UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

A web-based graphical user interface to display spatial data

by
Claudia Alexandra Fiedeldey

Submitted in partial fulfilment of the requirements for the degree MSc (Computer Science)
in the Faculty of Science
University of Pretoria
Pretoria
April 2000

Table of Contents

List Of Figures

List Of Tables

Abstract

A web-based graphical user interface to display spatial data

By Claudia Alexandra Fiedeldey
Supervisor: Prof. Judy Bishop
Department Computer Science
MSc (Computer Science)

This dissertation presents the design and implementation of a graphical user interface (GUI) to display spatial data in a web-based environment. The work is a case study for a web-based framework for distributed applications, the *Web Computing Skeleton*, using a distributed open spatial query mechanism to display the geographic data. The design is based on investigation of geographic information systems (GISs), GUI design and properties of spatial query mechanisms. The purpose of the GUI is to integrate information about a geographic area; display, manipulate and query geographic-based spatial data; execute queries about spatial relationships and analyse the attribute data to calculate the shortest routes for emergency response.

The GUI is implemented as a Java applet embedded in a web document that communicates with the application server via generic GIS classes that provide a common interface to various GIS data sources used in the spatial query mechanism to access a geographic database. Features that are supported by the distributed open spatial query mechanism include a basic set of spatial selection criteria, spatial selection based on pointing, specification of a query window, description of a map scale and identification of a map legend. The design is based on a formal design process that includes the selection of a conceptual model, identification of task flow, major windows and dialog flow, the definition of fields and detailed window layout and finally the definition of field constraints and defaults. The conceptual model characterises the application and provides a framework for users to learn the system model. This model is conceptualised as a map that the user manipulates directly. Unlike a typical map, which just shows spatial data such as roads, cities, and country borders, the GIS links attribute data like population statistics to the spatial data. This link between the map data and the attribute data makes the GIS a powerful tool to manipulate and display data.

To measure the performance of displaying spatial data, two main factors are considered, namely processing speed and display quality. Factors that affect the processing speed include the rate of data transfer from the generic GIS classes, the rate data is downloaded over the network and the speed of execution of the drawing. Two factors that influence the spatial data display quality are pixel distance and bitmap quality. The pixel distance set in the geographic database is represented by two pixels on the display screen, which affects the display quality since the pixel distance is the upper limit for display granularity. This means that setting the pixel distance is a trade-off between the processing speed and the display quality. Bitmaps are raster images that are made up of pixels or cells. To improve the raster image quality, the bitmap resolution can be adjusted to display more pixels per centimetre.

Opsomming

'n Web-baseerde grafiese grafiese gebruikerskoppelvlak om ruimtelike data voor te stel

Deur Claudia Alexandra Fiedeldey
Toesighouer: Prof. Judy Bishop
Departement Rekenaarwetenskap
MSc (Rekenaarwetenskap)

Hierdie verhandeling toon die ontwerp en implementasie van 'n grafiese gebruikerskoppelvlak (GGK) om ruimtelike data voor te stel in 'n web-baseerde omgewing. Hierdie is 'n gevallestudie vir 'n web-baseerde raamwerk vir verspreide toepassings, die *Web Computing Skeleton*, wat 'n verspreide oop ruimtelike navraagmeganisme gebruik om grafiese data voor te stel. Die ontwerp word baseer op eienskappe van geografiese inligting stelsels (GIS), GGK-ontwerp en ruimtelike navraagmeganismes. Die doel van die GGK is om inligting van 'n geografiese area te integreer; om geografies baseerde ruimtelike data te vertoon en manipuleer; om navrae oor ruimtelike verwantskappe uit te voer en attribuut data te analiseer om die kortste roete te bereken ter verbetering van noodreaksietye.

Die GGK is implementeer as 'n Java miniprogram wat in 'n web dokument ingebed is en met die toepassingsbediener kommunikeer deur generiese GIS klasse wat gebruik word in die ruimtelike navraagmeganisme om toegang tot 'n geografiese databasis te verkry. Kenmerke wat die verspreide oop ruimtelike navraagmeganisme ondersteun is 'n basiese stel van ruimtelike seleksiekriteria, ruimtelike seleksie gebaseer op verwysing, spesifikasie van 'n navraagvenster, beskrywing van die skaal 'n kaart skaal en identifikasie van 'n sleutel vir die kaart. Die ontwerp is baseer op 'n formele ontwerpsproses wat bestann uit die keuse van 'n konseptuele model, die identifisering van taakvloei, die identifisering van hoofvensters and dialoogvloei, die definisie van velde en die gedetaileerde vensteruitleg en definisie van veldbeperkings and verstekwaardes. Die konseptuele model karakteriseer die toepassing en verskaf 'n raamwerk vir gebruikers om die stelselmodel aan te leer. Hierdie model word gekonseptualiseer as 'n kaart wat die gebruiker direk manipuleer. Anders as 'n tipiese kaart wat net ruimtelike data soos paaie, stede en landgrense vertoon, koppel die GIS attribuutdata soos bevolkingstatistieke aan die ruimtelike data. Hierdie koppeling tussen die kaart data and die attribuutdata maak die GIS 'n kragtige gereedskap om data te manipuleer en vertoon.

Om die vertoon van ruimtelike data se prestasie te meet word twee hooffaktore in ag geneem: naamlik verwerkingsspoed en vertoonkwaliteit. Faktore wat verwerkingsspeod affekteer is die tempo van dataoordrag van die generiese GIS klasse, die tempo wat data afgelaai word oor die netwerk en die spoed van verwerking van die tekening. Twee faktore wat die kwaliteit van vertoon van ruimtelike data beinvoed is pixelafstand en bispatroonkwaliteit. Die pixelafstand soos gestel in die geografiese databasis, word voorgestel deur twee pixels op die vertoonskerm en affekteer die kwaliteit van vertoon, aangesien die pixel afstand die boonste limiet vir vertoongranulariteit is. Dit beteken dat pixelafstand 'n kompromis is tussen die verwerkingsspoed en die vertoonkwaliteit. Om die rasterbeeldkwaliteit te verbeter, moet die bispatroonresolusie gestel word om meer pixels per sentimeter te vertoon.

CHAPTER 1

1. Introduction

1.1 A web-based graphical user interface to display spatial data

Why develop a web-based graphical user interface to display spatial data?
There are three reasons:

- *Web-based*
  The Internet has extended the accessibility of information, allowing any browser-enabled client to access web-based applications across a network. This application demonstrates how a web-based application can be used within a heterogeneous distributed environment to access data worldwide.

- *GUI*
  Graphical user interface (GUI) development is an integral part of developing web-based applications. The various design issues that arise are shown in this application.

- *Spatial*
  To enable GIS applications to become usable to a wider audience, there is a greater need for spatially enabled applications. This application uses a spatial query mechanism within a Web Computing Skeleton [Šerbedzija et. al., 1997] to display spatial data.

This research studies issues concerning the design and implementation of a web-based GUI to display spatial data. The GUI illustrates how a browser-enabled client can access, display, query and manipulate spatial data available across the web.

1.2 Overview of the Dissertation

Chapter 2 provides background information on Geographic Information Systems (GIS) as a whole. Firstly a GIS is defined, a brief overview of how a GIS works is given and the possible GIS data models are described. These concepts are essential in understanding the principles used in designing the graphical user interface.

