DESIGN AND IMPLEMENTATION OF A SEARCH TOOL FOR ROADS ON POCKET PCs FOR MOBILE GIS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALPER DİNÇER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

DECEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

_____

Prof. Dr. Güney Özcebe
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science

_____

Assoc. Prof. Dr. Mahmut Onur Karslıoğlu
Supervisor

Examining Committee Members

Prof. Dr. Vedat Toprak (METU, GEOE) _____

Assoc. Prof. Dr. Mahmut Onur Karslıoğlu (METU, CE) _____

Assoc. Prof. Dr. Sadık Bakır (METU, CE) _____

Dr. Bahattin Coşkun (METU, CE) _____

Dr. Jurgen Friedrich (Başkent Uni, CENG) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Alper DİNÇER

Signature :

ABSTRACT

DESIGN AND IMPLEMENTATION OF A
SEARCH TOOL FOR ROADS ON POCKET PCs
FOR MOBILE GIS

Dinçer, Alper
MS., Department of Civil Engineering
Supervisor : Assoc. Prof. Dr. Mahmut Onur KARSLIOĞLU

December 2006, 70 pages

The aim of this study is to develop a search tool for roads for mobile GIS application. The satellite image of Ankara is the base map of program. There is also a search option for the roads. The application is based on open source libraries, which are ECW for imagery and SQLite for the database of vector. The application is coded in Embedded Visual C++.

The study shows that mobile GIS applications can be prepared by the help of open source libraries. There is no need to buy a commercial product to mobilize the GIS.

Keywords: Mobile GIS, ECW, SQLite, Open Source

ÖZ

CEP BİLGİSAYARLARINDA MOBİL
CBS İÇİN YOL ARAMA KİTİ
TASARIM VE KURULUMU

Dinçer, Alper
Yüksek Lisans, İnşaat Mühendisliği Bölümü
Tez Yöneticisi : Assoc. Prof. Dr. Mahmut Onur KARSLIOĞLU

Aralık 2006, 70 sayfa

Bu çalışmanın amacı yol arama kiti olarak kullanılabilecek bir mobil CBS uygulaması geliştirmektir. Ankara'nın uydu görüntüsü programda altlık olarak kullanılmıştır. Bu uygulama açık kaynak kodlu kütüphaneler kullanılarak yazılmıştır. Görüntüleme için ECW, vektör alt yapısı için gereken veritabanı olarak da SQLite kütüphaneleri kullanılmıştır. Uygulama Embedded Visual C++ altında kodlanmıştır.

Bu uygulama ile açık kaynak kodlu kütüphaneler kullanılarak mobil CBS yazılımı yazılabileceği gösterilmiştir. Böylelikle aynı işi yapan ücretli yazılımlara gerek olmayabileceği gösterilmiştir.

Anahtar Kelimeler: Mobil CBS, ECW, SQLite, Açık Kaynak Kod

To My Parents and My Fiancé

ACKNOWLEDGMENTS

I would like to express my special thanks to my supervisor Assoc.Prof.Dr. Mahmut Onur Karslıoğlu, for his guidance, patience and motivation. I am sincerely in debt to him for his continuous support throughout this study.

I would like to thank to Jurgen Friedrich for his suggestions and guidance.

I would like to thank to Önder Halis Bettemir for his valuable assistance and guidance in geodesy.

I would like to thank to Çağlar Şenaras for his cooperation and guidance in programming.

I would like to thank to Denizcan Çiğşar for sharing his valuable knowledge and experiences in programming.

Finally, I would express my special thanks to my family and my fiancé for their patience and compassion throughout the study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
API	Application Programming Interface
ArcIMS	ArcInfo Internet Mapping Server
ArcPad	ArcInfo Mobile GIS Application
ARM	Advanced RISC Machine
AWT	Advanced Windows Toolkit
BILSAT I	Satellite of BILTEN
C#	Microsoft's Programming Language
CAB	Cabinet (Microsoft Windows file extension)
CAD	Computer-Aided Design
CCD	Charge-Coupled Device
CDC	Connected Device Configuration
CF	Compact Framework
CLDC	Connected Limited Device Configuration
CMYK	Cyan-Magenta-Yellow-Key/blacK
COM	Component Object Model (Microsoft)
CPU	Central Processing Unit
DLL	Dynamic Linked Library
DN	Digital Number
DOS	Disk Operating System
ECW	Enhanced Compressed Wavelet
ESRI	A GIS Company
EXE	Executable (File Name Extension)
FP	Foundation Profile
GeoTIFF	Geostationary Earth Orbit Tagged Image File Format

GHz	Gigahertz (thousands of MHz)
GIF	Graphic Interchange Format
GIS	Geographic(al) Information System
GPL	General Public License (GNU)
GPS	Global Positioning System
GUI	Graphical User Interface
IDE	Integrated Development Environment
J2ME	Java 2 Micro Edition
JPEG	Joint Photographic Experts Group
JVM	Java Virtual Machine
KB	KiloByte (1024 Bytes)
KVM	Kilo Virtual Machine (Java)
LZW	Lempel-Ziv-Welch (algorithm)
MapInfo	A GIS Company
MapX	Mobile GIS Application of MapInfo
MB	Megabyte (1024 KB)
MHz	Megahertz (million Hertz)
MrSID MG2 and MG3	Multi-Resolution Seamless Image Database
OS	Operating System
PDA	Personal Digital Assistant
PDF	Portable Document Format (Adobe Acrobat)
PFP	Personal Basis Profile
PNG	Portable Network Graphics
POI	Point of Interest
PP	Personal Profile
ROI	Region-of-Interest
RTTI	Run Time Type Information
SD	Secure Digital
SDK	Software Development Kit

SE	Second Edition
SOAP	Simple Object Access Protocol (XML protocol)
SQL	Structured Query Language
Targa	Truvision Advanced Raster Graphics Array
TB	Terabyte (1,024 Gigabytes)
TIFF, TIF	Tagged Image File Format
UTM	Universal Transverse Mercator (cartography)
VB.NET	Microsoft’s Programming Language
Windows CE	Windows Compact Edition
XML	eXtensible Markup Language

CHAPTER 1

INTRODUCTION

Geographic Information System is a rapidly developing area nowadays, but it is not enough to use it on desktops as analyses tool. GIS is sometimes desired on field rather than offices as a data collecting or guiding tool. Mobile technologies are required to mobilize the GIS or GIS tools. Mobile phones, PDAs, tablet PCs are the examples of devices to mobilize the GIS.

Mobilizing GIS is not an easy task to do. There are lots of details to think about it. As the devices get smaller, the problems about programming become more difficult.

Mobile devices have moderate to low system sources than desktop or laptop computers. So one has less CPU power for calculations and less memory to store the data. These are the most important bottlenecks of mobile GIS.

There are also other limitations such as screen size, battery life and memory management. These are also very important for mobile GIS in the field.

GIS companies who have a large market shares have different kinds of mobile GIS toolkits to make mobile GIS applications, such as ESRI, MapInfo.

ESRI has a product called ArcPad, which is the mobile version of Arc Application and ArcPad Application Builder, which is the development framework for creating custom mobile GIS applications with ArcPad. ArcPad is a buy and use application, which has no flexibility for changing user needs. On the other hand ArcPad Application Builder is a framework for programmers to create or modify the interface.

ArcPad is designed for PDAs to collect data for ESRI's desktop GIS application, ArcGIS. It cannot be a standalone GIS application; it must be used with ArcGIS in order to provide full range of GIS functionality. ArcPad Studio is the software development kit for ArcPad for building custom mobile GIS applications. (P. Clegg, 2006)

ArcPad does not collect information via touch screen. GPS receivers, rangefinders and digital cameras can be connected to ArcPad to collect different kind of information. So images of field can be taken while collecting the x,y and z coordinates. Linking the geographic data with sound or video is also possible with the integration of ArcPad.

ArcPad supports most popular data formats shown in next paragraph but this does not mean that every file can be opened by ArcPad because of the limitations of PDAs. These formats can be used as a reference layer or as editable layer.

Data Formats supported by ArcPad are as follows (ESRI, 2006);

- ESRI ShapeFile
- ArcPad graphics layers
- ArcPad photo layers
- MrSID MG2 and MG3
- JPEG
- JPEG2000
- GIF
- PNG
- Windows Bitmap
- CADRG
- TIFF
- ArcIMS Image Services via access to the Internet

Contrary to ESRI, MapInfo only serves as development tools for professionals, which is called MapX Mobile. MapX Mobile has a software development kit (SDK) which can be used with Embedded C++ 4.0 or .NET technologies. MapX Mobile is less popular than ArcPad because of only giving SDK but this does not mean that it has less ability to process GIS data.

As it is seen in next paragraph, MapX Mobile also supports the most popular data formats, including its own format, TAB files,

but it has also some limitations like ArcPad due to PDAs' limitations.

Data Formats supported by MapX Mobile are as follows (MapInfo, 2006);

- MapInfo Tab Files
- ECW
- Spot
- MrSID
- JPEG
- JPEG2000
- Targa
- Windows Metafile
- GIF
- PNG
- Windows Bitmap
- PCX
- TIFF
- Photoshop
- ESRI ShapeFile

Until now some popular mobile GIS applications have been introduced. These applications are evolving everyday and gaining new features day by day, but the newest feature of these applications is "*Wavelet Compressed Raster*" support, JPEG 2000 and ECW. This support is a must for mobile GIS applications because it gives more flexibility and speed to applications due to its wavelet compression. JPEG 2000 and ECW have open source

libraries, so there is no need to pay for using JPEG 2000 or ECW formats.

There are many solutions for mobile GIS. Only the most popular ones are introduced above but most of the solutions are commercial solutions. There are some rebates for academic purposes but they are still commercial. The data formats are available to public access, so there is no need to pay for mobile GIS applications. The goal of this work is to show that a component of a mobile GIS can be developed without requiring any commercial software.

The problem was showing the map of Ankara with frequent updates. Frequent update of vector maps is a problem for commercials, governments and municipalities. Instead of vector maps satellite images are used and it is very easy task to update the roads on satellite images. Final solution is to show roads as vectors on satellite image.

Solution of the problem has started with selection of the image formats. Image formats are important when using satellite images. Size of the satellite image of Ankara, which was taken from BILSAT I, is approximately 2 MB in TIFF format. 2 MB size is very enormous for PDAs, which is even large enough for desktop computers to handle. JPEG and TIFF image formats are selected first to use in application, but reading 2 MB of these images in PDAs is a problem. The whole JPEG or TIFF image must be loaded into memory to make computation, so this complicates the usage

of image on PDAs. There can be a solution not to load the whole image into memory, the image can be split up to pieces and each piece can be referenced to other piece, which can be called *tile grid concept. (*Hendrey, 2006) This is used by the commercial map servers, such as Google Maps. This can be good solution for stand-alone architecture but image must be splited into pieces for every zoom level, so there are lots of image pieces. Image can easily be loaded into memory but there is complexity of relations of image pieces.

Therefore, this method has been ignored. Instead, wavelet compression of images is applied. The first application of wavelet compression on images is JPEG 2000, which will be a new standard in the industry in few years.

