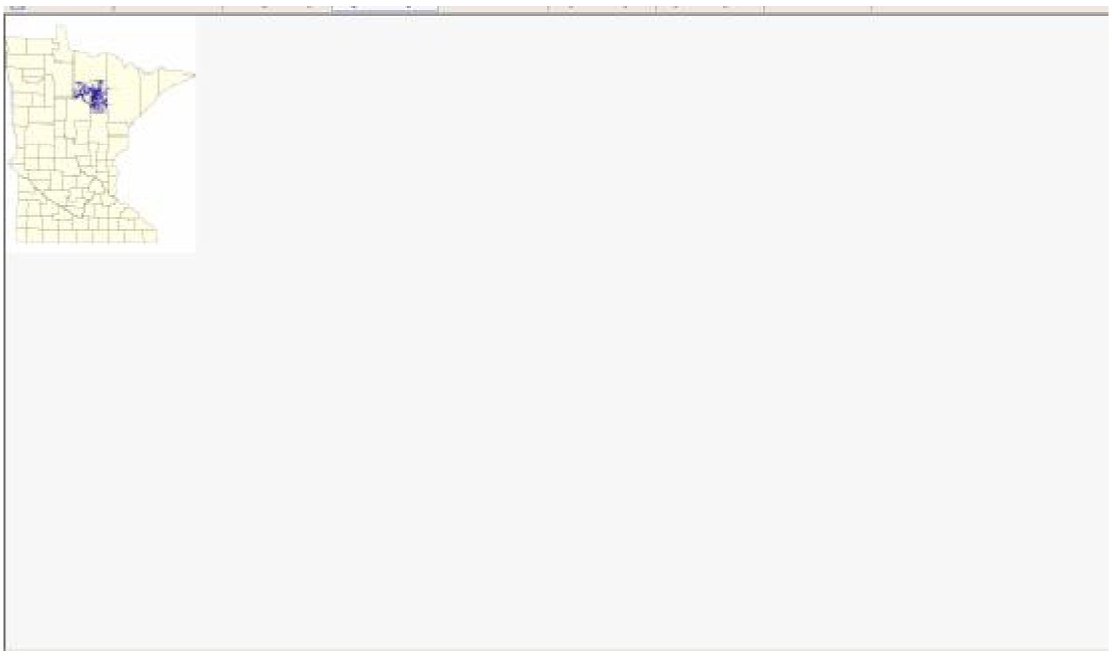


Fig. 15: SVG document with 240 (width) x 320 (height) pixels and viewBox with 571887 (width) x 656109 (height) m.

But, only reduce the displaying size of SVG map is not enough. The client should select only the displaying area where the users want to view, not necessary to load the whole map file.

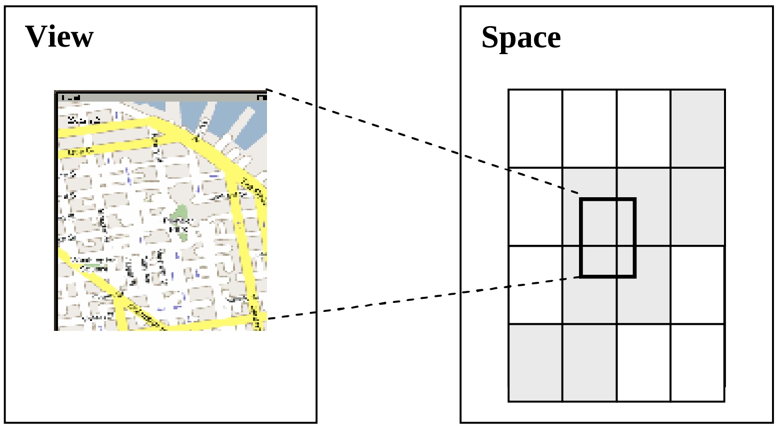


Fig. 16: Display area selection

Supposed that the user has input the coordinate of one point: pX and pY, the client should calculate the selected area around this point according to the map scale, using the same formulary (List. 8) to decide the view box location in the full area of map. After those transformation and calculation, the size of map i.e. width and height of SVG document were fixed as 240 and 320 (pixels) respectively, and the width and height of viewBox were also set as 24000m x 32000m with map scale 1: 400000. See Figure 17.

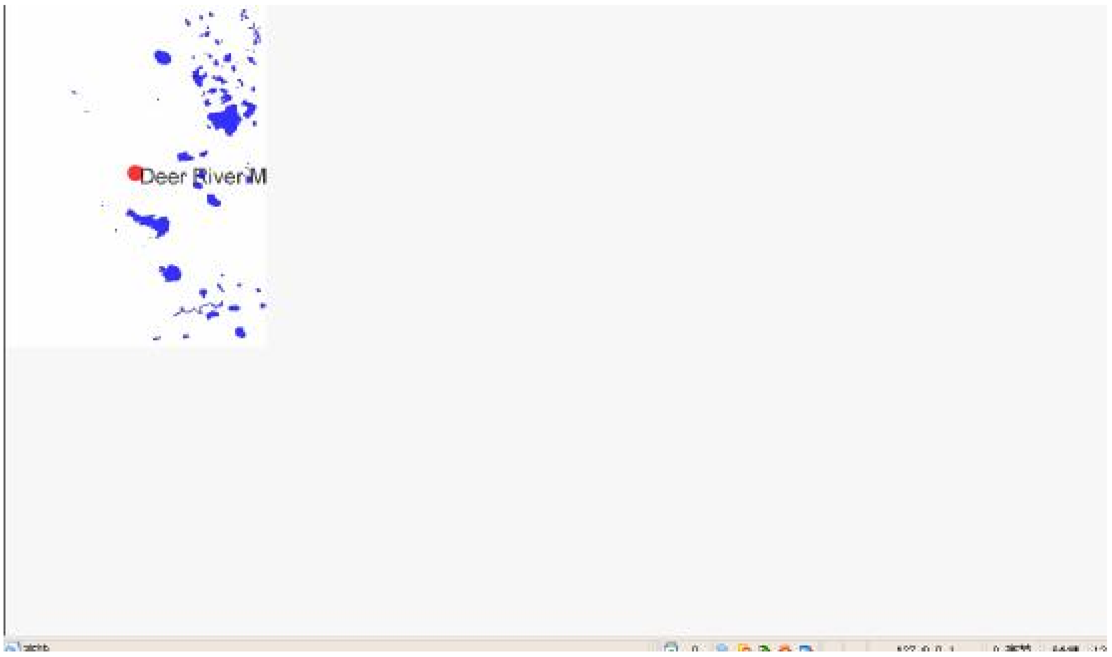


Fig. 17: SVG document with 240 (width) x 320 (height) pixels and viewBox with 24000 (width) x 32000 (height) m.

This size of result SVG file in Figure 17 is only 40.4 KB, comparing to the full size SVG file (1.76 MB) in Figure 15, undoubtedly such transformation is of important to solve the limited connection bandwidth problem.

3.3 Design Steps

3.3.1 Java programming&debugging environment

For designing this client, it is unnecessary and inconvenient to test every steps of debugging on the mobile phone, so we need a virtual platform, in which we can easily simulate the real mobile device while testing the client's capabilities and environment.

After all, we need set the java programming and debugging environment in window system. Install JDK (Java Development Kit) and set the windows environment variant Detailed installation steps refer to Appendix C.

3.3.2 Simulating mobile device

J2ME Wireless Toolkit, composes of one KVM (K Virtual Machine) and several API. KVM needs only 40-80KB memory, 20-40KB heap, and can be run on the 16bit 25MHz processor. For the further well debugging and testing on the real mobile phone, we need the other Toolkit from mobile phone companies according to the type and mark of mobile phone you want to test.

Here are the interface of WTK (Wireless Toolkit) and Series 60 Platform.

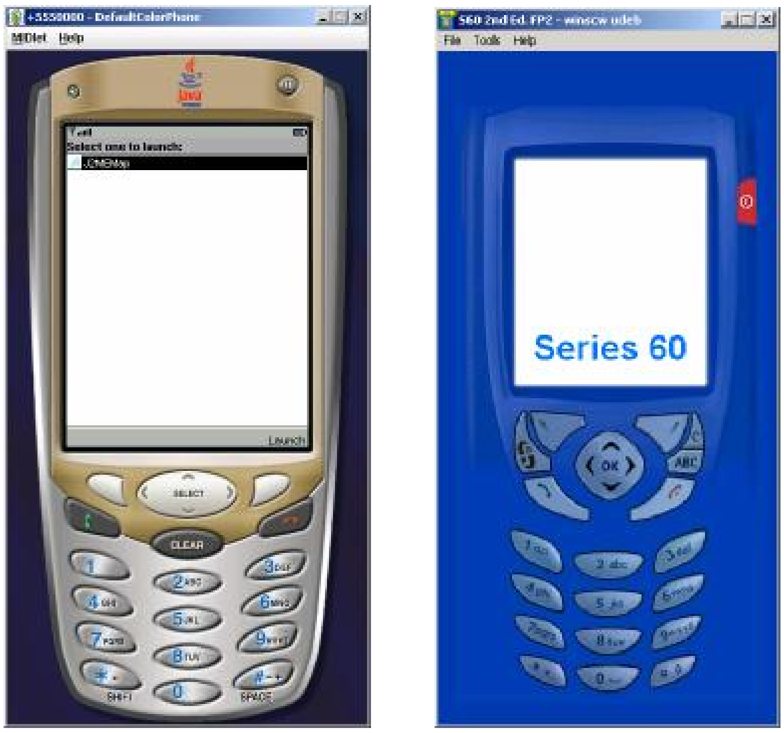


Fig. 18: Interfaces of WTK (Wireless Toolkit) and Series 60 Platform

Detailed installation steps refer to Appendix D.

3.3.3 Programming with Eclipse/eclipseme plug-in & Netbeans

In this research Eclipse and Nebeans are chosen as my Java debug environment, because they have friend interface and many debugging assistants, for example, you can check every variant during the bugging to know whether the result is correct. It has many plug-ins and can works with simulators well. The most important is that they are both free to download and use.

Detailed installation steps refer to Appendix D.

3.3.4 System Structure analyzing and designing

The structure of the code is shown in Figure 19 and Figure 20. From the user's point of view, the users never care about the structure of the code except the user interface. But for programmers, they have to take care of both aspects.