Chapter 3 introduces the open distributed spatial query mechanism, which is used as a case study for the implementation of the graphical user interface. Details of the design and implementation of the spatial query mechanism are provided. This chapter gives an overview of the Web Computing Skeleton, outlines the architecture and design of the distributed open spatial query mechanism and then explains the role of the GUI as case study for the distributed open spatial query mechanism. This chapter essentially summarises the work of others in this project, which is essential to the understanding of following chapters.

Chapter 4 looks at the actual design of a graphical user interface. The design is divided into the identification of the conceptual model, describes how task flow affects the design, incorporates dialog design as part of the GUI design process and includes various usability issues that arise. The conceptual model looks at the user analysis, task flow identifies the tasks that users require. The dialog design focuses on the GUI controls, defines the window components, explores windows navigation techniques and common window attributes, and introduces a presentation model. Usage and design issues of the menu system are also examined. The GUI design process explores the techniques used to implement effective GUI's, prototyping and usability testing techniques, GUI modelling techniques and principles of a user-centric design.

Chapter 5 applies the design techniques of Chapter 4 to the design of the GUI to display spatial data. Chapter 5 also describes the conceptual model that is used, identifies the required tasks within the task flow, presents the dialogs within the dialog flow and describes the details of the design procedure.

Chapter 6 goes into further detail of the implementation of the GUI. This includes the advantages and disadvantages, as well as the reasoning behind using Java as the language for the implementation. Further, this section shows how the GUI is implemented using the generic GIS classes of the distributed open spatial query mechanism presented in Chapter 3.

Chapter 7 provides performance criteria, a comparison of the available algorithms and optimisations that are used to display spatial data and calculate the shortest route.

Finally, the conclusion summarises the work accomplished and presents further research options.

1.3 Contribution of this Research

The research on GISs and GUI design conducted for this dissertation contains the following contributions:

- The GUI as a case study in the Web Computing Skeleton [Šerbedzija et. al., 1997] using the distributed open spatial query mechanism [Coetzee and Bishop ^{A}, 1998].
- The design of a GUI to display spatial data in a web-based environment.
- The evaluation of this design against properties and requirements of graphical user interfaces gathered during research done for this dissertation.
- The implementation of the GUI using Java.
- Optimisations of the performance of the GUI.

CHAPTER 2

2. Geographic Information Systems

2.1 Definition

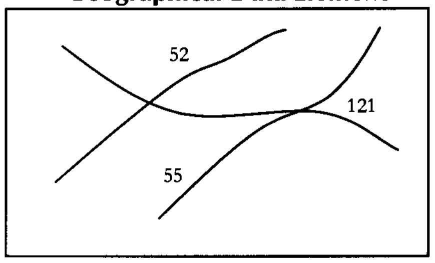
A geographic information system (GIS) is an information system that is designed to work with data referenced by spatial or geographic co-ordinates. In other words, a GIS is both a database system with specific capabilities for spatially referenced data, as well as a set of operations for working with the data [Deok-Yoon, 1995].

According to [Maguire, 1991] there is one single common feature in all definitions of a GIS, namely that GIS are systems which deal with geographical information as follows:

> "Reality is represented as a series of geographical features defined according to two elements. The *geographical data element* is used to provide a reference for the *attribute data element.* For example, administrative boundaries, river networks and point locations of hilltops are all geographical features that provide reference for, respectively, census counts, river water flows or site elevations."

Figure 2.1 illustrates this reference between the geographical and attribute data elements.

Geographical Data Element



Attribute Data Element

ROAD ID	ROAD NAME	DISTANCE (KM)
52	Hendrick Verwoerd Drive	5
55	Jean Avenue	7
121	Lenchen Avenue	3

Figure 2.1 The reference between geographical and attribute data elements

In the strictest sense, a GIS is a computer system capable of assembling, storing, manipulating, and displaying geographically referenced information, i.e. data identified according to their locations. Practitioners also regard the total GIS as including operating personnel and the data that go into the system [USGS Web Team, 1997].

Basic components of a GIS are:

- Cartographic data (images or vector-line data)
- Attribute data
- Hardware
- Software
- People
- Applications

2.2 How does a GIS work?

A GIS integrates maps and data by using links between map features and database records.

The following are functional elements of a GIS [Deok-Yoon K, 1995]:

- *Data acquisition* is the process of identifying and gathering of data required for the specific application.
- *Pre-processing* is the process of manipulating the data in several ways so that it can be entered into the GIS. This process can involve format conversion, data reduction and generalisation, error detection and editing, merging of points into lines and of points and lines into polygons, edge matching, rectification/registration, interpolation and photo interpretation.
- *Data Management* functions handle the creation of, and access to, the database itself. Issues of concern are efficiency, handling of multiple users, handling multiple data sets, reducing data redundancy, tiling and spatial indexing.
- *Manipulation and Analysis* applies analytic operators that work with the database contents to derive some new information that is of use to the user. Operations include reclassification and aggregation, rotation, translation and scaling, regression, correlation, and cross-tabulation.
- *Product Generation* is the phase in which the final outputs from the GIS are created. These can include thematic maps, dot line maps, charts and histograms.

The following GIS functions form the basis of the functional elements of a GIS [USGS Web Team, 1997]:

- Data capture and digitising
- Data integration and exchange
- Projection and registration
- Data structures for spatial, attribute and relationship analysis
- Data modelling
- Data access and query
- Topological modelling and thematic displays
- Overlay analysis
- Data presentation and display
- Map and map-book production
- Relating information from different sources

2.2.1 Data Capture

If the data to be used is not already in digital form, that is, in a form the computer can recognise, various techniques can capture the information. Maps can be digitised, or hand-traced with a computer mouse, to collect the co-ordinates of features. Electronic scanning devices will also convert map lines and points to digits. A GIS can be used to emphasise the spatial relationships among the objects being mapped. While a computer-aided mapping system may represent a road simply as a line, a GIS may also recognise that road as the border between wetland and urban development, or as the link between Main Street and Blueberry Lane. Data capture, or putting the information into the system, is the time-consuming component of GIS work. Identities must be specified for the map objects and spatial relationships must be defined between them. Data that has been captured automatically, must also be edited to remove blemishes. For example, a blemish might connect two lines that should not be connected. Extraneous data must be edited or removed from the digital data file.

2.2.2 Data integration

A GIS makes it possible to link, or integrate, information that is difficult to associate through any other means. Thus, a GIS can use combinations of mapped variables to build and analyse new variables. By using GIS technology and say Water Company billing information, it is possible to simulate the discharge of materials in the septic systems in a neighbourhood upstream from a wetland. The bills show how much water is used at each address. The amount of water a customer uses will roughly predict the amount of material that will be discharged into the septic systems, so that areas of heavy septic discharge can be located using a GIS.

2.2.3 Projection and registration

Map information in a GIS must be manipulated so that it registers, or fits, with information gathered from other maps. Before the digital data can be analysed, they may have to undergo other manipulations - projection conversions, for example - that integrate them into a GIS. Projection is a fundamental component of map making. A projection is a mathematical means of transferring information from the Earth's three-dimensional curved surface to a two-dimensional medium - paper or a computer screen. Different projections are used for different types of maps because each projection is particularly appropriate to certain uses. For example, a projection that accurately represents the shapes of the continents will distort their relative sizes. Since much of the information in a GIS comes from existing maps, a GIS uses the processing power of the computer to transform digital information, gathered from sources with different projections to a common projection.