"Despite the phenomenal success of the JPEG baseline system, it has several shortcomings that become increasingly apparent as the need for image compression is extended to such emerging applications as medical imaging, digital libraries, multimedia, internet and mobile." (Rabbani, 2002)

There is a need for a new format and this format will be JPEG 2000 or some format with Wavelet Compression. (Rabbani, 2002) There are lots of new features coming with wavelet compression; multiple resolution representation, tiling and Region-of-interest (ROI) coding. Multiple resolution representation and tiling are the most important characteristics for this work because there is no need to split image into pieces. There is an embedded tiling in

the image. Different resolutions of image are stored in the image as wavelets and multiplication of wavelets with coefficients makes the whole image. There is also other advantage of wavelet compression, which is Region-of-Interest (ROI) feature. By the help of this feature, there is no need to read the whole image to show a part of it. A part of an image can be read without reading the whole image.

JPEG 2000 is a standard for wavelet compressed images. This standard can be developed for specific areas, such as medical imaging or GIS. Organizations took the standards and develop their own wavelet compressed image formats. ECW and Mr.Sid are the GIS examples of this behavior. Both of them are compressed with wavelet encoding and designed for GIS applications. Designing for GIS means that the image holds geospatial metadata, which can be the geographic position of the image and the projection system being used.

As an image format ECW is chosen for this work because ER Mapper, which is the organization developing ECW, gives ECW source codes with Free or GPL license. Free license is up to 500 MB so it is enough for the satellite image used.

Vector part can be solved by standard data formats, such as ESRI ShapeFile or MapInfo Tab Files. As showing roads on a digital image is an easy task such vector data formats are not needed to use. If it is the case, standard GIS functions and standard database or XML files can be used in order to show the roads.

Since the necessary database must be worked under PDA operating system, the best database for PDA is selected which is SQLite. SQLite is an implementation of SQL92 standard and works on most of systems including PDAs. SQLite works without any SQL Daemon, so a single SQL file is enough for moving data. This is the best feature of SQLite to use it on PDAs. SQLite is also an open source project, which is another advantage of library.

After the selection of the database, the next step is the designing of database for the work to be done here. The database design must be simple enough not to force PDA CPU power because there are both image and vector processing at the same time and image processing spends most of the CPU power. Another advantage is that PDA uses less CPU power. As the CPU power increases, battery consumption also increases. There is a linear relation between CPU power and battery consumption.

Finally, all the libraries put together with a graphical user interface (GUI) and some GIS functions under Embedded Visual C++ 4.0. Embedded Visual C++ 4.0 is an integrated development environment (IDE) which normally consists of a source code editor, a compiler and interpreter, build-automation tools, and a debugger. Various tools are available to simplify the construction of a graphical user interface with programming are integrated as well. Like many modern IDEs also integrate a class browser, an object inspector and a class hierarchy diagram, for use with object oriented software development.

The final program has standard GIS functions; zoom in, zoom out, move and learn coordinates. There is also search option to search digitized roads on satellite imagery. Program is running on Windows Mobile 2003 and Windows Mobile 2003 Second Edition (SE) operating systems. The only required program to install is ECW library, no other program or daemons are necessary on PDA. When the ECW library is installed, the program folder can be just copied to run on PDA.

Finally, all the elements selected at the end of the thesis are shown in Table 1.1.

Table 1.1: Summary of elements selected.

Operating System	Windows Mobile 2003 SE
Programming Architecture	Stand-Alone Architecture
Image Format	ECW
Vector Format	Simple Line and Point Data
Database	SQLite
Programming Language	Embedded Visual C++ 4.0
Image SDK	ER Mapper ECW SDK
Database SDK	SQLite C++ SDK

CHAPTER 2

BACKGROUND INFORMATION

2.1. Geographic Information System (GIS)

GIS is the abbreviation of Geographic Information System, which is a collection of computer hardware, software, and geographic data for capturing, managing, analyzing, and displaying all forms of geographically referenced information. All information associated with geographic data can be handled by GIS. (Worboys and Duckham, 2004, 1-16)

Today GIS is an emerging technology in the world. Companies, universities and governments' demands are increasing on GIS technologies, because it makes life easier than before. By the day, GIS is available for public use. For example Google Maps, Yahoo Maps and Microsoft Local are the front side of public maps. They help people to travel easily, find places accurately and explore the world.

2.2. Personal Digital Assistant (PDA)

PDAs development life cycle can be divided into two parts. In the first era, they can be classified as digital assistants which have a functions of organizers, providing management of contact lists, calendars, diaries, calculators, etc. The second era is that they gain computer properties. They can have operating systems,

programs rather than organizers or calculators. RAM, ROM and even small hard drives can be included within the device. These devices can also have better connectivity options such as wireless LAN, Bluetooth and GPRS. (Casademont, 2004)

Different kinds of PDAs are also available: Palm OS PDAs, Windows Mobile PDAs and Linux PDAs. They have a lot of common features, but they differ in operating systems. Windows Mobile PDAs are the most popular ones because of the easy use of operating systems and variety of different programs running on it.

2.3. Mobile GIS

Developments in PDAs provided a new area in GIS, namely mobile GIS. Mobile GIS is not a different term than GIS. It is the integration of GIS into mobile devices such as mobile phones, pocket PCs and other handheld devices. The obvious definition is "GIS on mobile devices." However, the term Mobile GIS can also encompass stationary devices communicating with mobile devices over a network.

The necessity of mobile GIS are summarized as follows: (Hassin, 2003)

- Access to geographical data in the field
- Ability to collect data in the field and in real-time
- Ability to add geographical information to data captured

On the other hand, it isn't widely deployed from the following reasons;

- Hardware resources on mobile devices are limited.
- Software that is developed for these devices are not widely available.
- Connection options in the field are limited. It is very hard to find WI-FI hotspot or GPRS connection.
- Costs on mobile devices are more than desktop devices.

As a result, mobile GIS is a developing technology, so developing a GIS application on mobile devices requires more attention than other platforms. All the stages must be searched, planned and applied.

2.4. Displaying the Earth Features

Image is the two dimensional matrix which consists of real numbers. This matrix defines the boundary of image and color depth of image. An image is worth a thousand words in our daily life. Image is also very important in Geographic Information Systems like in our daily life. A small database can be replaced by an image in GIS in order to show the same area.

Image as a data source is one of the major components of GIS because of its functionality, but the term image in GIS is different than the daily life. The image term in GIS mostly describes the satellite images (remotely sensed images) or air photos.

A remotely sensed image (by CCD frame or push broom camera or radar) is a two-dimensional grid of data; each of its elements is a pixel (picture element) whose coordinates are known and whose light intensity has a DN (Digital Number) value. The coordinates of the pixels and their DN values describe the image as rows, called lines, and columns, called samples. An 8-bit pixel provides up to 256 brightness levels (level 0 is set to black, while level 255 is set to white), the brightness levels are also referred to as 'grey levels'. In false color image processing, those pixels which have the same DN value are marked. This technique is used, for example, to differentiate between various types of terrain or species of vegetation - to show changes, which are otherwise not perceptible to the human eye. (Kramer, 2002, 2431)

There are some properties of satellite images, which are spatial resolution, radiometric resolution and number of bands.

Spatial resolution is the most important property of an image. As the resolution increases, number of features on the image also increases. For example one can see the houses clearly on 1m resolution image, but one can not see the houses on 100m resolution images. (Kramer, 2002, 2452)

The satellite images are mostly in raw format when taken, which means that one can read the image through the matrix, but raw format is not suitable for storing. So raw images must be converted to another image format to store and serve. There are lots of image formats for GIS; most popular are JPEG, TIFF, GeoTIFF and JPEG2000.

2.4.1. Image Formats

Image file formats provides the standardization of storing and organizing of image data. Image files are composed of pixels, which comprise an image in the form of a grid of columns and rows. Every pixel has a value, which represents the brightness or color. The popular image formats used in GIS are explained below. (Wikipedia, 2006)

2.4.1.1 JPEG

The JPEG is an image format which is lossy. 3 channels of color, each is 8-bit, red, green and blue produces the colored JPEG image. Compression rate is very high in JPEG format, so difference between the original and compressed image is unnoticed. (Miano, 1999, 47-57)

"Photographic images are best stored in a lossless non-JPEG format if they will be re-edited in future, or if the presence of small "artifacts" (blemishes), due to the nature of the JPEG compression algorithm, is unacceptable. JPEG is also used as the image compression algorithm in many Adobe PDF files." (Wikipedia, 2006)

2.4.1.2 TIFF

The TIFF (Tagged Image File Format) has different options for storing the pixels. 16-bit per color or 8-bit per color can be selectable for images. So 2 different type of TIFF can be formed, which are 24-bit or 48-bit images. TIFF can be lossy or lossless. TIFF is capable of handling device-specific color spaces, such as the CMYK defined by a particular set of printing press inks. (Wikipedia, 2006)

2.4.1.3 GIF

GIF (Graphic Interchange Format) is limited to an 8-bit palette, or 256 colors. So simple shapes or diagrams can be stored easily with GIF format. (Miano, 1999, 171-188) GIF format also supports animation, which is widely used on internet. (Wikipedia, 2006)

2.4.1.4 PNG

The PNG (Portable Network Graphics) file format can be seen as an open-source competitor of GIF format. The most important difference between PNG and GIF is that the PNG format supports 16 million colors. (Miano, 1999, 189-212)

"The lossless PNG format is best suited for editing pictures, and the lossy formats like JPG are best for final distribution of photographic-type images because of smaller file size." (Wikipedia, 2006)

2.4.1.5 Mr.Sid, ECW and JPEG 2000

These formats are using the same principle, wavelet compression. Wavelet compression is a form of data compression well suited for image compression (sometimes also video compression and audio compression). The goal is to store image

data in a file using as little space as possible. A certain loss of quality is accepted, known as lossy compression.

Based on a wavelet transform, the wavelet compression methods are better at representing transients, such as percussion sounds in audio, or high-frequency components in two-dimensional images, for example an image of stars on a night sky. This means that the transient elements of a data signal can be represented by a smaller amount of information than it would be in case of some other transforms, such as the more widespread discrete cosine transform.

Wavelet compression is not good for all kinds of data: transient signal characteristics mean good wavelet compression - smooth, periodic signals are better compressed by other methods. (Wikipedia, 2006)

Properties of JPEG 2000 are as follows (Rabbani, 2002);

- Improved compression efficiency
- Lossy to lossless compression
- Multiple resolution representation
- Embedded bit-stream (progressive decoding and SNR scalability)
- Tiling
- Region-of-interest (ROI) coding
- Error resilience
- Random code stream access and processing
- Improved performance to multiple compression / decompression cycles
- A more flexible file format

2.5. Displaying Roads on Satellite Images

Roads on earth can be shown by the help of vector on satellite image. Vector is a data structure, used to store spatial data. Vector data comprises of lines or arcs, defined by beginning and end points, which meet at nodes. The locations of these nodes and the topological structure are usually stored explicitly. Features are defined by their boundaries only and curved lines are represented as a series of connecting arcs. Vector storage involves the storage of explicit topology, which raises overheads, however it only stores those points which define a feature and all space outside these features is 'non-existent'.

A vector based GIS is defined by the vectorial representation of its geographic data. According with the characteristics of this data model, geographic objects are explicitly represented and, within the spatial characteristics, the thematic aspects are associated.