From the users' point of view the code structure is only dealing with the interface which is visible to them such as a pull-down menu or a from with choice box, text box etc. As shown in Figure 19, all the elements are the client interfaces. Firstly the users have the possibility to start the client in their mobile phone, and select the menu pull-down menu list to operate, and the corresponding form appears afterwards. If the user does something wrong or the client meets some unexpected errors, for example, users input invalidate coordinate, the process will stop and jump one error alert form; if all the inputted parameters are valid, the SVGT viewer or raster image viewer will be displayed according to which image format the WMS supports.

During the data processing or responding from WMS, the users will see one waiting splash, and they can stop the process as they will or if the waiting time last too long time because of the bad Internet bandwidth. For map navigation form which the users could use other buttons to operate, for example, click Arrow buttons to move the map, the help form about how to operate should be include in these Forms.

The code structure from the users' point of view.

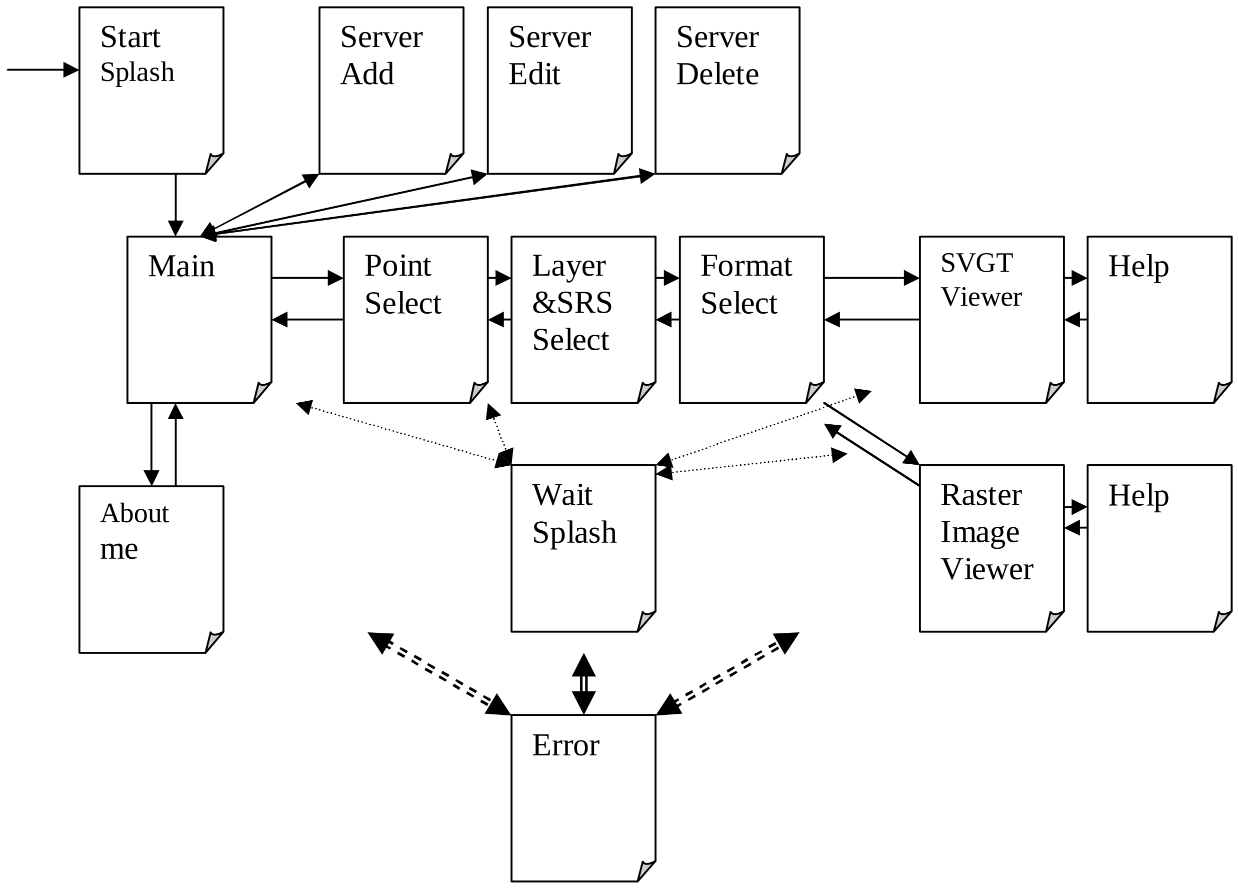


Fig. 19: Interface structure of the client from the users' point of view

In the developers' shoes, the structure is composed of not only the interface, but also several individual functions which will run invisibly (see Figure 20). Java has one famous wisdom,” Write Once，Run Anywhere”, but it is also called by somebody,” Write Once, Debug Anywhere”. That means the more transplantable, the more extendable, the more compatible, the better the codes are. In order to make the code more understandable by other programmers, even by the author himself, the structure of the code should be clear and easy to modify later. So Oriented Object Programming (OOP) method should be used here.

Simply to say, OOP is a type of programming in which programmers define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects. One simple example is given as following, in this example, the server URL as variant was enveloped and transferred.

```
    /** Set server's URL */
  public void setServerURL(Stringsurl){
          this.serverurl=url;
     }

     /** Return server's URL */
     public String getServerURL(){
          return serverurl;
     }
```

Listing 11: Example of codes written in OOP method

But the more OOP methods the developers use, the more classes are created, that means the more storage and memory will be occupied and used. Because of some limitation as mentioned before, J2ME enters in a dilemma embarrassedly. The developer should decide when OOP could or had better to be used, depending on whether the variant declaration should be enveloped, in order to prevent the data overflowing and other aspects.

In this application, the core process has four,

1. Records the data what the users create when they edit, add or delete the server list.

2. Parse the contents of XML file which got from GetCapabilities and GetMap functions from WMS and transfer them to the other class.

3. Validate the result file from WMS after getMapRequest, justify the image format to choose which viewer will be loaded

4. Load the SVGT or PNG image from WMS and display on screen.

In the code structure, after users choose the operation of server list to edit, add, delete server, the "RMS" class was called to store the record storage which has been changed. Once the users select one server and ready to connect the server, "Validate the server, get the WMS Head" class was called to validate the selected server and get the HEAD and Content Type from WMS.

After inputting the XY coordinate, GetCapabilities request was send, and "Parse the XML file from WMS" class was called to parse the XML document from WMS, pick up the request parameters from the content of XML document which are needed to be added in the GetMapRequest URL.

Once users have set all the needed values of parameters and ready to send the GetMapRequest, "Send GetMapRequest, validate the result" and "Justify the image format" classes were called to validate the result file which was got from WMS, if it is XML Document, these are two possibilities, the first is SVGT file, then go to load the SVGT Viewer; the second is XML document that contains the errors description from WMS, when the value of parameters was set in invalidate form before. And if it is image document, then justifies whether it is PNG format image and loades the raster image viewer, otherwise throws one error dialog.

If any processes mentioned before meet any unexpected error, the error dialog class was called, and users can jump back to the former interface they have stayed just now.

The code structure from the developers' point of view:

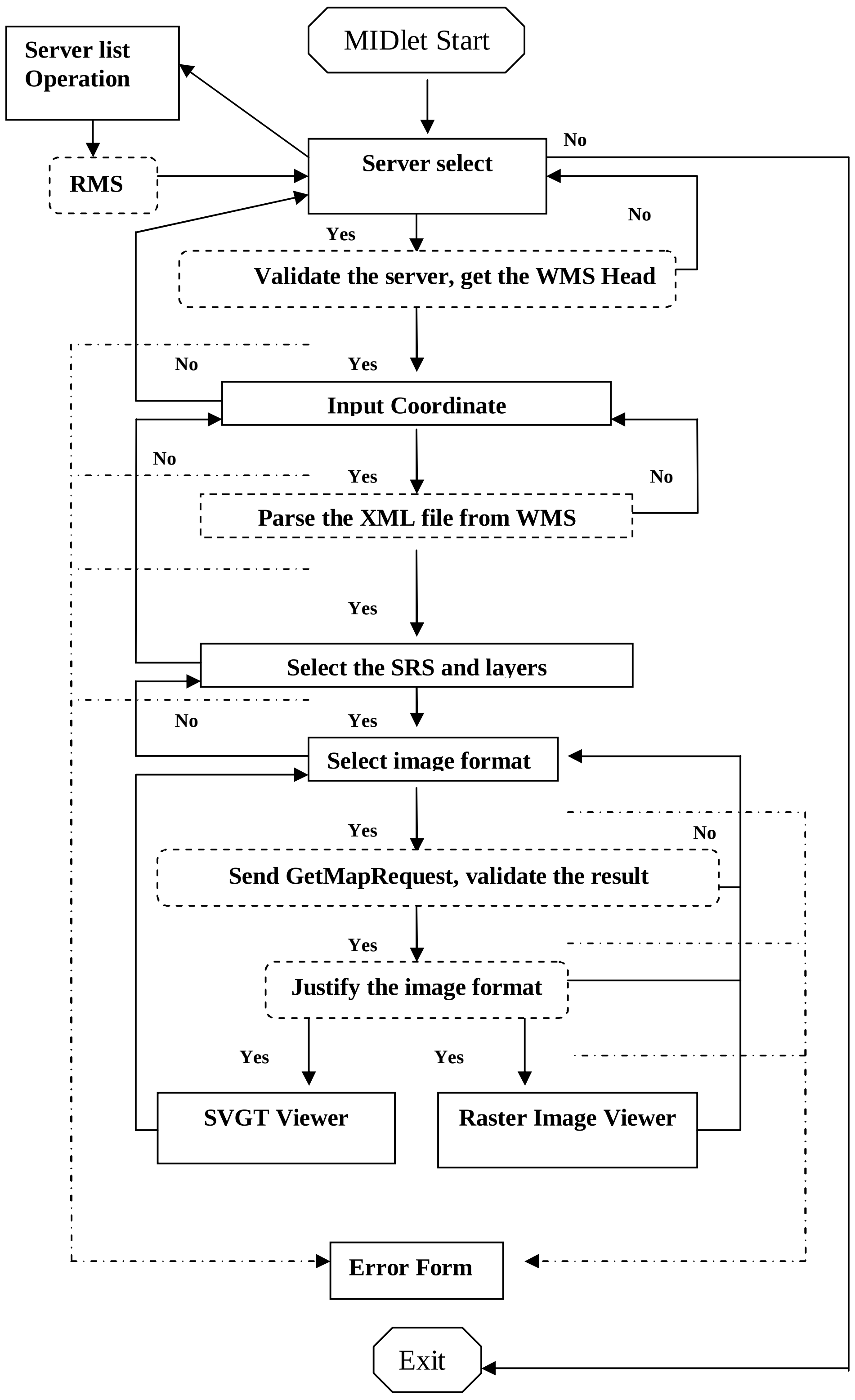


Fig. 20: Code structure of the client from the developers' point of view

4.4 J2ME Record Management Store (RMS)

The Mobile Information Device Profile -- the platform for mobile Java applications -- provides a mechanism for MIDP applications to persistently store data across multiple invocations. This persistent storage mechanism can be viewed as a simple record-oriented database model and is called the record management system (RMS), through which MIDlets can persistently store data and retrieve it later. In a record-oriented approach, J2ME RMS comprises multiple record stores. An overview of J2ME RMS and MIDlet interfacing is given in Figure 21.