2.2.4 Data structures

A GIS must be able to convert data from one structure to another because data sources may not be compatible. Digital data is collected and stored in various ways and may not be interpretable to each other. Image data from a satellite that has been interpreted by a computer to produce a land use map can be read into the GIS in raster format. Raster data files consist of rows of uniform cells coded according to data values. An example would be land cover classification. Raster data files can be manipulated quickly by the computer, but they are often less detailed and may be less visually appealing than vector data files, which can approximate the appearance of more traditional hand-drafted maps. Vector digital data have been captured as points, lines (a series of point co-ordinates), or areas (shapes bounded by lines). An example of data that is typically held in a vector file would be the property boundaries for a housing subdivision. Data restructuring can be performed by a GIS to convert data into different formats. For example, a GIS may be used to convert a satellite image map to a vector structure by generating lines around all cells with the same classification, while determining the cell spatial relationships, such as adjacency or inclusion. Thus, a GIS can be used to analyse land use information in conjunction with property ownership information.

2.2.5 Data modelling

A GIS can be used to depict two- and three-dimensional characteristics of the Earth's surface, subsurface, and atmosphere from information points. For example, a GIS can quickly generate a map with lines that indicate rainfall amounts. Such a map can be thought of as a rainfall contour map. Many sophisticated methods can estimate the characteristics of surfaces from a limited number of point measurements. A two-dimensional contour map created from the surface modelling of rainfall point measurements may be overlain and analysed with any other map in a GIS covering the same area.

2.2.6 Information retrieval

A GIS retrieves recorded information at a location, object, or area on the screen from data files. A GIS can use scanned aerial photographs as a visual guide to determine, for example, the geology or hydrology of the area or how close a swamp is to a wetland. This analysis, allows the user to draw conclusions about, for example, the swamp's environmental sensitivity.

2.2.7 Topological modelling

A GIS can recognise and analyse the spatial relationships among mapped phenomena. Conditions of adjacency (what is next to what), containment (what is enclosed by what), and proximity (how close something is to something else) can be determined with a GIS. For example, the GIS can determine whether any gas stations operate within two miles and uphill from a wetland.

2.2.8 Overlay analysis

The data in a map is organised into map layers. Layers are like transparencies. Each layer contains different drawings, that stacked together can be seen as a single map. For example, there might be one layer that contains roads, another that contains cities, and another that contains state boundaries. When all of the layers are displayed, a single map with all of the data: roads, cities, and states is seen. To see the states and cities only, the roads layer can be turned off to hide the roads. Different types of data can be separated into different layers, with separate different levels of detail in separate layers. Additionally, because the map data is on separate layers, objects can be selected from a specific layer without selecting overlapping objects from other layers. Thus, the organisation of map data into layers improves the ability to work with the data while still displaying the data as a single map. For example, using maps of wetlands, slopes, streams, land use and soils, the GIS might produce a new map layer or overlay that ranks the wetlands according to their relative sensitivity to damage from nearby factories or homes.

2.2.9 Data output

A critical component of a GIS is its ability to produce graphics on the screen or on paper that convey the results of analysis. Wall maps and other graphics can be generated, allowing the viewer to visualise and thereby understand the results of analyses or simulations of potential events.

2.2.10 Map production

Researchers are working to incorporate the map making experience of traditional cartographers into GIS technology for the automated production of maps. Using a GIS and digital versions of the 1:100 000 - scale transportation network, political boundaries, and hydrographic features, cartographers produced a 1:500 000 - scale standard base map of New Jersey. This digital revision was done in three steps of map scale reduction: 1:100 000, 1:250 000, and 1:500 000. Each scale reduction required edge matching, or panelling, of the larger scale maps to produce the next small-scale map. In addition, through the process known as generalisation, the amount of information was reduced to make the smaller scale map readable.

2.2.11 Relating information from different sources

A GIS can use information from many different sources in many different forms to analyse data. The primary requirement for the source data is that the locations for the variables are known. Location may be annotated by x, y and z co-ordinates of longitude, latitude, and elevation, or by such systems as postal codes or distance markers. Any variable that can be located spatially can be read into a GIS. A GIS can convert existing digital information, which may not yet be in map form into forms it can recognise and use. For example, digital satellite images can be analysed to produce a map-like layer of digital information about vegetative covers.

2.3 GIS Data Models

In essence a GIS contains two different types of data, geographic data and attribute data. The physical storage of both types of data can similarly be separated, although the relationship has to remain intact. There are three approaches to database storage in GIS [Bernhardsen, 1999]:

- The integrated data model approach stores both geographic data and attribute data in a single DBMS.
- The hybrid data model approach comprises two separate DBMS's, one for geographic data and one for attribute data.
- The third approach is essentially a special case of the hybrid model, where one DBMS for geographic data is connected to several different DBMS's for attribute data.

2.3.1 The Integrated Data Model

In the Integrated Data Model, the GIS serves as the query processor on top of the DBMS. Most implementations are vector topological. This means that relational tables store map co-ordinate data for the points and line segments. Attributes may be stored in the same tables as the geographic feature database or in separate tables accessible via relational joins. This results in co-ordinate pairs for individual vertices along a line to be stored in different rows of a table.

The problem with this design is that if large amounts of data have to be retrieved, there is a substantial performance overhead. To achieve satisfactory retrieval performance, co-ordinate strings can be stored in long or bulk data columns in tables, but this means that the database is no longer in first Normal Form, because each column value is not atomic.

2.3.2 The Hybrid Data Model

To overcome the performance overhead of the integrated data model, the hybrid data model was developed. Geographic data is stored in a set of direct access operating system files for speed of input/output. The attribute data is stored in a DBMS. Unique identifiers link the data stored in the relational database table of attributes that allow them to be tied to individual geographic elements.

2.3.3 The Specialised Hybrid Data Model

To cater for multiple DBMS's with attribute data, the specialised hybrid data model was developed. Vendors use generic relational DBMS interfaces to access the data, and the widespread use of the Open Database Connectivity (ODBC) standard from Microsoft makes this possible, and now Java Database Connectivity (JDBC) from Sun Microsystems Inc.

CHAPTER 3

3. The Distributed Open Spatial Query Mechanism

The Emergency Response System serves as a case study for the implementation of the distributed open spatial query mechanism as described by [Coetzee and Bishop ^{A}, 1998]. The distributed open spatial query mechanism, in turn, serves as a case study for the Web Computing Skeleton described by [Šerbedzija et al, 1997]. This chapter gives an overview of the Web Computing Skeleton, outlines the architecture and design of the distributed open spatial query mechanism and then chapter 4 shows how the distributed open spatial query mechanism is used as a case study.

3.1 The Web Computing Skeleton

The Web Computing Skeleton is a web-based system whose function is to provide the basic software structures for web computing. The Web Computing Skeleton is constructed from prefabricated web-enabled components with the ability to open and maintain web connections and provide collaboration over the Internet. The Web Computing System uses distributed object-oriented techniques to accomplish heterogeneous and dynamically configured software.