There are different ways of organizing this double data base (spatial and thematic). Usually, vectorial systems are composed of two components: the first one manages spatial data and the second one manages thematic data. This is called hybrid organization system, as it links a relational data base for the attributes with a topological one for the spatial data. A key element in this kind of systems is the identifier of every object. This identifier is unique and different for each object and allows

the system to connect both data bases. (Worboys and Duckham, 2004, 221-255)

2.6. Programming Architectures

2.6.1. Stand-Alone

The simplest programming architecture is the Stand-Alone Client architecture which is shown in Figure 2.1. All the data connections are completed within the device in this architecture. The device stores, interprets and displays the geographic data. If there is a possibility to store all the geographic data on the device, the speed of the application is better than the client-server type application.

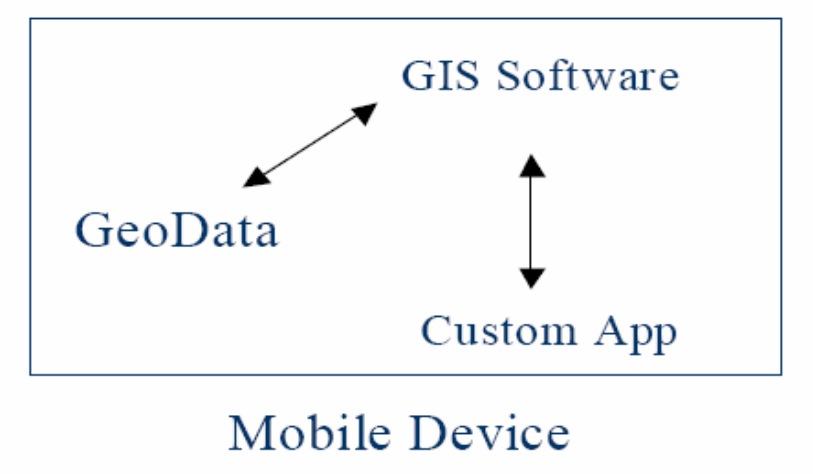

Figure 2.1: Work Diagram of Stand-Alone Client Architecture

Regardless of the benefits, this architecture has some pretty major limitations. First, the storage is limited to store all the

geographic data. Another limitation is that there is no connection to outside to update geographic data. (Hassin, 2003)

2.6.2. Client-Server

The limitations of stand-alone architecture can be eliminated by using different approach. Storing geographical data on other computer and making a connection between mobile device and server is a good solution for storage limitation on mobile devices. The mobile GIS application remains the same except the geographical data source. The architecture can be seen on Figure 2.2.

This approach is good for multiple mobile devices with single geographical data source. Multiple clients get the same data from server, so update process is easier than stand-alone architecture, because updating the server geodata affect all the clients. (Hassin, 2003)

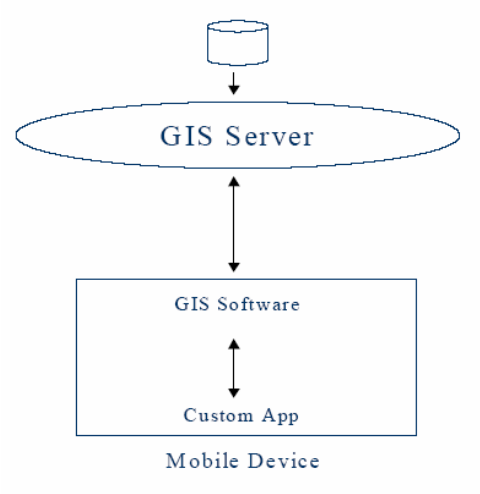

Figure 2.2: Work Diagram of Client-Server Architecture

2.7. Programming Languages for Mobile Devices

2.7.1. Java (J2ME)

Java 2 Micro Edition (J2ME) is the mobile version of Java for mobile devices. There are four configuration-profile options for developers to deploy a J2ME application. (Hassin, 2003)

- Connected Limited Device Configuration (CLDC) – This configuration uses the KVM, an extremely lightweight version of the Java Virtual Machine (JVM). (Sun, 2005)

    - Mobile Information Device Profile (MIDP) – The MIDP is the only profile available for the CLDC and is intended for mobile devices with very limited resources, such as mobile telephones.

- Connected Device Configuration (CDC) – This configuration uses a full JVM, but its profile determines how much functionality is available. (Sun, 2005)

    - Foundation Profile (FP) – As its name indicates, this profile provides just the basic foundation classes.

    - Personal Basis Profile (PFP) – The PFP provides the foundation classes as well as additional graphics functionality.

    - Personal Profile (PP) – This profile is a superset of the PFP and actually provides the entire Advanced Windows Toolkit (AWT) to the developer.

To balance portability with performance and feasibility in the real world, J2ME contains several components known as configurations, profiles, and optional packages. Each valid combination of a configuration and a profile targets a specific kind of device. The configurations provide the most basic and generic language functionalities. The profiles sit on top of configurations and support more advanced APIs, such as a graphical user interface (GUI), persistent storage, security, and network connectivity. The optional packages can be bundled with standard profiles to support specific application needs. (Yuan, 2003, 20)

2.7.2. .NET Compact Framework

.NET Compact Framework (CF) is the mobile version of .NET Framework for mobile devices by Microsoft. It can be seen as an answer to J2ME. .NET Compact Framework applications are developed in exactly the same way as a regular .NET desktop application. Developing mobile applications with .NET Compact Framework requires very little experience than J2ME. (Hassin, 2003)

One of the design goals of the Compact Framework was to create a "portable (and small) subset of the desktop Framework, targeting multiple platforms." To support this goal Microsoft created the architecture shown in Figure 2.3. (Fox and Box, 2003, 38-49)

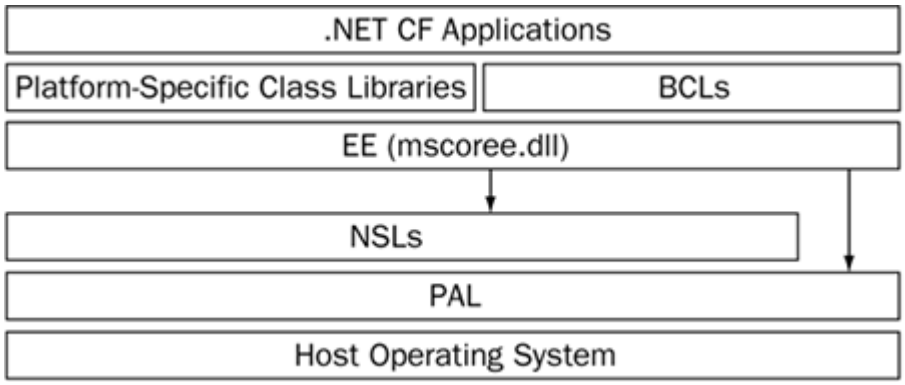

Figure 2.3: The Compact Framework Architecture (Microsoft, 2005)

2.7.3. Embedded Visual C++

C++ is a general-purpose, high-level programming language. It is a statically-typed free-form multi-paradigm language supporting procedural programming, data abstraction, object-oriented programming, and generic programming. Since the 1990s, C++ has been one of the most popular commercial programming languages.

"Windows CE is a modular operating system designed to build computing devices. Its modularity means that engineers can select which parts of the operating system are required—for example, a device may not need a keyboard or a display, but perhaps it needs networking capability. By selecting only those modules a device requires, the size and cost of the device can be controlled. Device manufacturers can use the Microsoft Platform Builder product to produce their own customized devices, or use one of the standard configurations such as the Pocket PC or Handheld PC." (Grattan and Brain, 2000, 3)

Embedded Visual C++ is the mobile version of C++ which can be a programming language for Windows CE operating system. Mobile devices have different features, so C++ can be optimized in order to use it on mobile devices. The visual part is also important for programmers. There is no need to deal with codes to build a GUI. All the tools to prepare a GUI is built in the Embedded Visual C++ IDE.

One thing should be discussed before using Embedded Visual C++, which is the selection between Embedded C++ and .Net Compact Framework. Table 2.1 and Table 2.2 must be analyzed before selecting the language if there are no other limitations such as SDK requirements.

Table 2.1: Properties: Native Code vs Managed Code (Yao, 2004)

Native Code	**Managed Code**
C/C++ & Win32 API	C#/VB.NET & .NET CF
EXEs & DLLs - Native CPU instructions	EXEs & DLLs - IL instructions; JIT to native CPU instructions
Portable source code	Portable binary code
Manual cleanup	Garbage Collection
ActiveX / COM	COM not supported
No run-time required (OS is the runtime)	Run-time required (.Net Compact Framework needed)

Table 2.2: Native Code vs Managed Code (Yao 2004)

Native Code	**Managed Code**
Device drivers	GUI application code
Shell extensions	Web service clients
Real-time threads	Build managed DLLs
ActiveX / COM	Custom display-based smart devices
CE property databases	Database (ADO.NET) clients

CHAPTER 3

METHODOLOGY USED

The work to be done is to show Ankara on map and search the roads on PDA. Due to popularity of Pocket PCs, Pocket PC with Windows Mobile 2003 Second Edition is chosen as a reference PDA.

The methodology of this work is as follows:

- Selection of Programming Architecture :

  Selection of programming architecture is an important step with respect to hardware requirements, connection speed and requirements, user needs and programming experience.

  Important parameters which effect the decision on selecting programming architecture are connection speed and requirements. In Turkey, it is not possible to find an internet connection everywhere. If there is a connection, its speed may be not enough for your application.

  The other important parameter is hardware requirements. It is unnecessary to publish maps and images through the internet if there is no internet connection. So server is redundant for this project.

According to parameters above, client-server architecture is unnecessary for this project and stand-alone architecture is chosen for the project.

Stand-alone architecture has also some disadvantages. Storing all the data on Pocket PC is a problem because satellite images or maps have larger sizes than Pocket PC's internal memory. This problem can be solved by providing an extra memory card. 256 MB Secure Digital (SD) card is enough for the work to store all the satellite images, vector data and program itself.

Another disadvantage of the stand-alone architecture is that processing all the data on Pocket PC's CPU takes longer time than processing on server's CPU, because Pocket PC CPUs' are working at very low speeds so work longer. Nowadays, most servers have 2-3 GHz, even dual, CPUs, but Pocket PCs have 200-600 MHz CPUs. This difficulty can be overcome by good programming design and well coded libraries.

- Selection of Image Format :

Some parameters have to be taken into account to choose the image formats which are image compression ratio and open source programming library.

According to image compression ration, JPEG format seems to be the best solution, but all of the image data must be dumped to the memory for calculations. Loading the whole image into memory of Pocket PC is not recommended. Pocket PC always hanged up when a JPEG image, having more than 500 KB size, was loaded. This is a big problem because satellite images have not small sizes.

If the image size is large, then image can be splited into pieces and collect pieces together to make the whole image. This trick is the heart of map server applications. Maps are divided into pieces and stored separately on servers. *(*Hendrey, 2006) When a user request the map, the pieces of that area can be collected and sent it to the user. This solution can be useful when one is dealing with servers, but not with Pocket PCs, because there are lots of image pieces after splitting. There are also other image pieces for each zoom level. At the end, there are huge numbers of image pieces, which is not suitable for Pocket PCs.

Instead wavelet compression is selected for this work because satellite images can have smaller file sizes and satellite images' interesting regions can be read directly without reading the whole image.