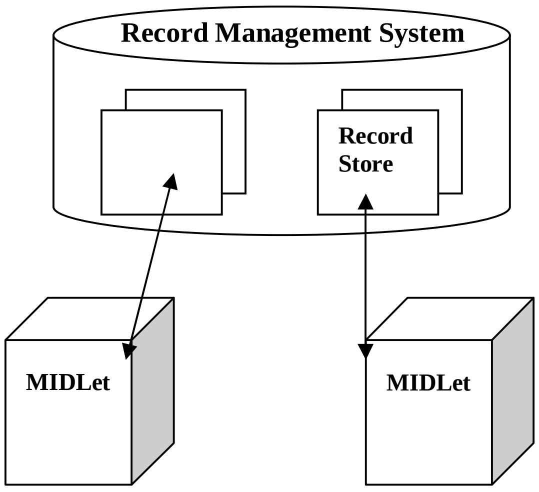


Fig. 21: Overview of J2ME RMS and MIDlet interfacing

Each record store can be visualized as a collection of records, which will remain persistent across multiple invocations of the MIDlet. A record store is created in platform-dependent locations, like nonvolatile device memory, which are not directly exposed to the MIDlets.

3.4.1 Managing the device database

MDIP provides the set of classes and Interfaces in the package *javax.micoredition.rms* to work with RecordStore and this set of API's are called as RMS. The *javax.microedition.rms.RecordStore* class represents a RMS record store. It provides several methods to manage as well as insert, update, and delete records in a record store.

The below statement will create a RecordStore with the name "databasename", if it is already not present otherwise this method returns a reference to the same record store object.

```
RecordStore rs = RecordStore.openRecordStore("databasename",true);
```

Listing 12: Create or open a RecordStore

After performing all operations, a call to the given below method closes the RecordStore with the given name.

```
Rs.closeRecordStore();
```

Listing 13: Close a RecordStore

Below given method will delete the named RecordStore.

```
RecordStore.deleteRecordStore("databasename ");
```

Listing 14: Delete a RecordStore

3.4.2 Data records operation

The given below statement will insert an array of bytes in RecordStore and will return the recordID of this inserted record after writing it to persistent storage.

```
String appt = "new record";
byte bytes[] = appt.getBytes();
rs.addRecord(bytes,0,bytes.length);
```

Listing 15: Insert a Record

RecordID is a unique identifier used by RMS for identifying the records in a particular RecordStore, so the recordID of the record has to be passed to be updated and updates new bytes of array using the given below method

```
String newappt = "update record";
Byte data = newappt.getBytes();
Rs.setRecord(RecordID, data, 0, data.length());
```

Listing 16: Update a Record

To read a specified record from RecordStore make a call to *getRecord* () method in the given below way.

```
byte [] readRecord = Rs.deleteRecord(RecordID);
```

Listing 17: Read a RecordStore

This given method call will delete the record with a specified recordID.

```
Rs.deleteRecord(RecordID);
```

Listing 18: Delete a RecordStore

In this application, RMS was used to restore the server list, including WMS name and URL. And in the further developing, RMS can also store the image as byte array, to reduce the repetition times of requesting to the same WMS map source. If the users access the same source again later, the byte array was read from RMS to rebuild the image.

3.5 KXML XML Parser

The XML parser used in this application is kXML 2.0. In order to make kXML classes available to the application, kxml2-src.zip package from the kXML site need to be downloaded. Then, copy the contents of the package to the "src" folder in the development system.

As described before, the GetCapabilities XML document should be parsed.

You can see one section of this XML document in Listing 19.

```
<Service>
<Name>OGC:WMS</Name>
<Title>Open Source Map Server</Title>
<Abstract>Open source based WMS compliant Web Map Sever developed
  for educational purpose maintained by Filmon Mehari under supervision
  of Professor Dr.-Ing. Franz Josef Behr .Contact:
  franz-josef.behr@hft-stuttgart.de or filmon44@yahoo.com.SVG data
  from all over the world.
 </Abstract>
<KeywordList>
<Keyword>WMS 1.1.1</Keyword>
  ………..
  </Service>
```

Listing 19: One section of the GetCapabilities XML document

In order to capture data from WMS, the application should open a URL connection and get GetCapabilities XML data on an InputStream. The InputStream is made available to the XMLParser through an InputStreamReader. This is illustrated in Listing 20.

```
HttpConnection hc = (HttpConnection)Connector.open(url);
InputStream is = hc.openInputStream();
Reader reader = new InputStreamReader(is);
KXmlParser parser = new KXmlParser();
parser.setInput( reader);
```

Listing 20: Get the GetCapabilities XML data from WMS

When the XMLParser's *read()* method encounters an item, such as the event name (tag name), and read event text (that is, the text enclosed between the start and end tags, also called content). In Listing 21, the parser finds the tag name of title of WMS, then reads the content further and extractes the text for displaying purposes.

```
if (parser.getName().equals("Service"))
{
String wmstitle = null;
parser.require(XmlPullParser.START_TAG, null, null);
if (name.equals("Title"))
wmstitle = parser.nextText();
parser.require(XmlPullParser.END_TAG, null, name);
}
```

Listing 21: Picking up WMS title from GetCapabilities XML data

The title of WMS will be extracted and be displayed on the screen, see Figure 22.

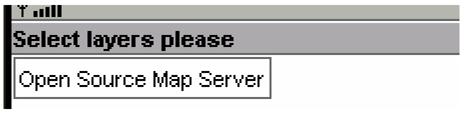

Select layers please
Open Source Map Server

Fig. 22: Display the WMS title

3.6 Tinyline SVGT Toolkit

TinyLine is Java and runs on various J2ME flavors (J2ME MIDP2.0, Nokia Series 60 MIDP, and J2ME Personal Profile). It is actually more than a viewer and offers a full-blown toolkit on which basis a viewer has been developed and made available for free download. TinyLine is currently in version 1.9 and is passing almost all of the SVG Tiny portion of the W3C SVG 1.1 Test Suite (a basis for claiming an SVGT implementation is compliant).

The screen shot is the interface of Tinyline SVGT viewer 1.9.

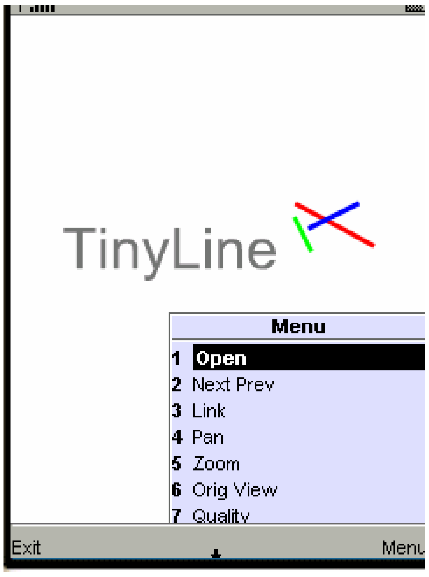


Fig. 23: Interface of Tinyline SVGT Viewer 1.9

TinyLine SVG Toolkit is SVG Tiny applications and SDK, which targets Java developers who want to use SVG Tiny in their applications and needs customization. Download SDK package and copy the contents to the "src" folder in the development system.

Once users get the SVGT file through the GetMap Method, it is easy to load Tinyline SVGT Viewer to view it, with panning and zooming function.

```
// Create the SVG canvas.
canvas = new MIDPSVGCanvas(display);
// Start the event dispatching queue
canvas.start();
//Load the SVGT file from WMS
canvas.goURL(GetMapurl);
```

Listing 22: load and Initialize the Tinyline SVGT Viewer

In this application, only Pan, Zoom, Orig View and Quality and Help functions are preserved, some functions like Link that will never be used has been taken out.

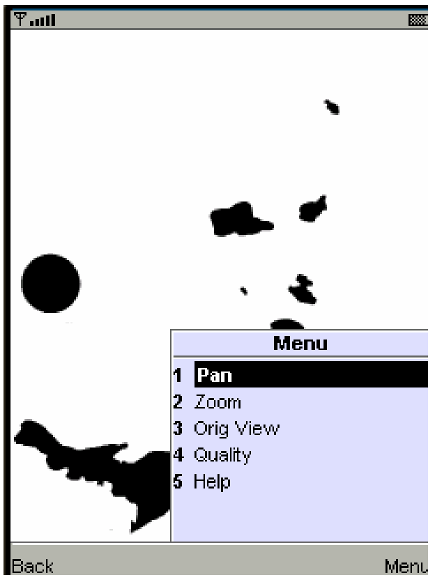


Fig. 24: Interface of Tinyline SVGT Viewer in current application

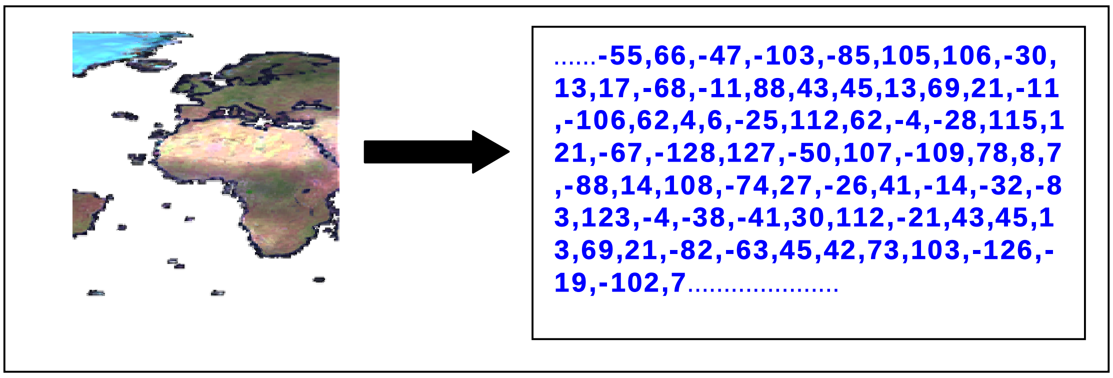
3.7 Raster image viewer development

As described before, PNG is the mostly image format used in mobile device application. In J2ME, the *Graphics* class provides a method for drawing images, static image and non-static image. Static image can not be modified after been created; non-static image is drawn by the J2ME drawing function. Because the limited process speed and memory, the PNG map should be created in form of static image. The static image can be created based on the binary PNG format.

```
Image img=Image.createImage(byte[], int offset, int length);
```

Listing 23: Create static image though binary PNG format.