The main components of the Web Computing Skeleton include an applet running in a remote user's web browser, the directory server and the application server. The applet serves as a user interface to the application server and the directory server maintains a database of available applications and their locations on the application servers.

Distributed GIS queries are conducted within the remote execution framework provided by the Web Computing Skeleton. Figure 2 illustrates how the distributed GIS queries fit into the Web computing skeleton. The purpose of the Java applet on the Web Browser is to serve as a user interface from where distributed GIS queries are launched and the results of these GIS queries are displayed. The Java applet on the Web Browser communicates with a Java application on the Application Server by means of the generic GIS classes in Java. The generic GIS classes provide a common interface to GIS's from different vendors. These classes are implemented by making calls to a driver DLL. There will be a driver for each vendor-specific GIS. At a low-level access to the data in a specific GIS is provided by a proprietary API in C or C++. The actual GIS data can either be on the application server itself or on a different machine.

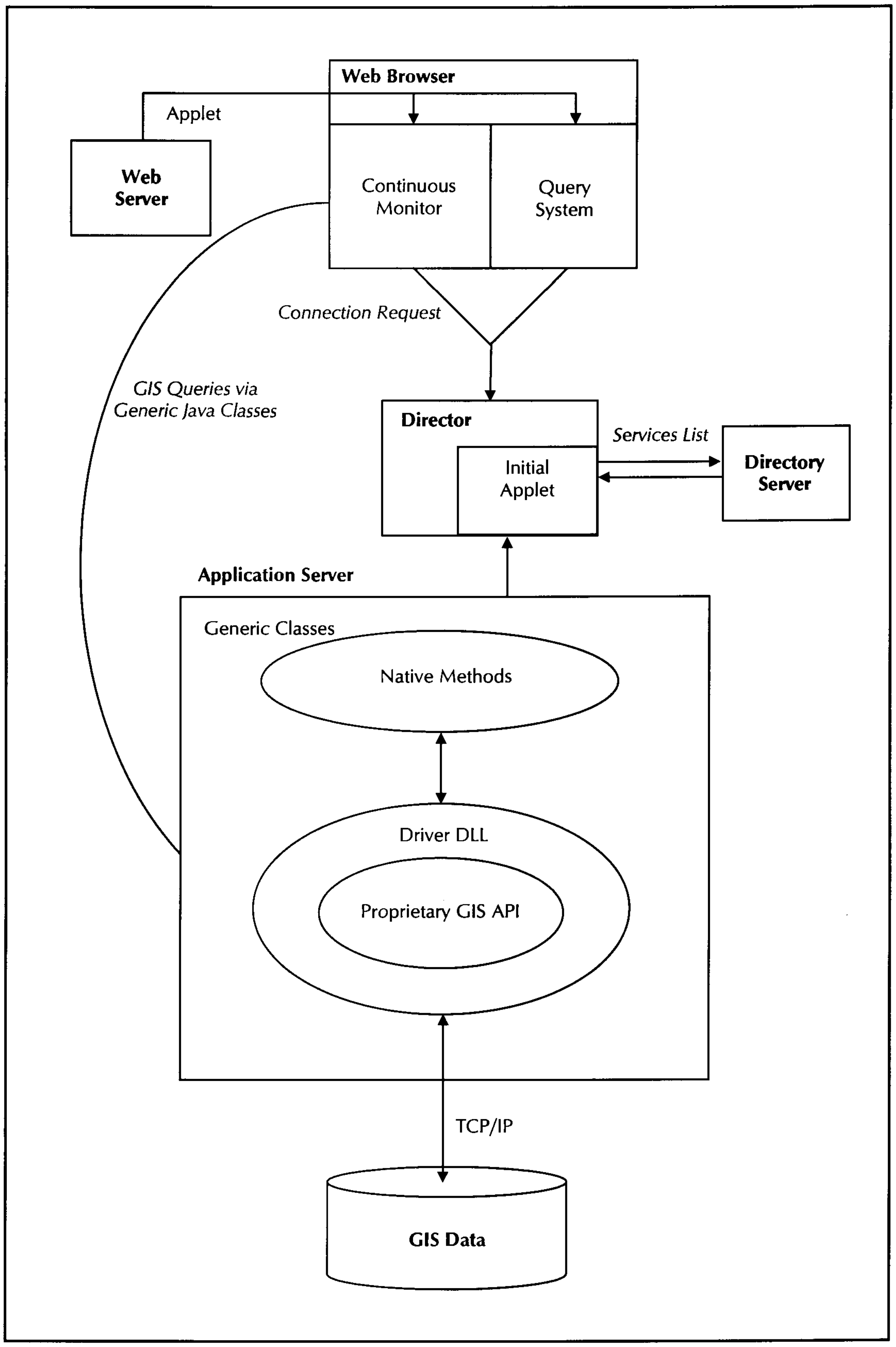


Figure 3.1 The Web Computing Skeleton [Šerbedzija et al, 1997]

3.2 Architecture of a Distributed Open Spatial Query Mechanism

As described by [Coetzee and Bishop ^{A}, 1998], the JDBC API is used as a model for the design of the distributed open spatial query mechanism that was used to implement the generic GIS classes. Section 3.3.1 gives an overview of JDBC, as introduction to the distributed open spatial query mechanism, discussed in 3.3.2.

3.2.1 Java Database Connectivity (JDBC)

The JDBC Database Access API was developed by JavaSoft, a division of Sun Microsystems Inc. According to [Sun Microsystems ^{B}, 2000] JDBC developed as a result of Java's lack of a native ability to link to local or remote database servers directly from Java applets. JDBC creates a programming level interface for connecting with heterogeneous SQL databases in a uniform manner. Consequently, all ODBC enabled databases should work with JDBC with no changes necessary.

3.2.1.1 JDBC API

The JDBC API allows developers to take advantage of the Java platform's cross-platform applications. The JDBC API, send SQL statements and process the results. The Data Access API is expressed as a series of abstract Java interfaces that allow application developers to open connections to a database or access any tabular data, execute SQL statements and process the results.

The JDBC API [Sun Microsystems ^{B}, 2000]:

- java.sql.DriverManager handles loading and unloading of appropriate database drivers to create new database connections.
- java.sql.Connection represents a database connection, exposing the database drivers.
- java.sql.Statement provides a container for executing SQL statements using a connection.
- java.sql.ResultSet controls access to data returned from the database server to the Java applet.