There are three implementations of wavelet compression: JPEG 2000, MrSID and ECW. JPEG 2000 is not a good choice for GIS applications because other two formats, MrSID and

ECW, are specially designed for GIS applications by GIS organizations. MrSID is designed by LizardTech™ for GIS applications. It has almost the same properties of JPEG 2000 because it is the implementation of JPEG 2000 standard. In addition to the JPEG 2000 standard, MrSID has geocoding information itself. There is also software developer kit (SDK) for MrSID which is open to the public. The last implementation of wavelet compression is ECW (Enhanced Compressed Wavelet). It is designed by ER Mapper™ for GIS applications. It almost resembles the MrSID format and has also software developer kit (SDK), which is opened to the public.

Since ECW SDK is simple and faster than MrSID SDK, ECW format is selected for displaying earth features.

- Displaying Roads on Image :

Displaying earth features can be succeeded by ECW format, but there are also roads to show on satellite images. This can be achieved by the help of vectors. File formats in GIS are designed for different kinds of vectors, but roads can be displayed by only line type vectors. So a simple vector format can be useful and successful on Pocket PCs.

The simple vector format for roads consists of points. Points make lines and lines make the roads. So the smallest part of vector is point. This structure is very simple to build.

Points must be stored on file or database to build lines. Therefore storage of points is needed. Flat file can be used as storage for points but flat file is not suitable for road queries if there are lots of roads. The points can be stored on flat file as lines without any indexes. As a result of this fact, the whole flat file must be read at every road search. This is not a good practice for computer programming even on desktop computers.

Other choice for storing points is database. It is generally known that database is very good solution for storing tabular data. Index also can be used with databases to accelerate the queries. There is another benefit of database to the programmer, which is Structured Query Language (SQL). SQL is a standard language for programmers to create, modify, retrieve and manipulate data, so it is not necessary to write extra code to filter the data. For that reason database is the choice for storing points for the project.

- Selection of Database for Vector :

One has two popular options for database running on Windows Mobile 2003. They are Microsoft SQL Server CE and SQLite. The first one is the commercial product of Microsoft, which is the mobile version of their product SQL Server 2000. The mobile service must be installed in order to use the SQL Server CE, which slows down the Pocket PC. Running a daemon at background is not an advised behavior for Pocket PC because the system sources are limited. On the other side,

SQLite is a open source database application and it can run on every platform. Also there is no need to install any application as a daemon at background. The database file can be easily moved just by copying it. As a result of comparing these two applications, SQLite is the best choice for the project as a database application.

- Selection of Programming Language :

For the selection of programming language, as discussed in chapter 2; J2ME, .NET Compact Framework and Embedded Visual C++ are available.

The most important factor to choose the programming language is the libraries' programming language. As it is written above, ECW and SQLite have both libraries for Windows Mobile operating system and C++. So there is no discussion about the language, Embedded Visual C++ is chosen. It is also a good choice because of the power of language itself. C++ is known as a performance language on all platforms, but this also brings a huge disadvantage. Disadvantage of C++ is to deal with all the memory problems. So the programmer must be focused on the application to deal with all the memory problems.

Embedded Visual C++ Version 3.0 is selected for coding because ECW library is compiled on Embedded Visual C++ 3.0. On the other side, SQLite library is compiled with version

4.0 and uses some advantages of version 4.0, so there is no way to compile SQLite in version 3.0. They must be compiled in same version to run. Then ECW library is compiled on version 4.0 to run with SQLite library. This is the advantage of open source programming, there is no worry about to wait for organizations to compile the library to project's version.

During testing phase, bugs are found in the program. These bugs are solved by reprogramming the vector functions.

CHAPTER 4

DEVELOPMENT PROCEDURE

Programming architecture, image and vector formats, database and programming languages of this work are selected. The next step is the development of application. The development procedure is as follows;

- Analysis of ECW SDK:

ECW SDK is an open source SDK used for both compression and decompression of ECW files for C or C++. There are some limitations in SDK on ECW image size (500 MB), but this is not important for this work. There is also another limitation. The SDK for Pocket PCs are only for decompression, which is also not important because compression on Pocket PCs is useless for this work.

Decompression of image depends on CNCSRenderer class of ECW SDK. The CNCSRenderer class directly inherits from NCSFile class to provide a mechanism to render ECW imagery into a device context. NCSError class is another class in ECW SDK to handle error that returns from CNCSRenderer class. Table 4.1 and 4.2 shows the methods and properties of CNCSRenderer class used in this work respectively.

As it is seen on Table 4.1, first image is opened for reading and then the view is set in cell units. Next step is to read image into internal buffer. Finally buffered image is drawn on screen.

Table 4.1: Methods of CNCSRenderer class

Open (char * pURLPath, BOOLEAN bProgressiveDisplay)	Opens a local or remote ECW image file
SetView (INT32 nBands, INT32 * pBandList, INT32 nWidth, INT32 nHeight, INT32 nDatasetTLX, INT32 nDatasetTLY, INT32 nDatasetBRX, INT32 nDatasetBRY)	Sets the view extents of the renderer object, in dataset (cell) units.
ReadImage (INT32 nWidth, INT32 nHeight)	Reads the image into an internal buffer ready for drawing.
DrawImage(HDC hDeviceContext, LPRECT pClipRect, IEEE8 dWorldTLX, IEEE8 dWorldTLY, IEEE8 dWorldBRX, IEEE8 dWorldBRY)	Draws the ECW image view on the screen, using the given extents.

Table 4.2: Properties of CNCSRenderer class

m_nNumberOfBands	The number of bands in the image.
m_bIsProgressive	Set to TRUE if the view is progressive.

- Analysis of SQLite CE SDK

SQLite SDK is an open source SDK used for database connection for C or C++. SQLite does not need daemon or service to connect database. One single file on the disk is just needed in order to prepare a database.

Database connection depends on CDbSQLite and CSqlStatement classes of SQLite CE SDK. CDbSQLite class read the database file and prepares the connection. CSqlStatement class reads and shows the query results. Table 4.3 and 4.4 shows the methods of CDbSQLite and CSqlStatement classes used in this work respectively.

As it is seen on Table 4.3 and Table 4.4, first database file is opened for reading and then road query is executed by CDbSQLite class. Finally, results are shown by CSqlStatement class.

Table 4.3: Methods of CDbSQLite class

Open (char * pURLPath)	Opens a local SQLite database file.
Statement (char * pSQLQuery)	Executes the query and returns statement object.

Table 4.4: Methods of CSqlStatement class

NextRow()	Returns TRUE if there is a next row.
ValueString(INT32 colonID)	Gets the value of colon.

- Design of own vector class

Designing own vector class is not an easy task. There are lots of things to think about. Points or lines must be derived from the database, and then they must be processed in order to show on screen. The functions of vector class can be seen on Table 4.5.

The important point in this vector class is to find the points which are in the frame. First all the road points are taken from database and passed to the vector functions. Vector functions go through all the points to check whether they are in the frame or not.

All the coordinates are stored as pixel coordinates in the database in order to accelerate the process.

Table 4.5: Methods of vector class

OnButtonSearch()	Gets the road points from database and prepare the road point array.
OnPaint()	Shows raster and vector on screen.
PrepareFramePoint()	Gets the points in the frame.
IsPointInFrame(int pointID)	Determines the location of a point, inside or outside the frame.
prepareRoad(int pointID, int type)	Prepares road slices by connecting the points.
FindExternalPoint(POINT p1, POINT p2);	Finds the intersection of frame and road slice which is outside the frame.
FindPointIntersection(POINT p1, POINT p2, POINT p3, POINT p4)	Finds the intersection of two lines.

- Combining all the classes

Combining all the classes is an easy task after analyzing ECW and SQLite SDK and designing vector classes. Embedded Visual C++ is an event-driven programming language; every object has an event to fire up. For example, when a button is clicked, a function associated with the button is called.

The first step of combining is to prepare a graphical user interface (GUI). All the steps of graphical user interface design are shown in Figure 4.1, 4.2, 4.3 and 4.4.

After GUI design, all the events are associated with the buttons, menu, toolbar and forms. The code generated during this period can be seen on Figure 4.5.

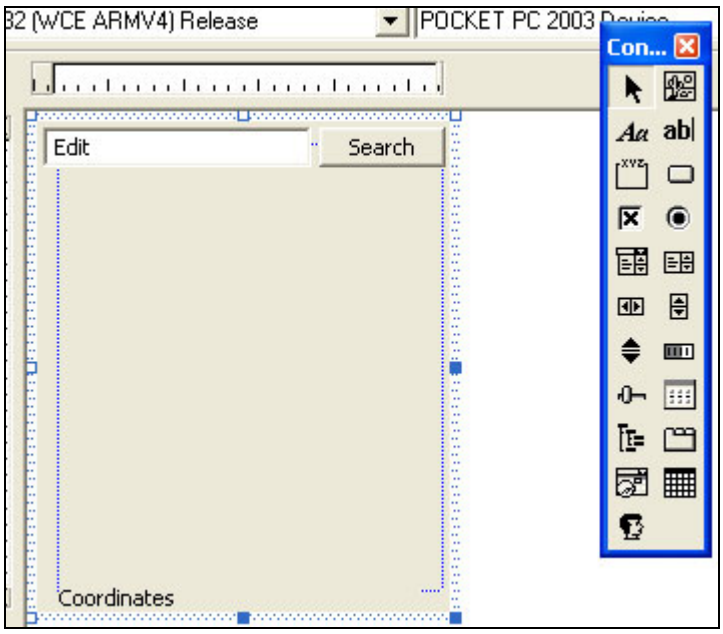

Figure 4.1: Screenshot of GUI design process – Main Form.

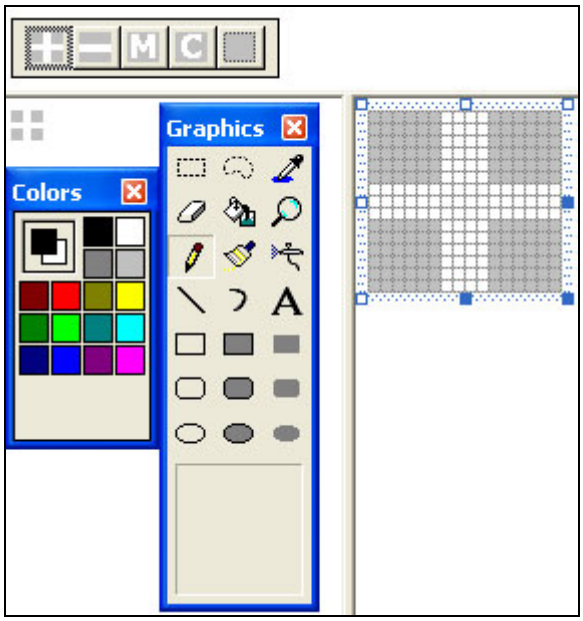

Figure 4.2: Screenshot of GUI design process - Toolbar.

Figure 4.3: Screenshot of GUI design process – About Form.