At first the PNG image data is read though the data stream from WMS, and then puts the data into one byte buffer. At last the Viewer reads those data from the byte buffer and paints the image on the screen according to the data.



```
......-55,66,-47,-103,-85,105,106,-30,
13,17,-68,-11,88,43,45,13,69,21,-11
,-106,62,4,6,-25,112,62,-4,-28,115,1
21,-67,-128,127,-50,107,-109,78,8,7
,-88,14,108,-74,27,-26,41,-14,-32,-8
3,123,-4,-38,-41,30,112,-21,43,45,1
3,69,21,-82,-63,45,42,73,103,-126,-
19,-102,7....................
```

Fig. 25: Read image data as byte array

Similar to SVGT Viewer, the PNG Viewer was developed with such functions, Pan, Zoom, Orig View, and Help. See Figure 26.

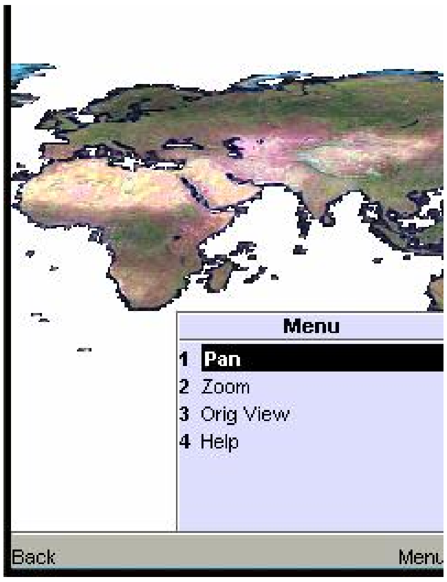


Fig. 26: Interface of raster image Viewer

4 Conclusion

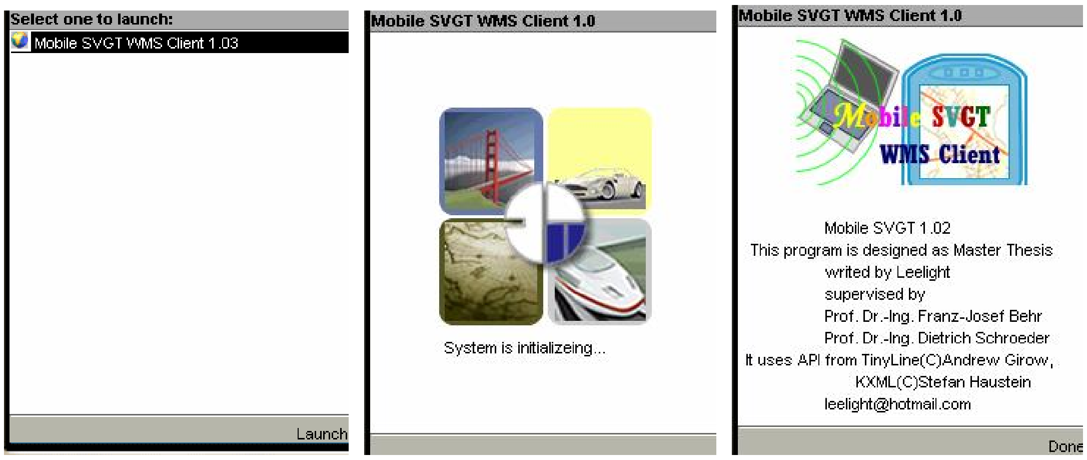
4.1 Usability testing

It is very necessary to evaluate the performance of the client after the development. Other WMS server and different map data will be tested on the client.

The client was tested on one real mobile device, because the codes debugging on the special simulator, for example, S60 platform, is complex and time-consuming, several parts of the codes should be modified to adapt to the special criterion of the mobile devices.

4.1.1 Test with SVG image WMS Server

Load the client, Figure 27 shows the Welcome wait splash and the About description of the client

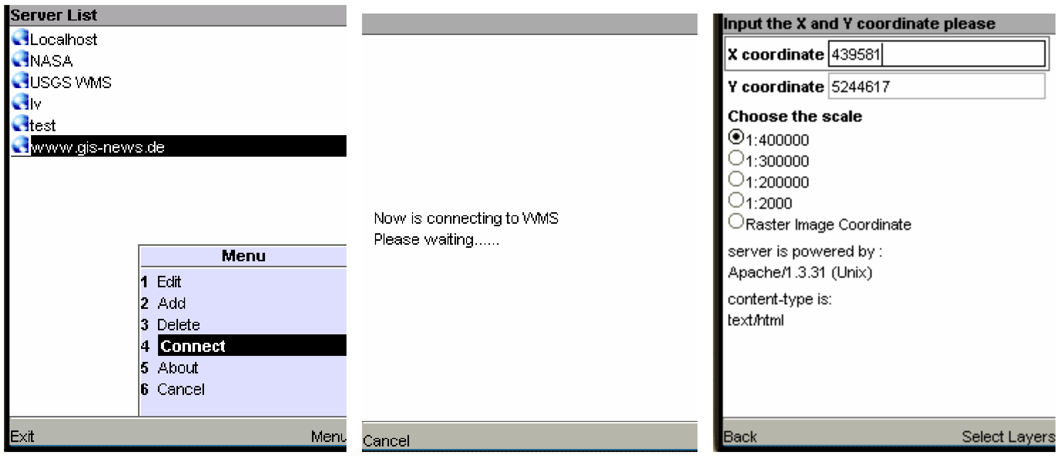

Select one to launch:
Mobile SVGT WMS Client 1.03
Launch
Mobile SVGT WMS Client 1.0
Mobile SVGT WMS Client 1.0
Mobile SVGT
WMS Client
Done

System is initializeing...

Mobile SVGT 1.02
This program is designed as Master Thesis
writed by Leelight
supervised by
Prof. Dr.-Ing. Franz-Josef Behr
Prof. Dr.-Ing. Dietrich Schroeder
It uses API from TinyLine(C)Andrew Girow,
KXML(C)Stefan Haustein
leelight@hotmail.com

Fig. 27: Loads client, Welcome wait splash, About description

First the vector image based WMS was chosen, and connect to the www.gis-news.de, then the wait splash is displaying during the connection. In the *Input the X and Y coordinate please* Form input point coordinate and then connect to WMS again to send GetCapabilities request, and the wait splash displays also during the sending. The information below the Form is server software and content type information of WMS. See Figure 28.


Server List
Localhost
NASA
USGS WMS
lv
test
www.gis-news.de

Menu
1 Edit
2 Add
3 Delete
4 Connect
5 About
6 Cancel

Exit Menu

Now is connecting to WMS
Please waiting......

Cancel

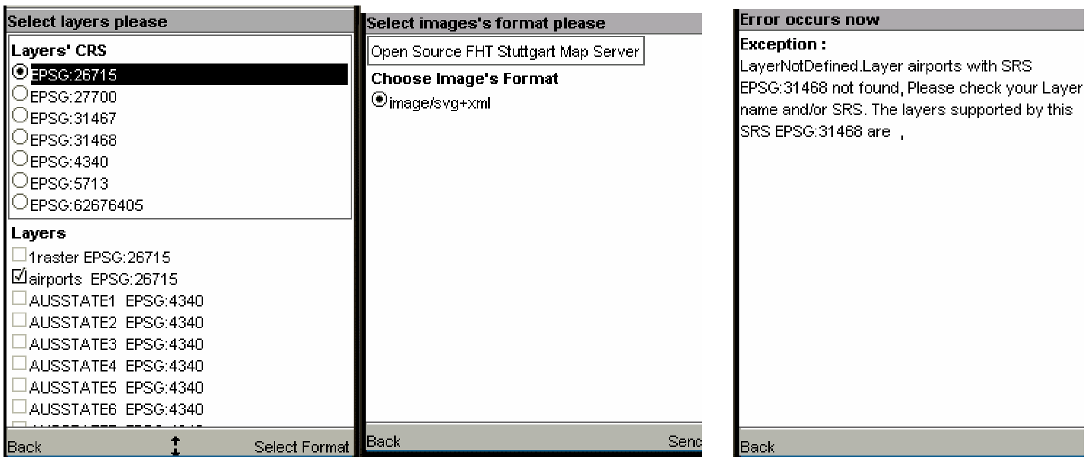
Input the X and Y coordinate please
X coordinate 439581
Y coordinate 5244617
Choose the scale
[✓ 1:400000]
1:300000
1:200000
1:2000
Raster Image Coordinate
server is powered by :
Apache/1.3.31 (Unix)
content-type is:
text/html

Back Select Layers

Fig. 28: Connect WMS, wait splash, input coordinate and WMS information

In Form *Select layers please*, select CRS and layers with the same CRS. Then select the image format in Form *Select image's format please*, and go on to load the SVG map. If parameters were inputted wrongly before, error dialog will be displayed.

For example, if you select CRS EPSG: 31468, but select the layer airports with different CRS EPSG: 26715, the error message will appear as following: LayerNotDefined Layer airports with SRS EPSG: 31468 not found. Please check your layer name and/or SRS. The layers supported by this SRS EPSG: 31468 are. See Figure 29


Select layers please
Layers' CRS
[✓ EPSG:26715]
EPSG:27700
EPSG:31467
EPSG:31468
EPSG:4340
EPSG:5713
EPSG:62676405
Layers
☐ 1raster EPSG:26715
☑ airports EPSG:26715
☐ AUSSTATE1 EPSG:4340
☐ AUSSTATE2 EPSG:4340
☐ AUSSTATE3 EPSG:4340
☐ AUSSTATE4 EPSG:4340
☐ AUSSTATE5 EPSG:4340
☐ AUSSTATE6 EPSG:4340

Back Select Format

Select images's format please
Open Source FHT Stuttgart Map Server
Choose Image's Format
[✓ image/svg+xml]

Back Send

Error occurs now
Exception :
LayerNotDefined.Layer airports with SRS EPSG:31468 not found, Please check your Layer name and/or SRS. The layers supported by this SRS EPSG:31468 are ,

Back

Fig. 29: Select CRS and layers, Select image format, Error dialog

If all the parameters are valid, after the wait splash (maybe takes a little bit long time), the SVG map will be displayed. The users can use *Pan* function to move the map, and *Zoom* function to zoom in or out. See Figure 30.