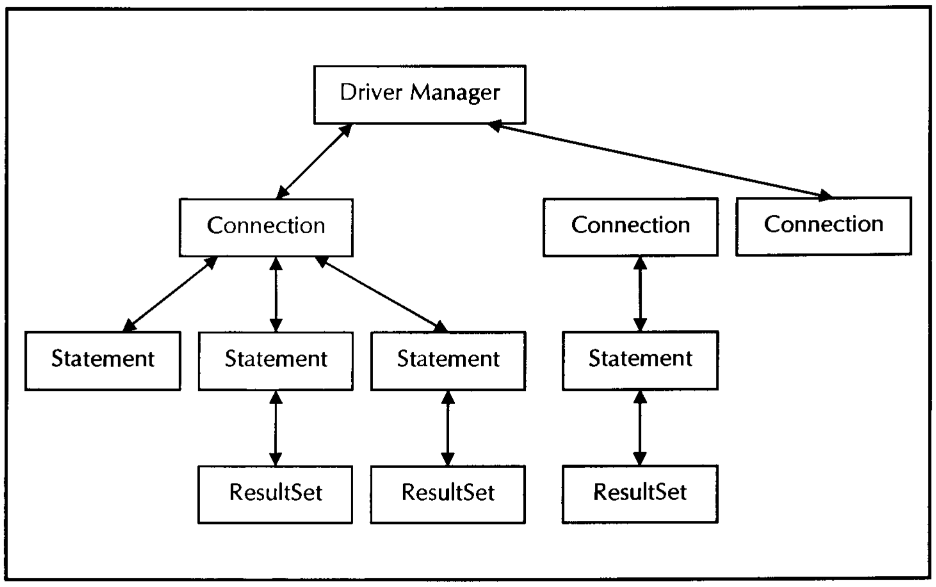


Figure 3.2 The JDBC API [Sun Microsystems ^{B}, 2000]

3.2.1.2 JDBC Architecture

Applets

JDBC uses the two-tiered client/server architecture for applet-to-database connection. Internet or Intranet users running browsers connect to a local or remote Web server and download an HTML document with the embedded applet. The applet executes within the Java-enabled browser environment by using a JDBC driver to a database server.

Currently, the most publicised use of Java is for implementing applets that are downloaded over the net as part of web documents. Among these will be database access applets, which could use JDBC to access the databases. The most common use of applets may be across untrusted boundaries, e.g. fetching applets from other sources on the Internet. However, applets might also be downloaded on to a local network where client machine security is still an issue. Figure 4 illustrates the Internet scenario.

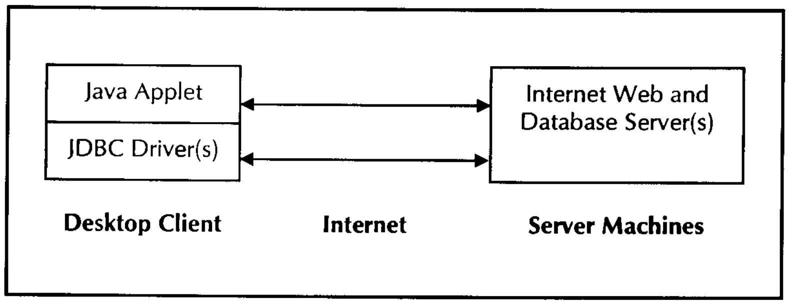


Figure 3.3 The JDBC Internet scenario [Sun Microsystems ^A^, 2000]

Database applets typically differ from traditional database applications in a number of ways:

- Untrusted applets (from an untrusted network source) are constrained in the operations they are allowed to perform. They are restricted from accessing local files and cannot open a connection to a host other than the one from which they came
- Applets on the Internet present new problems with respect to identifying and connecting to databases.
- Performance considerations for a JDBC implementation differ when the database can be located anywhere.

Applications

Java can also be used to build normal applications that run like any custom application on a client machine. In this case the Java code is trusted and can read and write files, open network connections, etc., just like any other application code. Trusted applets are applets that have convinced the Java virtual machine that they can be trusted. From the JDBC point of view, these applets can be treated the same as Java applications for security purposes, but they may behave more like applets for other purposes, e.g. locating a database on the Internet.

Server

The database server can exist anywhere on the Internet or Intranet. Database vendors support JDBC through the JDBC driver interface or through the ODBC connection. Developers can also interface ODBC through a JDBC-ODBC bridge.

Three-tiered client/server model

JDBC also offers a three-tiered development model for access to database servers. In the model the Java applet or application makes a call to a middle layer service, in contrast to direct client/server access from Java GUI's to DBMS servers. This "middle-tier" of services can be a transaction monitor, object request broker (ORB) or Web server API. The middle layer then makes a call to the database server.

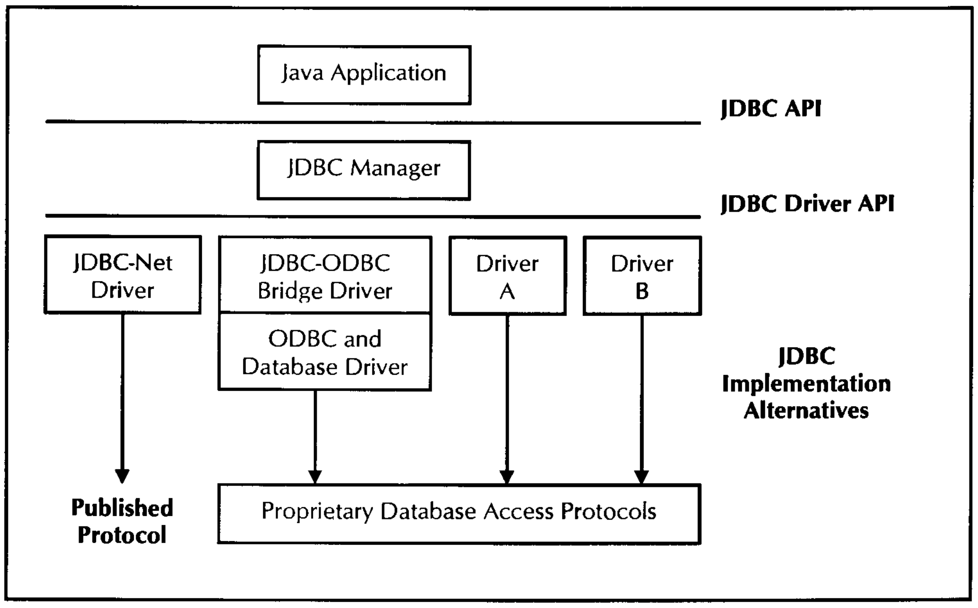


Figure 3.4 The JDBC implementation alternatives [Sun Microsystems ^{A}, 2000]

3.2.1.3 JDBC Database Access

To access a database a java.sql.Connection is obtained from the java.sql.DriverManager. The driver manager uses the Universal Resource Locator (URL) string as an argument to locate and load the appropriate drivers for the target database that the applet is trying to link to. The driver manager queries each driver, locating one that can connect to the URL. The drivers look at the URL to determine if a sub-protocol is needed. The driver then connects to the remote database returning the correct java.sql.Connection object to access services on the database. JDBC returns a result set that is a set of rows from the database. The results can then be accessed using the java.sql.ResultSet object.

3.2.1.4 Evaluation of JDBC

Advantages

The advantages of JDBC technology [Sun Microsystems ^{A}, 2000]:

- *Leverage existing enterprise data*
  There is no proprietary architecture, which means easy access to informations and existing installed databases on different database management systems.

- *Simplified enterprise development*
  The combination of the Java API and the JDBC API makes application development easy. JDBC hides the complexity of many data access tasks, is simple to learn, easy to deploy and maintain.

- *No network configuration*
  With the JDBC API, no configuration is required on the client side. This feature supports the network computing paradigm and centralises software maintainance.

- *Portability*
  Applications can read and write local files and open network connections.