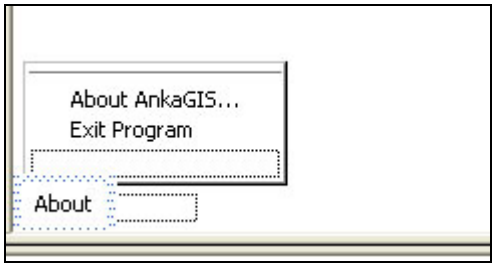

Figure 4.4: Screenshot of GUI design process – Menu.

```
BEGIN_MESSAGE_MAP(CAnkaGISView, CFormView)
    //{{AFX_MSG_MAP(CAnkaGISView)
    ON_COMMAND(ID_BUTTON1, OnButtonZoomIn)
    ON_COMMAND(ID_BUTTON2, OnButtonZoomOut)
    ON_COMMAND(ID_BUTTON3, OnButtonMove)
    ON_COMMAND(ID_BUTTON4, OnButtonCoordinates)
    ON_WM_PAINT()
    ON_WM_LBUTTONDOWN()
    ON_WM_LBUTTONUP()
    ON_BN_CLICKED(IDC_BUTTON_SEARCH, OnButtonSearch)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

Figure 4.5: Screenshot of associating of events with functions.

CHAPTER 5

ENVIRONMENT, INSTALLATION AND USAGE OF PROGRAM

Developing a mobile application is not an easy task as developing desktop application. Mobile application is coded on desktop, compiled for Pocket PC platform and then transferred to the Pocket PC. These steps are explained below.

5.1. Environment

Environment consists of ECW SDK, SQLite SDK and Embedded Visual C++ 4.0 IDE.

5.1.1. ECW SDK

ECW SDK is an open source library of a GIS company, ER Mapper who is the leader in the development and deployment of patented geospatial imagery technologies throughout the world. This Open Source library adds compatibility for large volumes of the ECW and JPEG 2000 standards image data to GIS applications. (ER Mapper 2006) The screenshot of setup screen can be seen on Figure 5.1 and also the properties of ECW SDK can be seen as follows (ER Mapper, 2006);

- Lossless and lossy compression
- 64 bit file support handles TB+ size images

- Support for NITF, NPJE, EPJE
- Widely used by GIS, CAD, imaging and office applications
- Simple no-hassle licensing
- Fast viewing at any resolution for any region
- Local and ECWP streaming image access
- Free, Open Source and commercial licensing available
- Free unlimited compression use in GPL style software
- Free unlimited compression use in GPL style software
- Free 500MB compression for commercial applications
- Royalty free unlimited compression for commercial apps
- 64 bit OS support
- Low memory footprint even on TB size images
- Supports many operating systems with no artificial limits
- Geolocation data preserved as embedded metadata

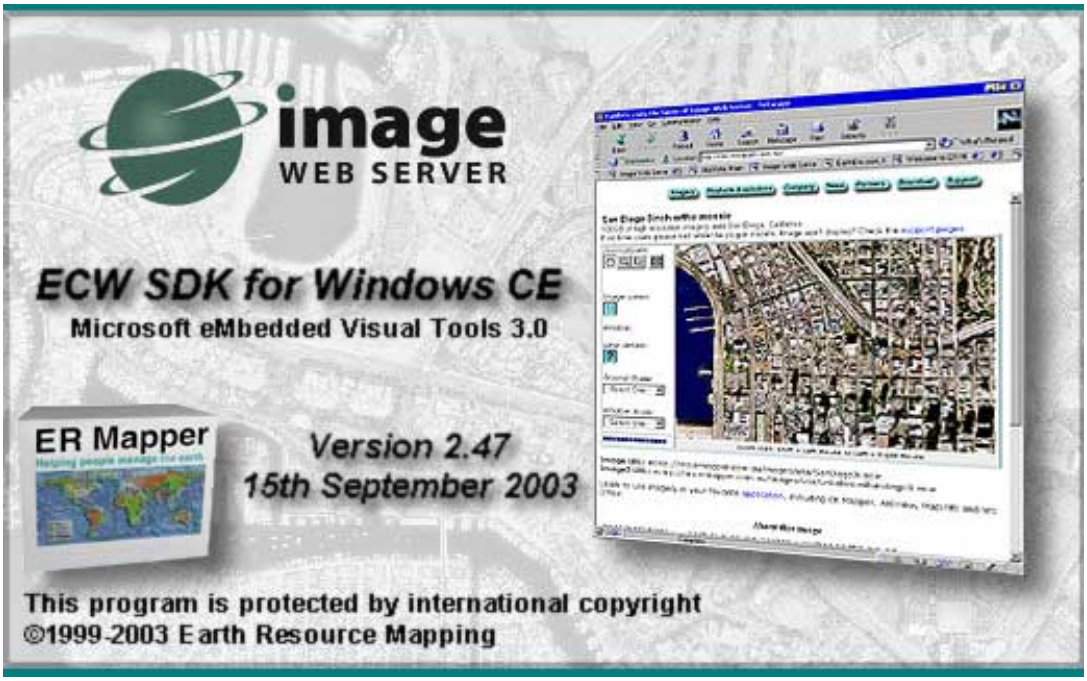

Figure 5.1: Installation screenshot of ECW SDK for Windows CE

5.1.2. SQLite SDK

SQLite is a small C library that implements a self-contained, embeddable, zero-configuration SQL database engine. SQLite is an open source project, which claims that it is different than other databases engines. (SQLite, 2006)

SQLite has some distinct characteristics as follows:

- Zero-Configuration
- Serverless
- Single Database File
- Compact
- Manifest typing
- Variable-length records
- Readable source code
- SQL statements compile into virtual machine code
- Public domain
- SQL language extensions

5.1.3. Embedded Visual C++ 4.0 IDE

The Microsoft Embedded Visual C++ 4.0 tool is a desktop development environment for creating applications and system components for Windows CE-powered devices. This version features new capabilities such as C++ exception handling, Run Time Type Information (RTTI), and a plethora of new debugger functionalities. (Microsoft 2003)

Installing Embedded Visual C++ 4.0 is not enough for developing applications for Windows Mobile 2003 devices. Service Pack 4 for Embedded Visual C++ 4.0 and Pocket PC 2003 SDK must be installed before developing a mobile application for Windows Mobile 2003 operating system. The screenshot of Embedded Visual C++ 4.0 can be seen on Figure 5.2.

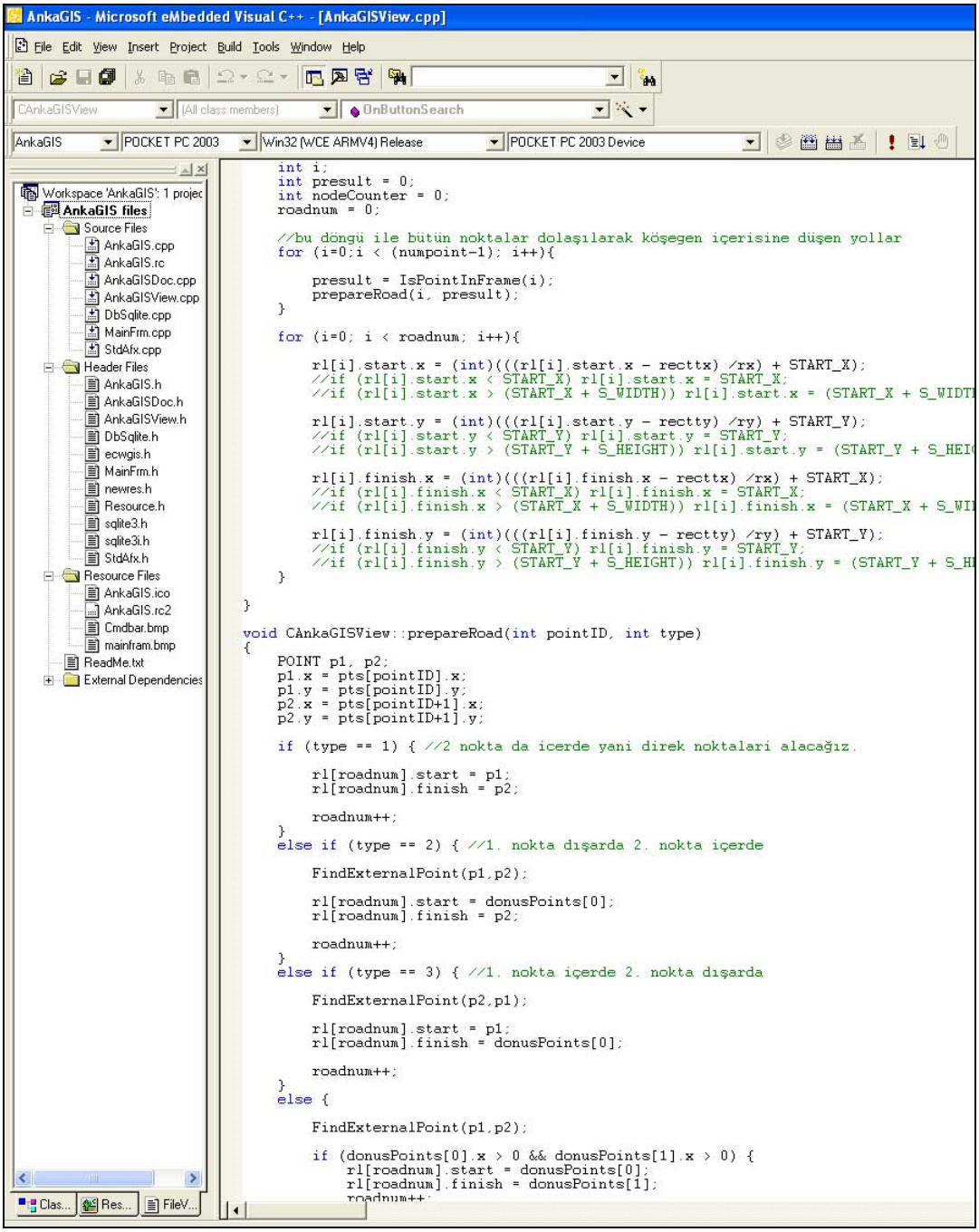

Figure 5.2: Screenshot of Embedded Visual C++ 4.0 IDE

5.2. Installation

Compiling the mobile application is not the final step to deploy the program. There are some steps to run compiled program on Pocket PC.

- First step is to install the ECW SDK redistributables. The files are found as :

C:\Program Files\Earth Resource Mapping\ECW CE SDK\redistributables

- As it is shown on Figure 5.3, there are different types of packages. The package that is suitable for Windows Mobile 2003 SE is the "Pocket PC 2002 ECWSDK Dlls.ARM.CAB" CAB files are the executable installation files for Windows Mobile Devices. There is also ARM indicator, which shows the CPU type of Pocket PC.

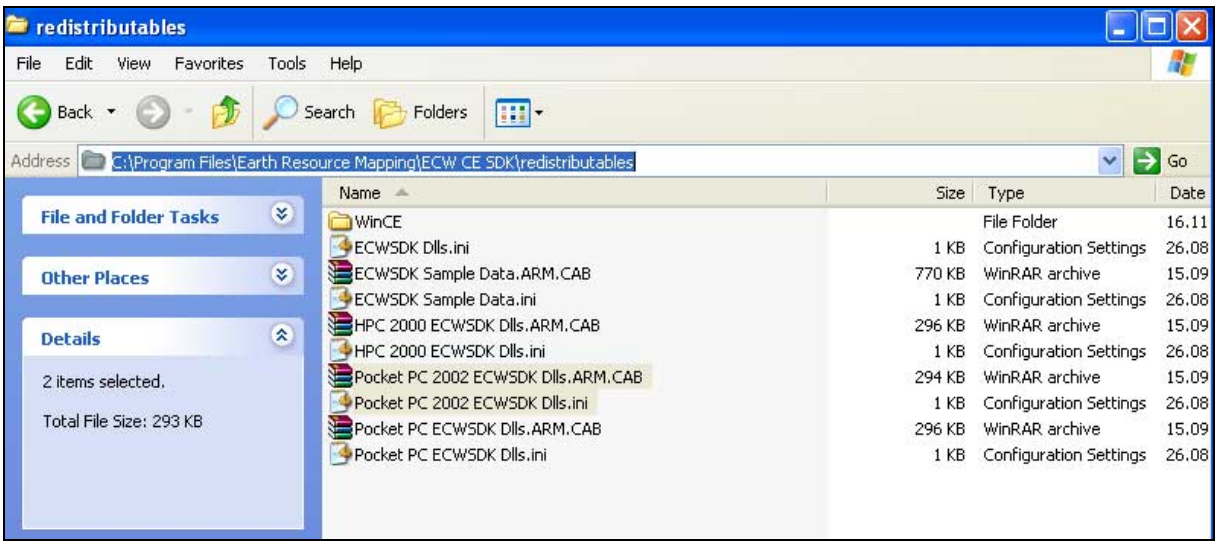

Figure 5.3: Screenshot of ECW SDK redistributables directory

- The following files must be copied into Pocket PC and must be installed : *Pocket PC 2002 ECWSDK Dlls.ARM.CAB* and *Pocket PC 2002 ECWSDK Dlls.ini.* The installation procedure can be seen on Figure 5.4 and Figure 5.5.

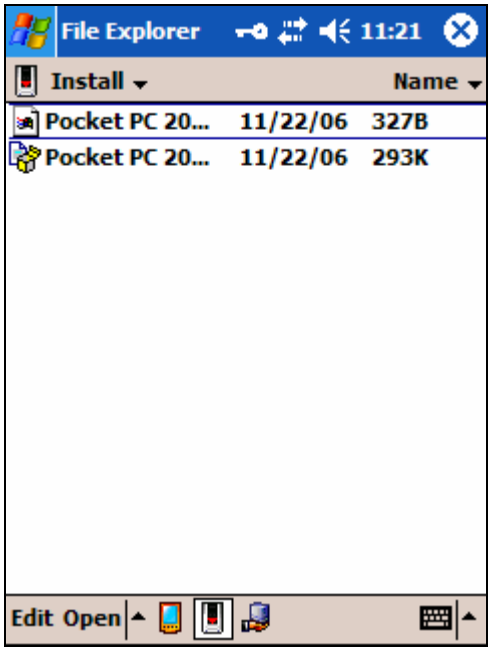

Figure 5.4: Screenshot of ECW SDK redistributables on Pocket PC

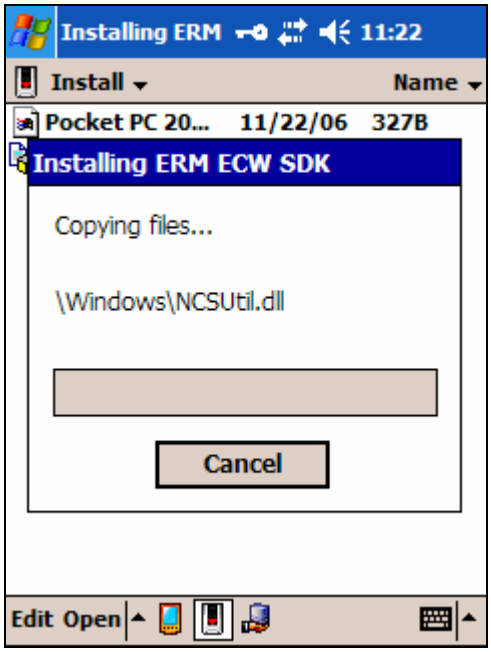

Figure 5.5: Screenshot of installation of ECW SDK on Pocket PC

- This installation is valid if the application is compatible with Embedded Visual C++ 3.0. So the new DLLs that are compiled with Embedded Visual C++ 4.0 must be overwritten on Pocket PC.

- *NCScnet.dll, NCSEcw.dll* and *NCSUtil.dll* are the files that are compiled with Embedded Visual C++ 4.0 must be overwritten on "*Windows"* directory of Pocket PC.

- Next step is to copy the application compiled to the Pocket PC. The application directory is named as "*AnkaGIS*" is copied to the SD Card on Pocket PC.

- The last step is the copying of SQLite libraries to the application directory. *wceSQLite3.dll* and *wceSQLite3.lib* files are copied to the "*AnkaGIS*" directory.

5.3. Usage of Program

Usage of AnkaGIS is an easy task even for a novice user. AnkaGIS includes basic GIS functions, such as zoom in, zoom out, move, get coordinates and search and usage of these functions are most like other GIS applications' functions.

Starting program is just clicking the "AnkaGIS" executable on Figure 5.6.

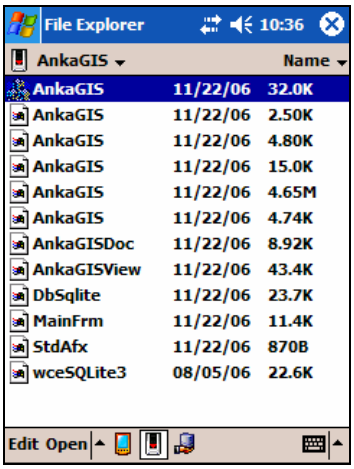

Figure 5.6: Screenshot of AnkaGIS directory

The initial screen of application is shown on Figure 5.7.

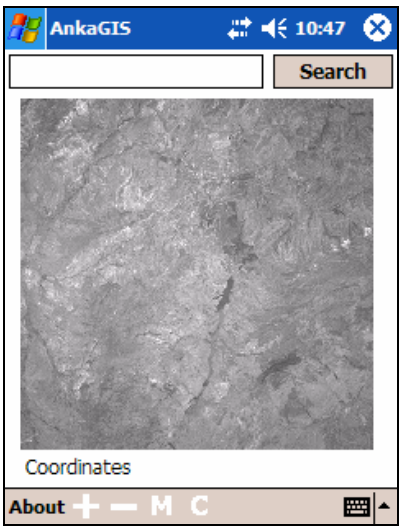

Figure 5.7: Screenshot of AnkaGIS application startup.

As it is seen on Figure 4.7, all the functions are clearly shown on the application screen.

- "+" button for zoom in
- "-" button for zoom out
- "M" button for move