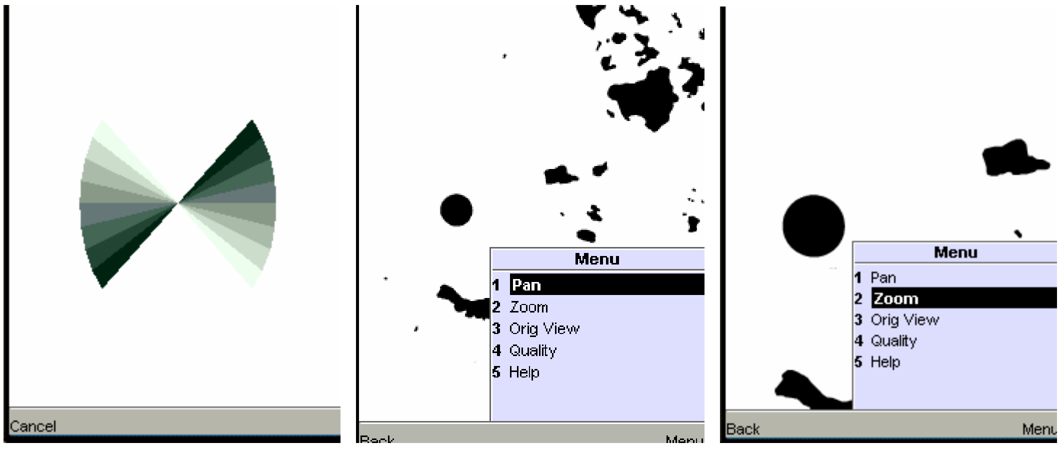
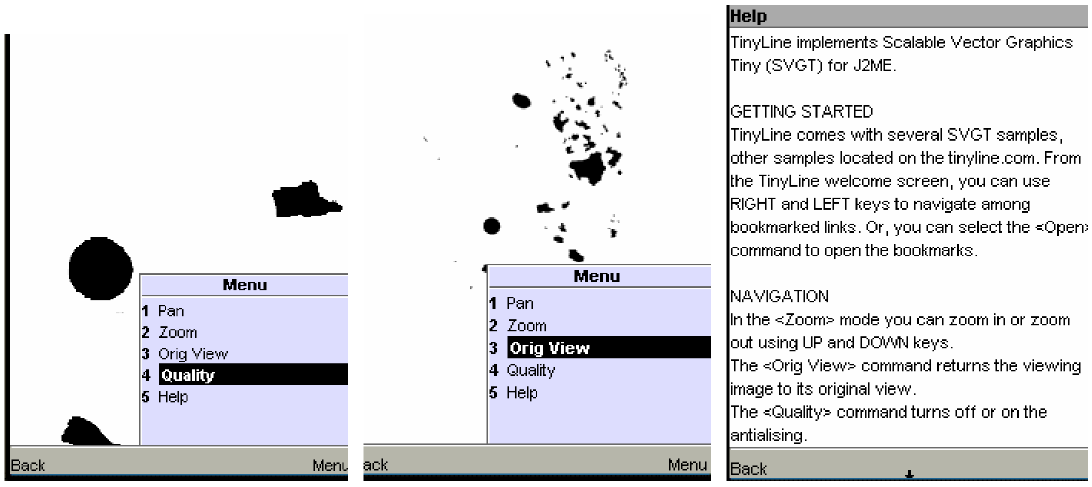


Fig. 30: Wait splash, Move SVG map, Zoom in or out

Function *Orig View* is for retrieveing the original map view and the image quality of SVG map can be reset. If the users are not clear about the operation, Help function can show all the operation description. See Figure 31.



Help

TinyLine implements Scalable Vector Graphics Tiny (SVGT) for J2ME.

GETTING STARTED
TinyLine comes with several SVGT samples, other samples located on the tinyline.com. From the TinyLine welcome screen, you can use RIGHT and LEFT keys to navigate among bookmarked links. Or, you can select the <Open> command to open the bookmarks.

NAVIGATION
In the <Zoom> mode you can zoom in or zoom out using UP and DOWN keys.
The <Orig View> command returns the viewing image to its original view.
The <Quality> command turns off or on the antialising.

Back

Fig. 31: Set the quality of SVG map, View original map, Show Help

4.1.2 Test with other Raster image WMS Server

For the raster image based WMS, USGS[14] (U.S. Geological Survey) WMS was chosen.

```
http://gisdata.usgs.net/servlet/com.esri.wms.Esrimap?REQUEST=GetCapabilities&SERVICE=wms
```

Listing 24: The GetCapabilities request in URL of USGS WMS

Use *Add* function to add one new WMS server, and save it. The Confirmation after saving tells users that they have add a new server successfully. See Figure 32.

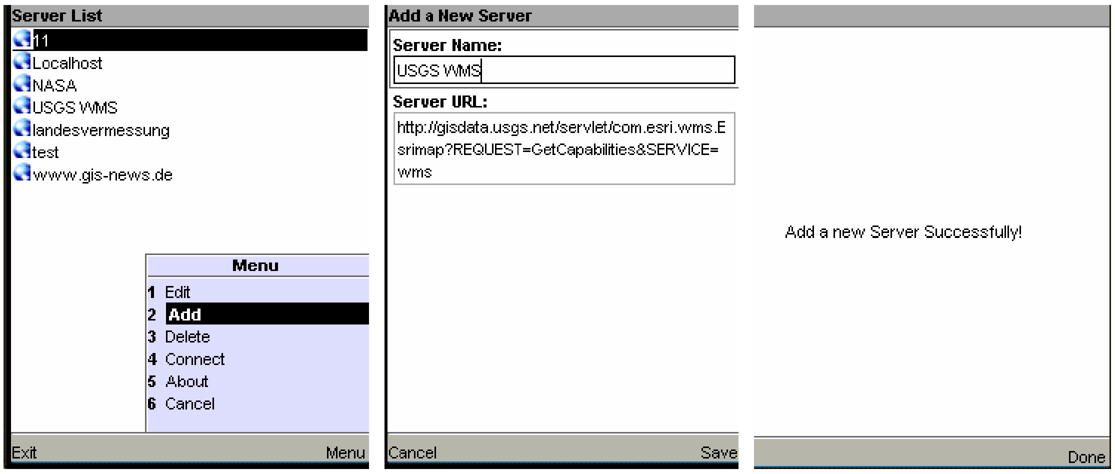


Fig. 32: Add one new WMS server to Server List

Use *Edit* function to edit the selected WMS server if the URL needs to be changed. After saving, the Confirmation tells users that the server was edited successfully. See Figure 33.

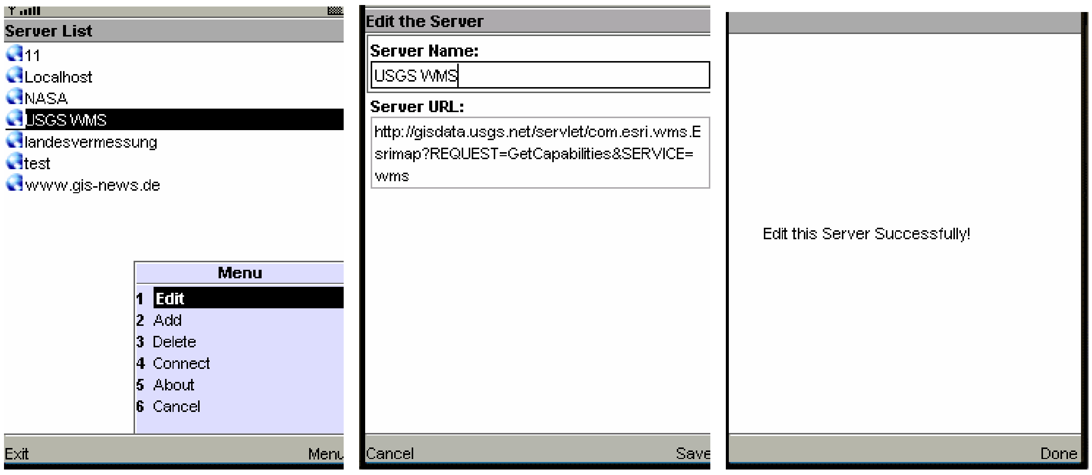


Fig. 33: Edit one WMS server in Server List

[14] http://www.usgs.net

Use Delete function to delete the WMS server, the URL will be displayed to let the users know whether the selected server is the server that they want to delete. And the client will ask users again to make sure this operation. After clicking *Sure*, the Confirmation will display.

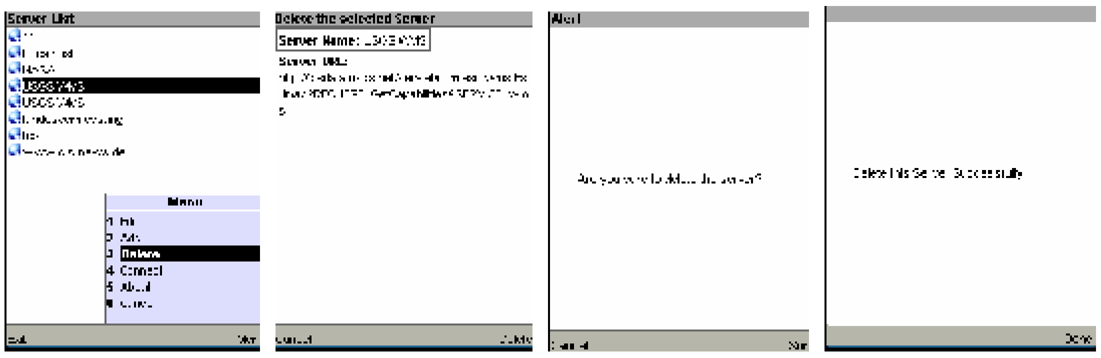
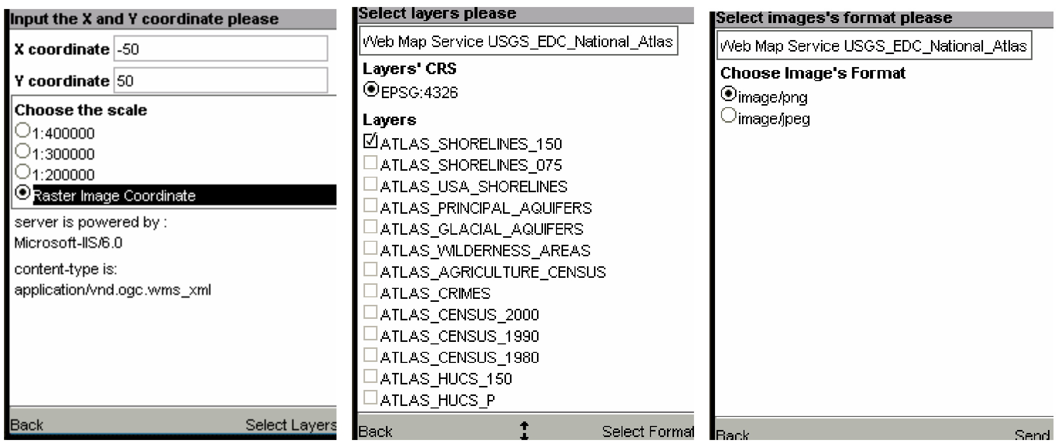


Fig. 34: Delete one WMS server from Server List

Same operation as in Chapter 4.1.1, connect to the WMS and input raster image coordinate. Because the raster image created in this WMS uses pixel coordinate, the Raster Image Coordinate option was chosen. And then select layers of interesting and the PNG image format. See Figure 35.