- *Security*
  Java database applications can be created behind the protection of a corporate firewall where trusted applets behind the firewall can be given privileges not given to untrusted applets downloaded from the Web.

A disadvantage of the JDBC technology is that it is still in the developmental stage. JDBC must still handle security on a public network, database recovery from dropped connections and performance through a translation layer.

Some of the alternatives for persistent storage of databases in Java are:

1. *Relational Database*
   JavaSoft defines the JDBC Database Access API.

2. *Object Database*:
   This uses Java's Object Model or transparent persistence for objects in Java.

3. *Serialising Java Objects*
   The simplest method to make Java persistent is provided by JavaSoft as part of Remote Method Invocation (RMI) in which Java objects are converted into a byte-stream that can be stored persistently in a file or sent out over a network.

3.2.2 Java Native Interface

The distributed open spatial query mechanism is implemented as a Java software layer on top of the vendor-specific C/C++ API's. The Java Native Interface (JNI) serves as a link between the Java classes of the spatial query mechanism and the vendor-specific API's. Figure 6 illustrates the role of JNI in the implementation of the generic GIS classes.

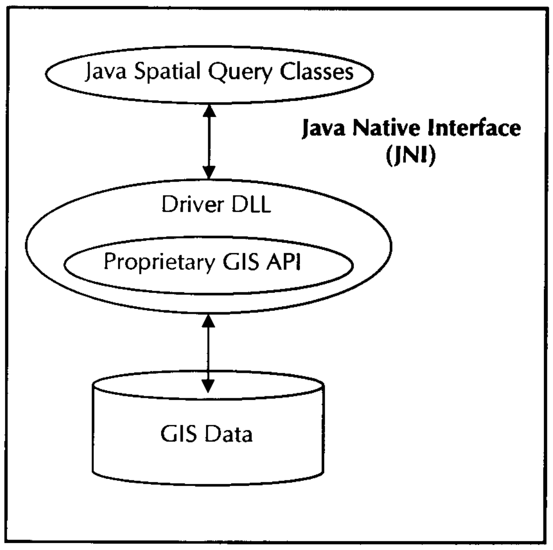


Figure 3.5 Implementation of the distributed open spatial query mechanism [Coetzee and Bishop^A, 1998]

JNI is a native programming interface of Java. JNI allows Java code, running within the Java Virtual Machine (VM), to inter-operate with applications and libraries written in other languages. Most vendors supply a query API in C or C++ with their GIS. There are three reasons why these C/C++ query mechanisms are not used directly in the implementation of the distributed open spatial query mechanism:

- The developer must be able to write one set of code to query GIS's from different vendors, in the same way that JDBC is relates towards relational DBMS's. This means that a layer is needed on top of the vendor-specific API's in C/C++.
- Java provides an advantage for a distributed open spatial query mechanism, to access any GIS through the Internet.
- The spatial query mechanism serves as a case study for the Web Computing Skeleton. The Web Computing Skeleton is implemented in Java, which means that for compatibility the spatial query mechanism is also implemented in Java.

3.2.3 Remote Method Invocation (RMI)

In the implementation of the distributed open spatial query mechanism, a RMI shell was developed around the generic GIS classes that accesses the GIS data. RMI enables developers to create distributed Java-to-Java applications and remote procedure calls (RPC). Methods of remote Java objects can be invoked from other Java virtual machines. A client can call a remote object in a server and that server can also be a client of other remote objects. In essence, RMI allows a developer to transport objects across a network and invoke methods on objects on another computer without having to move those objects to the machine calling the method [McCluskey, 2000]. This means that in the distributed open spatial query mechanism, objects that use native code can then be used as remote objects on a server. Figure 7 illustrates the use of RMI within the implementation of the distributed open spatial query mechanism.

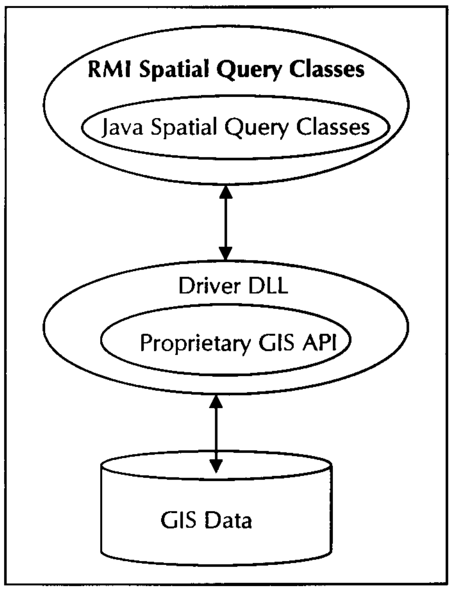


Figure 3.6 RMI in the implementation of the distributed open spatial query mechanism [Coetzee and Bishop^{A}, 1998]

3.2.4 Scenarios using the Web Computing Skeleton

The use of RMI has the advantage of portability and security; the disadvantage is a performance penalty on the server. The two scenarios, without and with RMI, are compared in tables 3.1 and 3.2. The scenario without RMI (table 3.1) is similar to the JDBC-ODBC bridge in the JDBC specification (see section 3.2.1.2), since the code is executed on the client machine.

SERVER MACHINE	CLIENT MACHINE	OTHER MACHINE
Web Server	Web Browser • Java Applet	GIS Database
Directory Server	Application Server • Java Classes (no RMI) • Driver DLL • Proprietary GIS API	

Table 3.1 The Web Computing Skeleton scenarios without RMI

The scenario with RMI (table 3.2) requires only Java code to run on the client machine.

SERVER MACHINE	CLIENT MACHINE	OTHER MACHINE
Web Server	Web Browser	GIS Database
Directory Server	Java Applet	
Application Server • RMI Registry • Driver DLL • Proprietary GIS API		

Table 3.2 The Web Computing Skeleton scenario using RMI

An alternative implementation of the scenario without RMI, would be that the C/C++ code is executed on an application machine, which is neither the client or server machine (table 3.3).

SERVER MACHINE	CLIENT MACHINE	OTHER MACHINE
Web Server	Web Browser • Java Applet	Application Server • Java Application • Driver DLL • Proprietary GIS
Directory Server		GIS Database

Table 3.3 The Web Computing Skeleton scenario without RMI using an application machine

This is the choice used in this work, since this scenario has a performance advantage over the other two scenarios.

3.3 Design of a Distributed Open Spatial Query Mechanism

3.3.1 Overview of the Generic Classes

There are some basic similarities between a relational query mechanism, like the JDBC API, and a spatial query mechanism such as loading a driver and establishing a connection. The difference is in the ways in which criteria for the query are specified, as well as the type and presentation of the resulting data. Instead of using traditional Time, Date, Timestamp and Types classes to represent data specific to a relational DBMS, classes appropriate for the representation of spatial data are included in the spatial query mechanism. Among these are the GeoClip, GeoFeature, GeoShape and GeoStyle classes.

The purpose of the generic GIS classes is to provide a common interface to various GIS data sources. They are designed as a set of abstract classes for which an implementation from different GIS vendors is provided. The following section describes how a specific type of GIS is identified, i.e. how do the generic classes know which implementation to use. A high level specification of the generic GIS classes is then given.