- "C" button for getting coordinates in UTM standard.
- "About" menu has two options: exit program and give program information
- "Search" button search road index if the textbox nearby is filled
- "Coordinates" text is used for giving coordinate information

The output of zoom in function of a program can be shown on Figure 5.8. Mogan Lake in the image can be the focus area of an application. This image can be taken by clicking the "+" button and then clicking the lake on screen by three times.

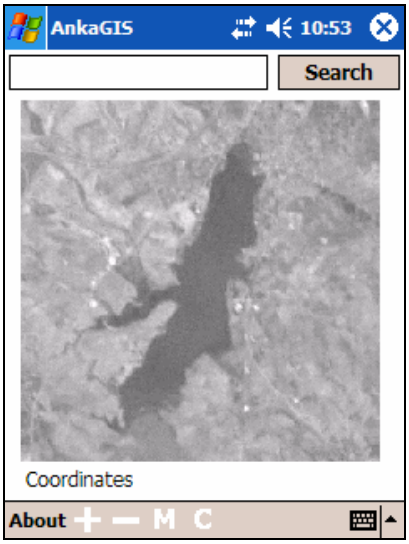

Figure 5.8: Screenshot of AnkaGIS application on zoom in.

Searching function can be used by the help of virtual keyboard, which can be activated by clicking the small icon at right bottom. The virtual keyboard and some text for searching can be seen on Figure 5.9.

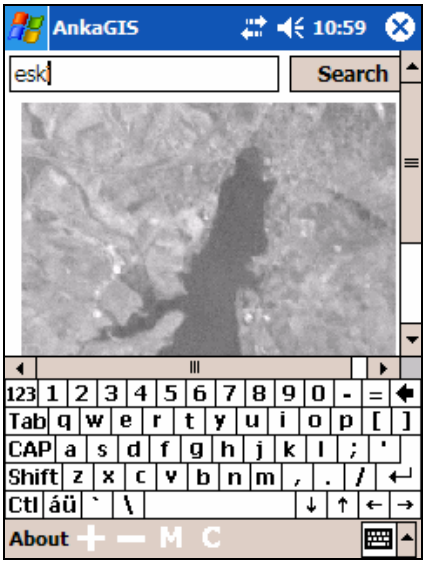

Figure 5.9: Screenshot of AnkaGIS application with virtual keyboard.

Searching a road is just typing the road name and clicking the search button. The result of this action can be seen on Figure 5.10. "*Eskişehir Road*" is searched as an example by writing "*eski*". Result can be seen as a yellow polyline on the image. Zoom in and out functions can also be used with search option, which can be seen on Figure 5.11.

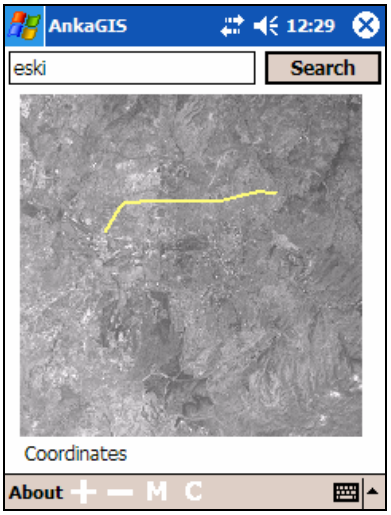

Figure 5.10: Screenshot of AnkaGIS application with search result, Eskişehir Road.

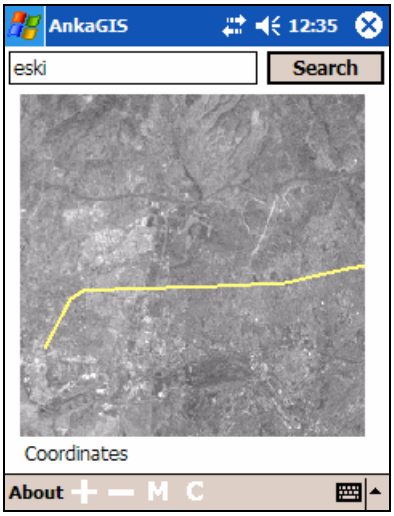

Figure 5.11: Screenshot of AnkaGIS application with zoom in on search result, Eskişehir Road.

Another GIS function that is shown above is getting coordinates. Coordinates of a point on the image can easily be got by "C" button. The demonstration of getting coordinates can be seen on Figure 5.12. The coordinates are in term of UTM format, Easting and Northing.

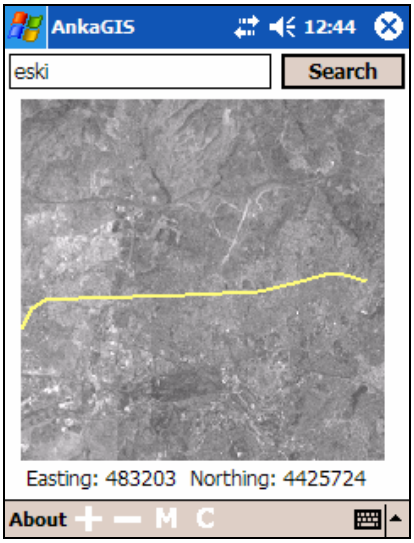

Figure 5.12: Screenshot of AnkaGIS application with getting coordinates option on search result, Eskişehir Road.

CHAPTER 6

CONCLUSIONS AND DISCUSSIONS

The aim of this work is to prepare search tool for roads for mobile GIS application, which is successfully achieved. As it is stated in previous chapter, there are lots of products that make the same work on PDA, but they are commercial products. It is obvious that commercial products are more comprehensive then the work done in this thesis, but generating such a guide can show that a mobile GIS can be provided without any commercial software. All the libraries and development environments used in this work are free to public access without limitations for public and academic purposes.

Some additional features can be added in the future. There are no limitations on the programming side.

One of the additional features can be a GPS support, which is quite easy to implement if there are any experienced programmer. Additional module can be written to listen the COM port to get GPS data and then show it on the map. By the help of this module, the program can be used as a guide to find nearest road.

Another additional feature can be POI support. POI is the abbreviation of Point of Interest. Historical places, shopping centers, hospitals, schools, religious places, parks and governmental places can be POI. There can be an option to store POIs on database and can be showed on a map.

Another suggestion for other programmers is to compile the program with a little effort on other platforms such as Palm, Java or Symbian because the program compiled on this work is able to work only on Windows Mobile operating system

REFERENCES

1. P. Clegg, L. Bruciatelli, F. Domingos, R.R. Jones, M. De Donatis and R.W. Wilson "Digital geological mapping with tablet PC and PDA: A comparison". *Computers & Geosciences* In press

2. Majid Rabbani, Rajan Joshi (2002) "An overview of the JPEG2000 still image compression standard" *Signal Processing: Image Communication* Vol.17 pp. 3–48

3. Worboys Michael F. and Duckham Matt. 2004. *GIS, a computing perspective.* Taylor & Francis

4. Casademont, Jordi. 2004. Wireless technology applied to GIS. Computers and Geosciences 30 : 671-682

5. Wikipedia: The Free Encyclopedia. 2006. Image File Formats. Available From : http://en.wikipedia.org/wiki/Image_file_formats

6. Wikipedia: The Free Encyclopedia. 2006. Wavelet Compression. Available From : http://en.wikipedia.org/wiki/Wavelet_compression

7. Hassin, Bryan. (2003 July) Mobile GIS: How to Get There From Here. ESRI User Conference 2003, San Diego, California, USA.

8. Yuan, Michael Juntao. 2003. Enterprise J2ME™: Developing Mobile Java™ Applications. Prentice Hall.

9. SUN. 2005. J2ME Reference. Available From : http://java.sun.com/javame/reference/apis.jsp

10. Fox, Dan and Box, Jon. 2003. Building Solutions with the Microsoft .NET Compact Framework: Architecture and Best Practices for Mobile Development. Addison Wesley Professional.

11. Microsoft. 2005. .Net Compact Framework Reference. Available From : http://msdn2.microsoft.com/en-us/netframework/aa497279.aspx

12. Grattan, Nick and Brain, Marshall. 2000. Windows® CE 3.0 Application Programming. Prentice Hall.

13. Yao, Paul. (2004 November) Programming With eMbedded Visual C++ 4.0. Microsoft MDC 2004, India.

14. Hendrey, Geoffrey. 2006. deCarta's Open Architecture for AJAX Draggable Maps. deCarta. Available From : http://www.decarta.com/products/dds/DDS_Web_Services_AJAX_Whitepaper.pdf

15. Kramer, Herbert J. 2002. Observation of the Earth and Its Environment: Survey of Missions and Sensors. Springer-Verlag.

16. ESRI. 2006. ArcPad – Mobile GIS Software for Field Mapping Applications. Available From : http://www.esri.com/software/arcgis/arcpad/index.html

17. MapInfo. 2006. MapX Mobile. Available From : http://extranet.mapinfo.com/products/Overview.cfm?ProductID=1661

18. ER Mapper. 2006. ECW JPEG 2000 SDK 3.1. Available From: http://www.ermapper.com/ecw/

19. SQLite. 2006. SQLite SDK. Available From : http://www.sqlite.org/

20. Microsoft. 2003. Embedded Visual C++ 4.0 IDE. Available From : http://www.microsoft.com/downloads/details.aspx?familyid=1DACDB3D-50D1-41B2-A107-FA75AE960856&displaylang=en

21. Miano, John. 1999. Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP. Addison-Wesley Professional.

APPENDIX A

SOURCE CODE