Input the X and Y coordinate please

X coordinate -50
Y coordinate 50

Choose the scale
○1:400000
○1:300000
○1:200000
[✓ Raster Image Coordinate]

server is powered by :
Microsoft-IIS/6.0

content-type is:
application/vnd.ogc.wms_xml

Back Select Layers

Select layers please

Web Map Service USGS_EDC_National_Atlas

Layers' CRS
[✓ EPSG:4326]

Layers
[✓ ATLAS_SHORELINES_150]
☐ATLAS_SHORELINES_075
☐ATLAS_USA_SHORELINES
☐ATLAS_PRINCIPAL_AQUIFERS
☐ATLAS_GLACIAL_AQUIFERS
☐ATLAS_WILDERNESS_AREAS
☐ATLAS_AGRICULTURE_CENSUS
☐ATLAS_CRIMES
☐ATLAS_CENSUS_2000
☐ATLAS_CENSUS_1990
☐ATLAS_CENSUS_1980
☐ATLAS_HUCS_150
☐ATLAS_HUCS_P

Back Select Format

Select images's format please

Web Map Service USGS_EDC_National_Atlas

Choose Image's Format
[✓ image/png]
○image/jpeg

Back Send

Fig. 35: Input raster image coordinate, Select

If all the parameters are valid, the client will load the PNG format map. *Pan* function can be used to move the map, and *Zoom* function is for zooming in or out. *Orig View* function can be used for view the original map. See Figure 36.

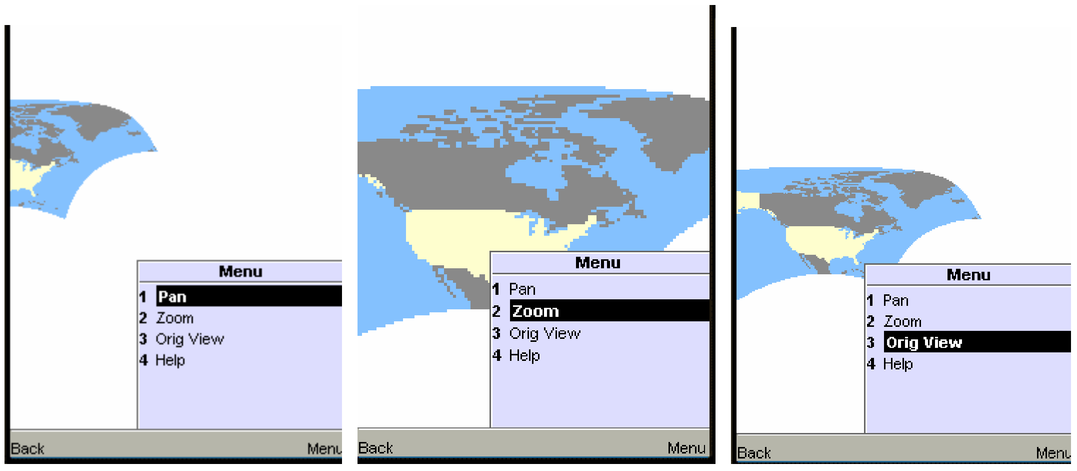
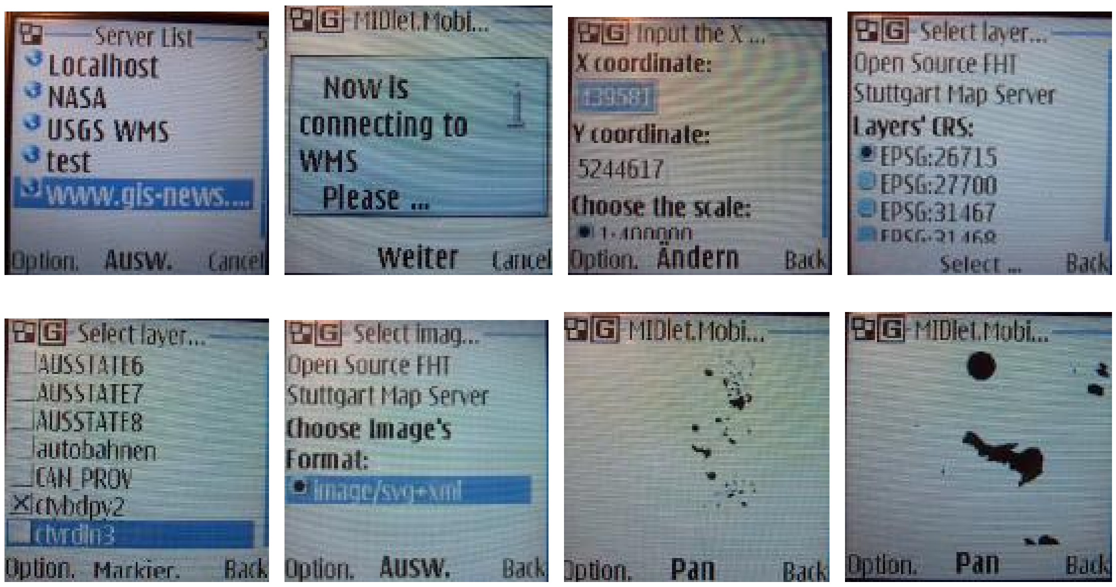


Fig. 36: Move SVG map, Zoom in or out, View original map

4.1.3 Test with Mobile Phone

Mobile phone Nokia 6230i with S40 and 6630 with S60 were chosen for the testing. The following are the interfaces of client in Nokia 6230i. The operations are the same as in chapter 4.1.1, www.gis-news.de WMS was chosen. See Figure 31.



Server List
Localhost
NASA
USGS WMS
test
www.gis-news...
Option. Ausw. Cancel

MIDlet.Mobi...
Now is connecting to WMS Please ...
Weiter Cancel

Input the X ...
X coordinate:
139581
Y coordinate:
5244617
Choose the scale:
1:100000
Option. Ändern Back

Select layer...
Open Source FHT Stuttgart Map Server
Layers' CRS:
EPSG:26715
EPSG:27700
EPSG:31467
EPSG:31468
Select ... Back

Select layer...
AUSSTATE6
AUSSTATE7
AUSSTATE8
autobahnen
CAN_PROV
ctvbdpy2
ctvrdln3
Option. Markier. Back

Select imag...
Open Source FHT Stuttgart Map Server
Choose Image's Format:
image/svg+xml
Option. Ausw. Back

MIDlet.Mobi...
Option. Pan Back

MIDlet.Mobi...
Option. Pan Back

Fig. 37: Test with Nokia 6230i

For non-developers (normal users), the steps of installation on mobile device and how to simulate the client on Personal Computer will be described in Appendix F.

4.2 Summary

The summary of the advantage problems of the developed client is discussed in the following:

4.2.1 Advantage

Web map server has been well developed, both theoretically and practically. The standard map formats of web map server are raster based. SVG format is newly integrated in digital cartography and begin to be supported by more and more WMS. This client can not only display the traditional raster image format PNG, but also the SVGT (SVG and SVGB also) format image. It has the capability to parse mostly of the GetCapabilities Requests XML document responded from WMS, which follow the OGC WMS 1.1.1 (or 1.1.0) Implementation Specification and have the very similar DTD[15] (Document Type Definition). The client can be seen as one common solution of WMS client points to the mobile devices.

4.2.2 Outlook

This thesis work designed and implemented one WMS client for the mobile device. Currently the function is working well, and some advanced improvements can be performed if the work could be continued.

4.2.2.1 Capability to parse all GetCapabilities Request XML Document

Current client can parse mostly of the GetCapabilities Requests XML document responded from WMS, which follow the OGC WMS 1.1.1 or 1.1.0 Implementation Specification and have the very similar DTD. That means if one WMS uses DTD that very different from the standard, the parser will stop parsing. For example, the following 2 Listings, are one part of the same element of two different XML document. The format types, *WMS_XML* and *INIMAGE* in tag *Format* should be picked up.

[15] A Document Type Definition defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements.

```
<Exception>
  <Format> WMS_XML </Format>
  <Format> INIMAGE </Format>
</Exception>
```

Listing 25: The Exception element content 1 in XML content

```
<Exception>
  <Format>
   <WMS_XML />
   <INIMAGE />
  </Format>
</Exception>
```

Listing 26: The Exception element content 2 in XML content

The client can parse the Exception element in Listing 25, these are two elements with same element tag name *Format* in this node *Exception*, which have different element contents: *WMS_XML* and *INIMAGE*. But when the parser meets the *Exception* element node in Listing 26, it doesn't work. Because the Exception element node in Listing 26 has different structure, the element contents in the Listing 25 before, now change their form to two new elements in the *Forma*t element node. The parser can not recognize these two new elements as the content of the *Format* element. The element contents: *WMS_XML* and *INIMAGE* now become tag name of elements. If we still want to parse the element in Listing 26, new codes must be written point to structure in Listing 26. Maybe some method will be used later, and the parser can parse all of the GetCapabilities XML documents that with similar DTD.

4.2.2.2 Dynamically load the image

Current application can display the area that users select, but if the users pan it and the screen boundary encroaches the view box boundary of the image, the map outside of the view box boundary can not be displayed dynamically. For raster image, preload image as byte array can solve the problem.