The use of the generic GIS classes in the spatial query mechanism to access a geographic database comprises four steps. Firstly, the appropriate spatial query mechanism driver is loaded. Secondly, a connection to a specific geographic database is established. A connection can be established exclusively, thereby allowing no access to other users while the connection is open. Also, a connection can be established in read-only mode, allowing only queries and not updates to the geographic database on this connection. Thirdly, the spatial query object is created and executed. The GeoQuery class is used instead of the JDBC Statement class to specify spatial criteria, execute a query and create a result set. Lastly, the results are processed. All the features in the result set can be displayed.

3.3.2 High Level Specification of the Generic Classes

Query Classes: These classes are involved in the execution of a spatial query.

- *GeoDriver Class*
  This class loads a vendor-specific version of the generic GIS classes.

- *GeoConnection Class*
  A connection represents a session with a specific geographic database. Within the context of a session queries are submitted to the geographic database and results are returned.

- *GeoDatabaseMetaData Class*
  This class provides information about the geographic database as a whole, including the extents of the geographic database, the feature classes, a list of display styles, a list of shapes and more.

- *GeoQuery Class*
  This class is used to represent queries on the geographic database, providing methods that set various spatial criteria. The query is submitted to the geographic database and the results are returned in a result set. Spatial queries are divided into display and analytical queries.

- *GeoResultSet Class*
  The GeoResultSet represents the results from a query to a geographic database. A result set always comprises a list of features from the geographic database. Features can be traversed one by one, as subsequent linked lists or as subsequent streams of feature data.

- *GeoFeature Class*
  This class represents a feature in a geographic database. The GeoFeature class contains internal references to the underlying proprietary GIS API. A feature is a geographic object, such as a point, line, area or text stored in the geographic database. A feature is defined on a certain feature class, that groups the features by type and display.

Data Classes: These classes are used to represent the spatial data that is returned in the result set after execution of a spatial query.

- *GeoD3Point Class*
  This class represents a three-dimensional floating point geographic co-ordinate.

- *GeoI2Point Class*
  This class represents a two-dimensional integer geographic co-ordinate.

- *GeoVector Class*
  This class represents a three-dimensional geographic vector.

- *GeoClip Class*
  This class represents a rectangular area in geographic co-ordinates.

- *GeoFeatureClass Class*

This class represents the properties of a feature class. Feature classes group features by type and display style.

- *GeoShape Class*
  This class represents a shape as a list of vectors. Shapes are used in display styles to specify how point features are displayed.

- *GeoStyle Class*
  This class represents the display style of a geographic feature. Properties include colour, shape, type, size and scale.

- *GeoItem Class*
  This class represents a geometric item of a feature. Feature data can be presented as a GeoObjectList of GeoItems, which can either contain the geometric items of only one feature, or more than one feature separated by a delimiter.

- *GeoD3Item Class*
  This class is derived from GeoItem, representing a three-dimensional floating point geometric item of a feature.

- *GeoI2Item Class*
  This class is derived from GeoItem, representing a two-dimensional integer geometric item of a feature.

- *GeoListItem Class*
  This class represents an item in a GeoObjectList, a singly linked list. Each GeoListItem refers to an object of any class derived from Object.

- *GeoObjectList Class*
  This class represents a singly linked list of objects. Each item in the linked list is a GeoListItem.

- *GeoInputStream Class*
  This class represents a stream of geographic data. The bytes in a stream are structured as indicators, specifying the number of bytes that follow, with data in between.

3.3.3 Properties of Spatial Query Languages

An important functional element of a GIS is the ability to query and analyse spatial data. A spatial database query is an interactive request for spatial information [Egenhofer and Herring, 1993] and spatial query languages provide the tool needed to analyse spatial data. The focus of this section is on properties and requirements of spatial query languages, especially where they differ from relational query languages. According to [Frank and Mark, 1991] the following spatial selection criteria have been noted.

- *Spatial Selection Criteria*
  The user must be able to select data not only based on predicates over attribute values (e.g. "select all roads with distance > 5 km"), but also based on spatial properties (e.g. "select all roads with distance > 5 km within 1 km of a hospital"). A spatial query language is extended with systematic predicates to select data. These predicates could be one or a combination of the following:
    - ➢ left of
    - ➢ right of
    - ➢ beside (alongside, next to, adjacent to)
    - ➢ above (over, higher than, on to)
    - ➢ below (under, underneath, lower than)
    - ➢ behind (in back of)
    - ➢ in front of
    - ➢ near (close to; next to)
    - ➢ far
    - ➢ touching
    - ➢ between
    - ➢ inside (within)
    - ➢ outside

- *Selection Based on Pointing*
  A spatial query language must be able to accept as value objects visible on the screen to which the user points.

- *The Combination of Query Results*
  The visual integration of the result from more than one query is an important feature of a GIS. It must be possible to specify that the result of a new query is added to (superimposed on) the already displayed map, that it is removed from it, or that the objects selected are highlighted to make them easier to find. The results of a GIS query may contain geographic elements that are displayed on a map, as well as attribute data elements that can be displayed in a table. The coherency between the two types of query results should be clearly visible in the user interface.

- *Spatial Context*
  A spatial query language must either supply a means for the user to specify the necessary context needed to understand the result; or be able to expand the query to include a minimal context if the user did not specify a context.

- *Selection of Query Window*
  Data in a spatial database is usually accessed explicitly or implicitly within a *window of interest*. A spatial query language must supply a means for the user to specify the window or area of interest; or contain rules of how a default is selected if the user did not specify the window or area of interest.

- *Description of Map Scale and Map Legend*
  The user must be able to specify a map scale for the display of query results. Scale influences the material included in the graphically displayed query results (map). The map legend

describes a mapping between the data objects in the database and their graphical rendering. The user must be able to select a map legend for a certain query and change it when necessary.

- *Differentiation of Graphical Representation Based on Attribute Value*
  An attribute value can be spatially distributed; e.g. the average number of police stations per square hectare. To differentiate between value ranges, classes for objects with similar values are usually created, assigning specific graphical values to each of them. This kind of query is called thematic mapping. A spatial query language must include the necessary functionality to perform this kind of thematic mapping.

Features that are supported by the distributed open spatial query mechanism include:

- *Spatial Selection Criteria*
  The GeoQuery class allows spatial selection criteria to be set. A basic set of spatial selection criteria is provided and combinations of these can be used to specify more complex selection criteria.
- *Selection Based on Pointing*
  The GeoQuery class returns the feature that is closest to the specified point.
- *Selection of Query Window*
  The GeoQuery class allows a window of interest to be specified, as three-dimensional co-ordinate sets.
- *Description of Map Scale*
  The distance between two pixels on the display device can be set (in millimetres). Only co-ordinates further apart than the specified pixel distance will be returned in the result set.
- *Description of Legend*
  The map legend lists the feature classes and their associated display styles.

CHAPTER 4

4. Graphical User Interfaces (GUI's)

4.1 Historical Overview

The first graphical user interface was designed by Xerox Corporation's Palo Alto Research Centre (PARC) in the 1970's, but it was not until the 1980's and the emergence of the Apple Macintosh that graphical user interfaces became popular. One reason for their slow acceptance was the fact that they require considerable CPU power and a high quality monitor, which were until recently prohibitively expensive.