```
// AnkaGISView.cpp : implementation of the CAnkaGISView class
//

#include "stdafx.h"
#include "AnkaGIS.h"
//#include "math.h"

#include "AnkaGISDoc.h"
#include "AnkaGISView.h"

#include "NCSRenderer.h"
#include "DbSQLite.h"
#include "ecwgis.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAnkaGISView

IMPLEMENT_DYNCREATE(CAnkaGISView, CFormView)

BEGIN_MESSAGE_MAP(CAnkaGISView, CFormView)
	//{{AFX_MSG_MAP(CAnkaGISView)
	ON_COMMAND(ID_BUTTON1, OnButtonZoomIn)
	ON_COMMAND(ID_BUTTON2, OnButtonZoomOut)
	ON_COMMAND(ID_BUTTON3, OnButtonMove)
	ON_COMMAND(ID_BUTTON4, OnButtonCoordinates)
	ON_WM_PAINT()
	ON_WM_LBUTTONDOWN()
	ON_WM_LBUTTONUP()
	ON_BN_CLICKED(IDC_BUTTON_SEARCH, OnButtonSearch)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CAnkaGISView construction/destruction

CAnkaGISView::CAnkaGISView()
	: CFormView(CAnkaGISView::IDD)
{
	zoom = 2;
```

```
wx = IMG_WIDTH;
wy = IMG_HEIGHT;
wpx = 0.0;
wpy = 0.0;
px = 0;
py = 0;
spx = 0;
spy = 0;
iniratiox = (float)(IMG_WIDTH / S_WIDTH);
iniratioy =     (float) (IMG_HEIGHT / S_HEIGHT);
temp_width = (float) IMG_WIDTH;
temp_height = (float) IMG_HEIGHT;
rx = (temp_width / S_WIDTH);
ry = (temp_height / S_HEIGHT);
buttonFunction = 0;
searchdone = false;
numpoint = 0;
roadnum = 0;

testnum = 0;

lp.lopnStyle = PS_SOLID;
lp.lopnWidth.x = 2;
lp.lopnWidth.y = 2;
lp.lopnColor = RGB (248, 249, 122);


llparam1 = 29.5100471702142;
llparam2 = 3.18512103847722;
llparam3 = 443479.596261279;
llparam4 = 3.46765511992251;
llparam5 = -28.7601568987722;
llparam6 = 4430327.80329682;

recttx = 0;
rectty = 0;
rectbx = IMG_WIDTH - 1;
rectby = IMG_HEIGHT - 1;

m_Rect.bottom = START_Y + S_HEIGHT;
m_Rect.left = START_X;
m_Rect.right = START_X + S_WIDTH;
m_Rect.top = START_Y;


m_bIsProgressive = FALSE;
rend1 =  new CNCSRenderer();
eError = rend1->Open("SD CARD\\ankara3.ecw", m_bIsProgressive);
INT32 i;
for(i = 0; i < rend1->m_nNumberOfBands; i++)
        BandsArray[i] = i;

CString filename = "SD CARD\\test1.db";
sqlTest = sqlite.Open(LPCTSTR(filename));
if ( !sqlTest )
```

```
	{
		MessageBox(LPCTSTR("Database Connection
Problem!"),NULL,MB_OK);
	}


}

CAnkaGISView::~CAnkaGISView()
{
}

void CAnkaGISView::DoDataExchange(CDataExchange* pDX)
{
	CFormView::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAnkaGISView)
		// NOTE: the ClassWizard will add DDX and DDV calls here
	//}}AFX_DATA_MAP
}

BOOL CAnkaGISView::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: Modify the Window class or styles here by modifying
	//  the CREATESTRUCT cs

	return CFormView::PreCreateWindow(cs);
}

/////////////////////////////////////////////////////////////////////////////
// CAnkaGISView diagnostics

#ifdef _DEBUG
void CAnkaGISView::AssertValid() const
{
	CFormView::AssertValid();
}

void CAnkaGISView::Dump(CDumpContext& dc) const
{
	CFormView::Dump(dc);
}

CAnkaGISDoc* CAnkaGISView::GetDocument() // non-debug version is inline
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CAnkaGISDoc)));
	return (CAnkaGISDoc*)m_pDocument;
}
#endif //_DEBUG


void CAnkaGISView::OnPaint()
{
	CPaintDC dc(this); // device context for painting

	hPen = CreatePenIndirect (&lp);
```

```
	hOldPen = (HPEN)SelectObject (dc, hPen);

	eError = rend1->SetView(rend1->m_nNumberOfBands, BandsArray,
						S_WIDTH, S_HEIGHT,
						recttx, rectty,
						rectbx, rectby);

	eError = rend1->ReadImage(S_WIDTH, S_HEIGHT);
	eError = rend1-
>DrawImage(dc,&m_Rect,START_X,START_Y,(START_X+S_WIDTH),(START_
Y+S_HEIGHT));


	if (searchdone == true) {

		testnum++;


		PrepareFramePoint();
		for (int i=0; i < roadnum; i++){
			tempRoad[0] = rl[i].start;
			tempRoad[1] = rl[i].finish;
			Polyline(dc,tempRoad,2);
		}
	}

	SelectObject (dc, hOldPen);
	DeleteObject (hPen);
}


/////////////////////////////////////////////////////////////////////////////
// CAnkaGISView message handlers

void CAnkaGISView::OnButtonZoomIn()
{
	buttonFunction = 1;
}

void CAnkaGISView::OnButtonZoomOut()
{
	buttonFunction = 2;
}

void CAnkaGISView::OnButtonMove()
{
	buttonFunction = 3;
}

void CAnkaGISView::OnButtonCoordinates()
{
	buttonFunction = 4;
}
```

```
CAnkaGISView::ChangeMapView()
{

	rx = (temp_width / S_WIDTH);
	ry = (temp_height / S_HEIGHT);

	spx = px - START_X;
	spy = py - START_Y;

	wpx = (float) recttx + (spx * rx);
	wpy = (float) rectty + (spy * ry);

	if (buttonFunction == 2) {
		rx = (rx * zoom);
		ry = (ry * zoom);
	}

	if (buttonFunction == 1) {
		rx = (rx / zoom);
		ry = (ry / zoom);
	}

	if ( ((rx < (iniratiox + 1.8)) || (ry < (iniratioy + 1.8))) && ((rx > 1.0)
|| (ry > 1.0)) ) {

		temp_width = (rx * S_WIDTH);
		temp_height = (ry * S_HEIGHT);

		recttx = (int) (wpx - (temp_width / 2));
		rectty = (int) (wpy - (temp_height / 2));

		rectbx = (int) (wpx + (temp_width / 2));
		rectby = (int) (wpy + (temp_height / 2));


		if (rectbx > (IMG_WIDTH - 1)) {
			rectbx = (IMG_WIDTH - 1);
			recttx = rectbx - temp_width;
			wpx = (int) (rectbx - (temp_width / 2));
		}

		if (recttx < 0) {
			recttx = 0;
			rectbx = recttx + temp_width;
			wpx = (int) (rectbx - (temp_width / 2));
		}

		if (rectby > (IMG_HEIGHT - 1)) {
			rectby = (IMG_HEIGHT - 1);
			rectty = rectby - temp_height;
			wpy = (int) (rectby - (temp_height / 2));
		}

		if (rectty < 0) {
			rectty = 0;
```

```
                rectby = 0 + temp_height;
                wpy = (int) (rectby - (temp_height / 2));
            }


            if (rx < 1.0) rx = (float) 1.01;
            if (ry < 1.0) ry = (float) 1.01;