The whole raster image could be read firstly and stored as byte array in memory of the mobile devices. Because of the PNG special format, each area can be found its corresponding place in the byre array in one sequence. So if the image display moves outside of view box boundary, the client can load one part conjoint byte array from the whole array, and can redraw the byte array to image.

4.2.2.3 Support Google and Microsoft map database

Google map comes new but has shown its multitudinous, amazing and attracting map search functions for all users and developers in very short time. The further development can base on Google or Microsoft map API. Those APIs let developers easily embed Google or Microsoft Maps in their own web pages with JavaScript. If users have need for placing Google Maps on your pages, their page must contain a script tag pointing to that URL, using the Maps API key[16] got from Google Map when users signed up for the API. If users' Maps API key were "abcdefg", then their script tag might look like this:

```
<script src="http://maps.google.com/maps?file=api&v=1&key=abcdefg"

        type="text/javascript">
</script>
```

Listing 27: Embed Google Map in users own web page using JavaScript

For mobile client, the map API can also be embedded in application with J2ME.

4.2.2.4 LBS/GPS Functions

LBS or GPS Functions could be added later to determine the location of the user, the user don't need to input the coordinate and the client can automatically and directly display the map of the location.

[16] A single Maps API key is valid for a single "directory" on your web server.

For LBS function developing, the client can use the Location API for J2ME (JSR 179), a set of generic APIs that can be used for developing location-based services. This API defines an optional package, *javax.microedition.location*, which enables developers to write wireless location-based applications and services for resource-limited devices like mobile phones, and provides mobile applications with information about the device's present physical location and orientation (compass direction), and support the creation and use of databases of known landmarks, stored in the device.

For Location API and GPS receiver, location can be expressed in the widely used *latitude-longitude-altitude* coordinate system. Latitude is expressed as 0-90 degrees north or south of the equator, and longitude as 0-180 degrees east or west of the prime meridian, which passes through Greenwich, England. Altitude is expressed in meters above sea level.

After got the coordinate of one location, if we want to display the location on the map in client, the *latitude-longitude-altitude* coordinate should be converted to North-East (X,Y) Coordinates, such as UTM (Universal Transverse Mercator), using some complex calculation coordinate transformation.

Appendix A[17]

Methods of class KXmlParser

void	**defineEntityReplacementText**(java.lang.String entity, java.lang.String value)
int	**getAttributeCount**()
java.lang.String	**getAttributeName**(int index)
java.lang.String	**getAttributeNamespace**(int index)
java.lang.String	**getAttributePrefix**(int index)
java.lang.String	**getAttributeType**(int index)
java.lang.String	**getAttributeValue**(int index)
java.lang.String	**getAttributeValue**(java.lang.String namespace, java.lang.String name)
int	**getColumnNumber**()
int	**getDepth**()
int	**getEventType**()
boolean	**getFeature**(java.lang.String feature)
java.lang.String	**getInputEncoding**()
int	**getLineNumber**()
java.lang.String	**getName**()
java.lang.String	**getNamespace**()
java.lang.String	**getNamespace**(java.lang.String prefix)
int	**getNamespaceCount**(int depth)
java.lang.String	**getNamespacePrefix**(int pos)
java.lang.String	**getNamespaceUri**(int pos)

[17] THE INFORMATION IS CITED IN JavaDoc of kXML

java.lang.String	**getPositionDescription**()
java.lang.String	**getPrrefix**()
java.lang.Object	**getProperty**(java.lang.String property)
java.lang.String	**getText**()
char[]	**getTextCharacters**(int[] poslen)
boolean	**isAttributerDefault**(int index)
boolean	**isEmptyrElementTag**()
boolean	**isWhitespace**()
int	**next**()
int	**nextTag**()
java.lang.String	**nextText**()
int	**nextToken**()
void	**require**(int type, java.lang.String namespace, java.lang.String name)
void	**setFeature**(java.lang.String feature, boolean value)
void	**setInput**(java.io.InputStream is, java.lang.String _enc)
void	**setInput**(java.io.Reader reader)
void	**setProperty**(java.lang.String property, java.lang.Object value)
void	**skipSubTree**() Skip sub tree that is currently porser positioned on.

Appendix B

The steps to install the SVG WMS in localhost.

1. Install the Apache http server under the folder C:\Apache, download and run apache_2.0.54-win32-x86-no_ssl.msi
2. Install MySQL 4.02, also under the C:\MYSQL folder. Set the supervisor "root" and password "root".
3. Install PHP 5.0.5.5, unzip the php-5.0.5-Win32.zip file under C:\PHP folder
   - Copy C:\PHP\Php.ini-dist to C:\Windows\ and rename it as Php.ini

     COPY F:\PHP\Php.ini-dist C:\Windows

     Ren Php.ini-dist Php.ini
   - Copy all the files in C:\PHP\dlls folder to C:\Windows\System32\, if system is WINDOWS 2000 the path is: C:\WINNT\System32\; if system is WIN9X / 2003 the path is: C:\Windows\System32\

     COPY F:\PHP\dlls\*. C:\Windows\System32\*
   - Copy C:\PHP\php4ts.dll file to C:\Windows\System32\

     COPY F:\PHP\php4ts.dll C:\Windows\System32\
   - Now edit the C:\Windows\Php.ini, find *extension_dir* String and change the path to your own extensions path.

     ; Directory in which the loadable extensions (modules) reside.

     extension_dir = F:\PHP\extensions

   - Find *cgi.force_redirect* String and chage the value from default 1 to 0, and delete the semicolon front

     Find *MySQL* Sting, and modify as following:

     mysql.default_port = 3306

     mysql.default_host = localhost

     mysql.default_user = root

     mysql.default_password = test

   - Find the

     extension=php_mysql.dll,

extension=php_gd2.dll,

extension=php_mbstring.dll String, delete the semicolon front

4. Install phpMyAdmin 2.39.2.2, unzip the file into C:\Apache\htdocs\ phpmyadmin
   - Open the config.inc.php file, find $cfg['Servers'][$i]['host'], and replace the host with localhost, and edit the following user name and password.
   - Find :$cfg[\'PmaAbsoluteUri\'], and input the relative path, localhost\ phpmyadmin

5. Install PHPMyWMS, also put the dictionary into C:\Apache\htdocs\phpmywms

   In the browser input http://localhost/phpmywms/install.php
   - Click install->I Agree
   - Input Database Server: localhost

     Database Name: test

     Database User Name: root

     Database Password : test

     Table Prefix: wms_
   - Now enter the General Setting, input all empty column, this will be included in the GetCapabilities Request XML document.

6. In browser input http://localhost/phpMyAdmin/index.php, go to set the database.

   Choose test database in the left, select db99958geom, import text file:

opengeodb_0_1_3_MI_EPSG31467_2.txt and Bundeslaender_EPSG31467_2.txt.

Field terminated by ",", the others were set by blank.

Till now, the geometry was imported into the database.

Appendix C

Set the Java programming and debugging environment

1. Install jdk-1_5_0_04-windows-i586-p.exe to C:\Program Files\Java\jdk1.5.0_04
2. Set the environment variant
    - In window desktop, right click My Computer, select property.
    - Choose top tab Advanced, click Environment Variant in the bottom.

      In System Variant colomn, click Add New, add new system variant as:

      JAVA_HOME = C:\Program Files\Java\jdk1.5.0_04

      CLASSPATH= .;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar

      Edit Path variant,

      add ;%JAVA_HOME%\bin in the Path variant.
    - Now we have finished the installment setting of JDK, we can have a test HelloWorld

      create one txt file, input the codes below, and rename the txt file with postfix *java* as helloWorld.java

```
class   HelloWorld
 {
   public static void main(String[] args)
   {
     System.out.println("Hello World!");
   }
  }
```

    - Click Start buttom in the windows desktop and click Run, input cmd and the command DOS interface will appear. Input the dictionary where you put the java file. Input "javac helloWorld.java", and the java code will be run. Hello World String will display in the interface.

Appendix D

Install the simulator and programming software

1. Install WTK(sun_java_wireless_toolkit-2_3-beta-windows.exe)
2. Download Eclipse and extract it to any folder
3. Extract eclipseme.feature_1.0.0_site.zip to the Eclipse location, this is the plug-in in Eclipse for the WTK, and set it in Eclipse.
   - Start Eclipse click pull-down menu Help->Software Updates->Find and Install
   - Select eclipseme.feature_1.0.0_site.zip, and click Finish
   - Click pull-down menu Windows->preferences
   - In the left column, choose J2ME->Platform Components, Wireless Toolkits icon will display in the right column.
   - Right click this Wireless Toolkits icon, Add Wireless Toolkit, set the dictionary to where WTK locates.

4. Create new J2ME project
   - Click pull-down menu in Eclipse, File->New->Project
   - Select J2ME->j2ME Midlet Suite->Next->Project Name(HelloWorld)->DefaultColorPhone Platform->Finish
   - Input the codes below:

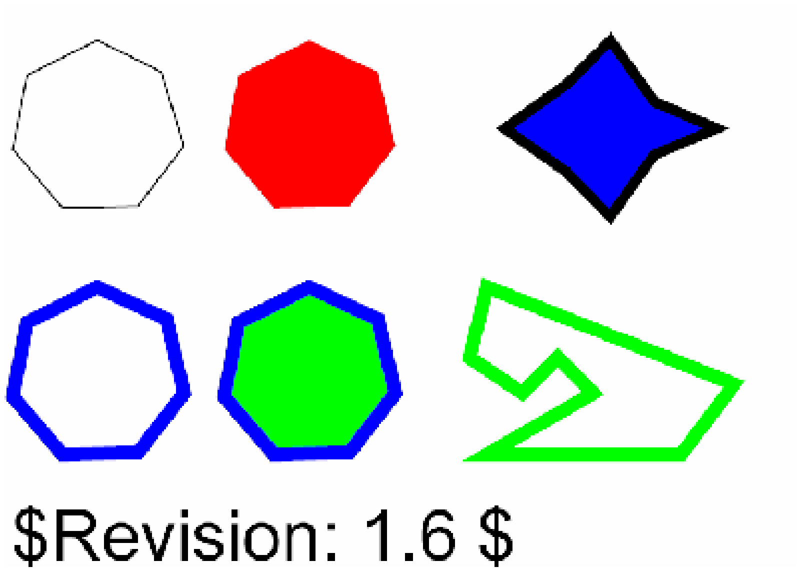
```
 import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
public class Hello extends MIDlet
{
   private Display display;
public Hello()
  {
     display = Display.getDisplay(this);
   }

public void startApp()
{
      Form f = new Form("Test");
       f.append("hello world!");
      display.setCurrent(f);
  }
  public void pauseApp()
  {


  }
  public void destroyApp(boolean unconditional)
  {


  }
}
```

and run the code.