        }
        else {
            if (buttonFunction == 2) {
                rx = (rx / zoom);
                ry = (ry / zoom);
            }

            if (buttonFunction == 1) {
                rx = (rx * zoom);
                ry = (ry * zoom);
            }
        }

}

void CAnkaGISView::OnLButtonDown(UINT nFlags, CPoint point)
{
        px = int(point.x);
        py = int(point.y);
        spx = 0;
        spy = 0;

        if ((px > START_X) && (px < (START_X + S_WIDTH))){
            if ((py > START_Y) && (py < (START_Y + S_HEIGHT))){

                if (buttonFunction == 1 || buttonFunction == 2) {
                    ChangeMapView();
                    Invalidate(TRUE);
                }
                else if (buttonFunction == 3) {
                    spx = px - START_X;
                    spy = py - START_Y;
                }
                else if (buttonFunction == 4) {
                    showCoordinates();
                }
            }
        }


        CFormView::OnLButtonDown(nFlags, point);
}

void CAnkaGISView::OnLButtonUp(UINT nFlags, CPoint point)
{
        if(buttonFunction == 3){
```

```
			int px2 = int(point.x);
			int py2 = int(point.y);
			int spx2 = px2 - START_X;
			int spy2 = py2 - START_Y;

			totx = spx2 - spx;
			toty = spy2 - spy;

			wpx = (float) wpx - (totx * rx);
			wpy = (float) wpy - (toty * ry);

			recttx = (int) (wpx - (temp_width / 2));
			rectty = (int) (wpy - (temp_height / 2));

			rectbx = (int) (wpx + (temp_width / 2));
			rectby = (int) (wpy + (temp_height / 2));

			if (rectbx > (IMG_WIDTH - 1)) {
				rectbx = (IMG_WIDTH - 1);
				recttx = rectbx - temp_width;
				wpx = (int) (rectbx - (temp_width / 2));
			}

			if (recttx < 0) {
				recttx = 0;
				rectbx = recttx + temp_width;
				wpx = (int) (rectbx - (temp_width / 2));
			}

			if (rectby > (IMG_HEIGHT - 1)) {
				rectby = (IMG_HEIGHT - 1);
				rectty = rectby - temp_height;
				wpy = (int) (rectby - (temp_height / 2));
			}

			if (rectty < 0) {
				rectty = 0;
				rectby = 0 + temp_height;
				wpy = (int) (rectby - (temp_height / 2));
			}

			Invalidate(TRUE);
		}

		CFormView::OnLButtonUp(nFlags, point);
}

void CAnkaGISView::showCoordinates()
{
		double stx, sty;
		stx = 0.0;
		sty = 0.0;

		rx = (temp_width / S_WIDTH);
```

```
	ry = (temp_height / S_HEIGHT);

	spx = px - START_X;
	spy = py - START_Y;

	wpx = (float) recttx + (spx * rx);
	wpy = (float) rectty + (spy * ry);

	stx = (llparam1 * wpx) + (llparam2 * wpy) + llparam3;
	sty = (llparam4 * wpx) + (llparam5 * wpy) + llparam6;

	CString str;
	str = "";
	str.Format(_T("Easting: %.0f  Northing: %.0f"), stx, sty);

	SetDlgItemText(IDC_XYCOORD,str);
}

void CAnkaGISView::OnButtonSearch()
{



   CEdit* myEdit3 = (CEdit*) GetDlgItem(IDC_EDIT_BOX);
   CString text;
   myEdit3->GetWindowText(text);

	CString str;
	str = "SELECT x,y FROM roads WHERE name LIKE '%" + text + "%'
ORDER BY rid";
	stmt = sqlite.Statement(str);

	roadnum = 0;
	numpoint = 0;

	if ( stmt != NULL )
	{
		while ( stmt->NextRow() )
		{
			pts[numpoint].x = (int) (_wtoi(stmt->ValueString(0)));
			pts[numpoint].y = (int) (_wtoi(stmt->ValueString(1)));
			numpoint++;
		}
	}

	if (numpoint == 0) {
		MessageBox(_T("There is no such named road!"),NULL
,MB_OK);

	}
	else {
		searchdone = true;

		int tdx, tdy;
```

```
tdx = abs(pts[(numpoint-1)].x - pts[0].x);
tdy = abs(pts[(numpoint-1)].y - pts[0].y);

if (tdx > tdy) selected = tdx;
else selected = tdy;

wpx = (int) ((pts[0].x + pts[(numpoint-1)].x) / 2);
wpy = (int) ((pts[0].y + pts[(numpoint-1)].y) / 2);


if (selected >= 1024 && selected < 2048) {

        rx = (float) 9.53;
        ry = (float) 9.53;
        temp_width = (float) IMG_WIDTH;
        temp_height = (float) IMG_HEIGHT;
}

else if (selected >= 512 && selected < 1024) {

        rx = (float) (9.53 / 2);
        ry = (float) (9.53 / 2);
        temp_width = (float) (IMG_WIDTH / 2);
        temp_height = (float) (IMG_HEIGHT / 2);

}
else if (selected >= 256 && selected < 512) {

        rx = (float) (9.53 / 4);
        ry = (float) (9.53 / 4);
        temp_width = (float) (IMG_WIDTH / 4);
        temp_height = (float) (IMG_HEIGHT / 4);
}
else if (selected >= 128 && selected < 256) {

        rx = (float) (9.53 / 8);
        ry = (float) (9.53 / 8);
        temp_width = (float) (IMG_WIDTH / 8);
        temp_height = (float) (IMG_HEIGHT / 8);
}

recttx = (int) (wpx - (temp_width / 2));
rectty = (int) (wpy - (temp_height / 2));

rectbx = (int) (wpx + (temp_width / 2));
rectby = (int) (wpy + (temp_height / 2));

if (rectbx > (IMG_WIDTH - 1)) {
        rectbx = (IMG_WIDTH - 1);
        recttx = rectbx - temp_width;
        wpx = (int) (rectbx - (temp_width / 2));
}

if (recttx < 0) {
        recttx = 0;
```

```
                rectbx = recttx + temp_width;
                wpx = (int) (rectbx - (temp_width / 2));
            }

            if (rectby > (IMG_HEIGHT - 1)) {
                rectby = (IMG_HEIGHT - 1);
                rectty = rectby - temp_height;
                wpy = (int) (rectby - (temp_height / 2));
            }

            if (rectty < 0) {
                rectty = 0;
                rectby = 0 + temp_height;
                wpy = (int) (rectby - (temp_height / 2));
            }



            Invalidate(TRUE);

        }

}


POINT CAnkaGISView::FindPointIntersection(POINT p1, POINT p2, POINT p3,
POINT p4)
{
        POINT fp;

        float ua_t, ub_t, u_b;
        ua_t = (p4.x - p3.x) * (p1.y - p3.y) - (p4.y - p3.y) * (p1.x - p3.x);
        ub_t = (p2.x - p1.x) * (p1.y - p3.y) - (p2.y - p1.y) * (p1.x - p3.x);
        u_b  = (p4.y - p3.y) * (p2.x - p1.x) - (p4.x - p3.x) * (p2.y - p1.y);

        if ( u_b != 0 ) {
                float ua = ua_t / u_b;
                float ub = ub_t / u_b;

                if ( 0 <= ua && ua <= 1 && 0 <= ub && ub <= 1 ) {
                        fp.x = p1.x + ua * (p2.x - p1.x);
                        fp.y = p1.y + ua * (p2.y - p1.y);
                } else {
                        fp.x = -1;
                        fp.y = -1;
                }
        }
        else {
      if ( ua_t == 0 || ub_t == 0 ) {
         fp.x = -1;
                        fp.y = -1;
      }
                else {
         fp.x = -1;
                        fp.y = -1;
```

```
      }
   }

      return fp;
}



void CAnkaGISView::FindExternalPoint(POINT p1, POINT p2)
{
      POINT rectPoints[4];
      POINT tempPoint;

      int count = 0;

      rectPoints[0].x = recttx;
      rectPoints[0].y = rectty;
      rectPoints[1].x = recttx + temp_width;
      rectPoints[1].y = rectty;
      rectPoints[2].x = recttx + temp_width;
      rectPoints[2].y = rectty + temp_height;
      rectPoints[3].x = recttx;
      rectPoints[3].y = rectty + temp_height;

      int i,j;
      i = j = 0;


      for (i = 0; i < 4; i++){
            j = i + 1;
            if (j == 4) j = 0;

                  tempPoint = FindPointIntersection(rectPoints[i],
rectPoints[j], p1, p2);
                  if (tempPoint.x != -1){

                        donusPoints[count] = tempPoint;
                        count++;
            }
      }
}


int CAnkaGISView::IsPointInFrame(int pointID)
{

      boolean firstPoint, secondPoint;
      int birinci = pointID;
      int ikinci = birinci + 1;

      if (((recttx < pts[birinci].x) && (rectbx > pts[birinci].x)) && ((rectty <
pts[birinci].y) && (rectby > pts[birinci].y))){
            firstPoint = true;
      }
      else firstPoint = false;
```

```
	if (((recttx < pts[ikinci].x) && (rectbx > pts[ikinci].x)) && ((rectty <
pts[ikinci].y) && (rectby > pts[ikinci].y))){
		secondPoint = true;
	}
	else secondPoint = false;

	if (firstPoint && secondPoint) return 1;
	else if (!firstPoint && secondPoint) return 2;
	else if (firstPoint && !secondPoint) return 3;
	else return 0;
}

void CAnkaGISView::PrepareFramePoint()
{
	int i;
	int presult = 0;
	int nodeCounter = 0;
	roadnum = 0;

	for (i=0;i < (numpoint-1); i++){

		presult = IsPointInFrame(i);
		prepareRoad(i, presult);
	}

	for (i=0; i < roadnum; i++){

		rl[i].start.x = (int)(((rl[i].start.x - recttx) /rx) + START_X);

		rl[i].start.y = (int)(((rl[i].start.y - rectty) /ry) + START_Y);
		rl[i].finish.x = (int)(((rl[i].finish.x - recttx) /rx) + START_X);
		rl[i].finish.y = (int)(((rl[i].finish.y - rectty) /ry) + START_Y);

	}

}

void CAnkaGISView::prepareRoad(int pointID, int type)
{
	POINT p1, p2;
	p1.x = pts[pointID].x;
	p1.y = pts[pointID].y;
	p2.x = pts[pointID+1].x;
	p2.y = pts[pointID+1].y;

	if (type == 1) {

		rl[roadnum].start = p1;
		rl[roadnum].finish = p2;

		roadnum++;
	}
	else if (type == 2) {
```

```
                FindExternalPoint(p1,p2);

                rl[roadnum].start = donusPoints[0];
                rl[roadnum].finish = p2;

                roadnum++;
        }
        else if (type == 3) {

                FindExternalPoint(p2,p1);

                rl[roadnum].start = p1;
                rl[roadnum].finish = donusPoints[0];

                roadnum++;
        }
        else {

                FindExternalPoint(p1,p2);

                if (donusPoints[0].x > 0 && donusPoints[1].x > 0) {

                        rl[roadnum].start = donusPoints[0];
                        rl[roadnum].finish = donusPoints[1];
                        roadnum++;
                }
        }
}
```