Till now we have succeeded in setting the J2ME develop environment in Eclipse.

Appendix E[18]

An example of SVGT image



The source codes:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1 Tiny//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-tiny.dtd">
<svg version="1.1" baseProfile="tiny" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"    id="svg-root" width="100%" height="100%"
viewBox="0 0 480 360">
            <SVGTestCase xmlns="http://www.w3.org/2000/02/svg/testsuite/description/">
                                    <OperatorScript version="$Revision: 1.6 $"
testname="shapes-polygon-01-f.svg">
                <Paragraph>
                     The rendered picture should match the reference image, except
                     for possible variations in the labelling text (per CSS2 rules).
                </Paragraph>
            </OperatorScript>
            </SVGTestCase>
            <title id="test-title">shapes-polygon-01-t</title>
            <desc id="test-desc">Test that viewer has the basic capability to handle the 'polygon'
element.</desc>
        <!--=============================================================
=====-->
        <!--Content of Test Case follows...                    =====================-->
        <!--=============================================================
=====-->
            <g id="test-body-content">
       <!-- Test case label. -->
       <!--
============================================================= -->
       <!-- First two polygons, convex and "regular".                             -->
```

[18] This example can be found in http://www.tinyline.com/svgt/samples/tiny_test/index.html

```
        <!--
=================================================================== -->
        <!-- Open, convex, "regular". -->
            <polygon id="polygon-01" fill="none" stroke="#000000"
points="59,45,95,63,108,105,82,139,39,140,11,107,19,65"/>
        <!-- Closed, convex, "regular". -->
          <polygon id="polygon-02" fill="red"
points="179,45,218,63,228,105,202,139,159,140,131,107,139,65,179,45"/>
        <!--
=================================================================== -->
        <!-- Third polygon, concave and irregular.                               -->
        <!--
=================================================================== -->
        <!-- Closed, convex, "irregular". -->
        <polygon id="polygon-03" fill="blue" stroke="black" stroke-width="6"    points="350,45 375,80
410,95 375,110 350,145 325,120 290,95 325,70,350,45"/>
        <!--
=================================================================== -->
        <!-- Fourth and fifth polygons.                                              -->
        <!--
=================================================================== -->
        <!-- Closed, convex, "regular". -->
                                <polygon id="polygon-05" fill="none" stroke="#0000FF" stroke-width="8"
points="59,185,98,203,108,245,82,279,39,280,11,247,19,205,59,185"/>
          <!-- Open, convex, "regular". -->
                         <polygon id="polygon-06" fill="#00FF00" stroke="#0000FF" stroke-width="8"
points="179,185,218,203,228,245,202,279,159,280,131,247,139,205"/>
        <!--
=================================================================== -->
        <!-- Sixth polygons, irregular with both concave and convex angles.                -->
        <!--
=================================================================== -->
                                <polygon id="polygon-07" fill="none" stroke="#00FF00" stroke-width="8"
points="270,225 300,245 320,225 340,245 280,280                      390,280 420,240
280,185"/>
                </g>
                <text id="revision" x="10" y="340" font-size="40" stroke="none" fill="black">$Revision:
1.6 $</text>
                <rect id="test-frame" x="1" y="1" width="478" height="358" fill="none"
stroke="#000000"/>
</svg>
```

Appendix F

Install the client on mobile device

At first, the binary files of Mobile_SVGT_WMSClient_S60_1.xx.jad (or Mobile_SVGT_WMSClient_1.xx.jad) and Mobile_SVGT_WMSClient_S60_1.xx.jar (or Mobile_SVGT_WMSClient_1.xx.jar) are in hand.

a). For Nokia mobile phone users, please download Windows PC software for Nokia phone: Nokia PC Suite from http://europe.nokia.com/nokia/0,,72014,00.html. After the installation, connect mobile phone to the PC with USB cable (Nokia production), and run the JAR file, the client will be automatically installed in your Nokia phone. And next b method can also be used.

b). For non-Nokia mobile phone users, please use mobile phone to browser the following website:
http://home.arcor.de/ar11073462/Mobile_SVGT_WMSClient/wap.html, using WAP (Wireless Application Protocol) and GPRS (General Packet Radio Service) surfing functions. Select the link of *Mobile_SVGT_WMSClient_1.xx*, the client will be downloaded and run it.

Simulate the client on PC

Please download JDK and WTK (see Appendix C and D) and install them, and run the JAD file in the PC.

Declaration

The following Master thesis was prepared in my own words without any additional help. All used sources of literature are listed at the end of the thesis.

I hereby grant to Stuttgart University of Applied Sciences permission to reproduce and to distribute publicly paper and electronic copies of this document in whole and in part.

Stuttgart, Date

Signature

References

Kim Topley (2002): *J2ME in a Nutshell.*
Publisher: O'Reilly ISBN: 0-596-00253-X, 478 pages

Yu Feng and Dr. Jun Zhu. (2001): *Wireless Java Programming with J2ME.*
By Sams Publishing, ISBN: 0-672-32135-1, 487 pages

Wang Seng (2004): *Java mobile phone/PDA programming*.
Publisher: Publishing House of Electronics Industry, ISBN: 7-5053-9606-4, 522 pages

Rajinder Singh Nagi (2004): *Cartographic visualization for mobile applications*
. Master Thesis,
International Institute for Geo-information Science and Earth Observation, India.

KMS, National Survey and Cadastre - Denmark/ Flemming Nissen, Anders Hvas,
Jørgen Münster-Swendsen and Lars Brodersen (2003): *Small - Display Cartography* .
Project Thesis, GiMoDig Project. .

Shuichi TAKINO (2001): *GIS ON THE FLY.* Thesis,
International Symposium on Asia GIS 2001, Japan.

Sun Yundong, Liu Changzheng, Gu Ming (2003): *LBS implement using Java and WebServices.* Thesis,
Qinghua University, China.

OGC (2006): *OpenGIS Web Map Service WMS Implementation Specification.*
http://www.opengeospatial.org/docs/01-068r2.pdf [Accessed 20 Feb, 2006]

Naveen Balani (2004): *Using kXML to access XML files on J2ME devices*
Technical document, ibm.com/developerWorks

WEBOPEDIA.COM (2006): *Object-Oriented Programming*. [Internet]. Available from:
http://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html
[Accessed 20 Feb, 2006]

IBM (2006): *J2ME record management store* [Internet]. Available from:

http://www-128.ibm.com/developerworks/library/wi-rms/ [Accessed 20 Feb, 2006]

IBM (2006): *Add XML parsing to your J2ME applications* [Internet]. Available from: http://www-128.ibm.com/developerworks/library/wi-parsexml/
[Accessed 20 Feb, 2006]

IBM (2006): *Combine J2ME applications with WMS* [Internet]. Available from: http://www-128.ibm.com/developerworks/cn/webservices/ws-javamobile/index.html?ca=dwcn-newsletter-webservices
[Accessed 20 Feb, 2006]

W3C (2006): *Extensible Markup Language (XML)* [Internet]. Available from: http://www.w3.org/XML/ [Accessed 20 Feb, 2006]

W3C (2006): *Portable Network Graphics (PNG) Specification (Second Edition)* [Internet]. Available from: http://www.w3.org/TR/PNG/ [Accessed 20 Feb, 2006]

W3C (2006): *Mobile SVG Profiles: SVG Tiny and SVG Basic*[Internet]. Available from: http://www.w3.org/TR/SVGMobile/ [Accessed 20 Feb, 2006]

SUN (2006): *Parsing XML in J2ME* [Internet]. Available from: http://developers.sun.com/techtopics/mobility/midp/articles/parsingxml/
[Accessed 20 Feb, 2006]

SUN (2006): *Introduction to J2ME Web Services* [Internet]. Available from: http://developers.sun.com/techtopics/mobility/apis/articles/wsa/
[Accessed 20 Feb, 2006]

SUN(2006): *Java 2 Platform, Micro Edition (J2ME)* [Internet]. Available from: http://java.sun.com/j2me [Accessed 20 Feb, 2006]

HTMLGOODIES.COM (2006): *Image Formats* [Internet]. Available from: http://www.htmlgoodies.com/tutorials/web_graphics/article.php/3479931
[Accessed 20 Feb, 2006]

Rahul Kumar Gupta (2003): *Intelligent applicances and J2ME's RMS*. [Internet]. Available from: http://java.ittoolbox.com/documents/peer-publishing/intelligent-applicances-and-j2mes-rms-2755 [Accessed 20 Feb, 2006]

WIKIPEDIA – THE FREE ENCYCLOPEDIA (2006): *PNG*. [Internet]. Available from: http://en.wikipedia.org/wiki/Png [Accessed 20 Feb, 2006]

BU.EDU (2006): *Raster Image Files* [Internet]. Available from:

http://scv.bu.edu/tutorials/ImageFiles/image101.html [Accessed 20 Feb, 2006]

DEVELOPER.COM (2006): *MIDP Programming with J2ME* [Internet].
Available from: http://www.developer.com/java/j2me/print.php/10934_1561591_6
[Accessed 20 Feb, 2006]

XML.COM (2006): *Going Mobile With SVG: Standards* [Internet].
Available from: http://www.xml.com/pub/a/2004/06/16/mobilesvg.html
[Accessed 20 Feb, 2006]

XML.COM (2006): *Mobile SVG* [Internet]. Available from:
http://www.xml.com/pub/a/2004/08/18/sacre.html [Accessed 20 Feb, 2006]

SVG.ORG (2006): *Shipping and Announced SVG Phones* [Internet].
Available from: http://svg.org/special/svg_phones [Accessed 20 Feb, 2006]

DEVELOPER.COM (2006): *MIDP Programming with J2ME* [Internet].
Available from: http://www.developer.com/java/j2me/article.php/10934_1561591_1
[Accessed 20 Feb, 2006]

INTERNETNEWS.COM (2006): *Symbian at a Mobile Loss* [Internet].
http://www.internetnews.com/wireless/article.php/3584431 [Accessed 20 Feb, 2006]

CARTO.NET (2006): *SVG tutorial, example and demonstration* [Internet].
http://www.carto.net/papers/svg/samples/ [Accessed 20 Feb, 2006]

GOOGLE MAP (2006): *Develop Your Own Location-Based Services Using Google Maps* [Internet].
http://www.google.com/apis/maps/ [Accessed 20 Feb, 2006]