However, today GUI's are commonplace. A reason for their success was its use of a tightly restricted vocabulary – a canonical vocabulary based on a limited set of mouse actions – for communicating with users. It also offered a richer visual interaction [Tognazzini, 1995].

4.2 Definition of a GUI

A GUI is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. The intention of a graphical user interfaces is to free the user from learning complex command languages.

Graphical user interfaces generally have the following basic components:

- ***Pointer***
  A symbol that appears on the display screen and that you move to select objects and commands. Usually, the pointer appears as a small angled arrow. Text processing applications, however, use an I-beam pointer.

- ***Pointing Device***
  A device, such as a mouse or trackball, that enables you to select objects on the display screen.

- ***Icons***
  Small pictures that represent commands, files or windows. By moving the pointer to the icon and pressing a mouse button, you can execute a command or convert the icon into a window. You can also move the icons around the display screen as if they were real objects on your desk.

- ***Desktop***
  The area on the display screen where icons are grouped is often referred to as the desktop because the icons are intended to represent real objects on a real desktop.

- ***Windows***
  Divides the screen into different areas. In each window, you can run a different program or display a different file. You can move windows around the display screen, and change their shape and size at will.

- ***Menus***
  Most graphical user interfaces let you execute commands by selecting a choice from a menu.

In addition to their visual components, graphical user interfaces also make it easier to move data from one application to another. A true GUI includes standard formats for representing text and graphics. Because the formats are well defined, different programs that run under a common GUI can share data. This makes it possible, for example, to copy a graph created by a spreadsheet program into a document created by a word processor.

Many DOS programs include some features of GUI's, such as menus, but are not graphics based. Such interfaces are sometimes called graphical character-based user interfaces to distinguish them from true GUI's [Martin and Eastman, 1996].

4.3 Graphical User Interface Design

4.3.1 Elements of Design

There are several elements of design to be considered. Here are some basic principles and their relevance in software systems [Zetie, 1995].

- ***Visibility***
  How easily and directly the user can see *what* can be done and *how* to do it. This applies to all user interface elements and is the key to encouraging users to discover and utilise all the functionality provided.

- ***Feedback***
  How easily and quickly you can determine the results of your actions. Good feedback operates at every level of the user interface and shows not only what you did, it also shows the effect of what you did. The effect is expressed at a level of abstraction that is meaningful e.g. when you click on a button it depresses or when you select text it becomes highlighted. Feedback needs to be presented at the right place, time and level.

- ***Mappings***
  The correspondence between the function desired and the action required achieving it. The user perceives good mappings as those that are concrete, natural and direct. Mappings with fewer controls than there are functions will always be less direct.

- ***Cues and Affordances***
  A clue that utilises the user's existing knowledge to hint at the way a control is used. A control with an obvious usage has strong affordance. A new control will nearly always give a weaker cue than a familiar control, therefore custom controls should only be used if mapping is significantly improved, affordance is adequate and the gain in mapping is worth the loss in affordance. This provides the user with a more consistent use of controls and improves the GUI consistency.

- ***Interpretation***
  There are three levels of information to consider in a system, namely representation, presentation and interpretation. At the lowest level there is a representation of information, which is typically a database of storage facility. At the next level is the presentation of that information by one or more applications, typically through screens and reports. Finally there is the utilisation of information by a user. Between each level there is a requirement for interpretation. The burden of interpretation is a measure of how close the representation and presentation of the data is. The closer the relationship, the greater the burden placed on the user to interpret the data before it can be utilised. Reducing this burden means using the user's vocabulary. In a well-designed system, the functions provided by the system correspond closely to the actions intended by the user, not the internal behaviour or states of the system. Also, the system presents information about its state and the results of actions in terms that match those required by the user, rather than terms of the underlying system.

4.3.2 Stages of Design

The following stages represent a small portion of the total design effort, and to be fully effective need to be incorporated into an encompassing design method that relates the user interface to task analysis, prototyping, user analysis and other design tasks. These stages include selecting a conceptual model and identifying task flow, identifying major windows and dialog flow, defining fields and detailed window layout and finally defining field constraints and defaults [Zetie, 1995].

- ***The conceptual model***
  The conceptual model characterises the application and provides a framework for users to learn the system model.

- ***Workflow and Task flow***
  Workflow describes the sequence of actions performed by a user to complete a business task. Task flow, by analogy, is the sequence of actions supported by the application.

- ***Dialogs and Dialog flow***
  A dialog is a subset of an application's windows consisting of a collection of connected windows that allow a user to carry out an action. Dialog flow describes the modality and possible paths through a dialog.

- ***Detailed design***
  Detailed design includes detailed screen layouts, defining field sizes, positions and prompts and choices of controls.

4.3.3 Conceptual Models

The conceptual model is the unifying vision behind an application and facilitates the illusion that the user is directly manipulating the objects of interest. One of the roles of the conceptual model is to provide a consistent interpretation across an application. Individual actions and objects on the desktop use metaphors within the framework of the conceptual model. A metaphor is an external representation of a component of or an action in the underlying system and can be used to represent both data and actions of the system. The main objective of a metaphor is to relate to a referent that is familiar or readily understood. This may be a physical object, an abstract idea or an established idea in a computer system. A good choice of referent is one that is already well understood by the user and has a broad range of characteristics that correspond in the application.

The user interface should present an abstraction of the user's task and represent the work task, as the user understands it. To achieve a coherent and consistent interface, two key factors have to be considered. Firstly, the choice of centre of the model. There are several options: the model can be data-centred, centred on a task or action or centred on external entities. Second is the choice of a domain of expression. It is important for the user that the domain of expression is able to describe both input and output.

Models can be divided into abstract, symbolic and concrete expressions [Zetie, 1995].

- ***Abstract expressions*** are easy to identify because they have no direct correspondence to the object or action they express. For example, the *Save* command is expressed as *Ctrl+F12.* There is no compelling reason for choosing this key mapping over any other, so the expression is abstract. Abstract expressions are almost always arbitrary mappings and often exhibit poor visibility. However, for experienced users of an application they can provide rapid access to frequently used actions.

- ***Symbolic expressions*** are expressions where a data or action is expressed as being *like* another. Actions within the expression have some, but not all, of the characteristics of the symbolic referent modelled. Examples are Cut, Copy and Paste as symbolic representation of the idea of moving information.

- ***Concrete expressions*** express objects as behaving like the object they represent. An example is the Macintosh desktop. To delete a document, you throw it away, to move a document to another folder you pick it up and move it. Actions in the expression correspond directly to the action they represent in the 'real' world. Concrete expressions of actions are however more programmer-intensive because they rely more heavily on direct manipulation. The ease of use of concrete expressions arises from drawing on knowledge that the user already possesses.

For user interfaces to become more powerful, certain aspects of the interface can be transformed to make the application more understandable. Transformations can be divided into those suitable for output, those suitable for input, and those suitable for data both entered and displayed. Output transformations represent the user presentation of the system information. Since these transformations usually lose information, they are only appropriate when the user does not need to determine the system state. Input transformations, on the other hand, map system values to a subset of possible user interface presentation.

4.3.4 Task Flow

According to [Zetie, 1995] the user's task flow is the way in which the user carries out tasks. This is the interface between real world requirements and application specifications. While still maintaining a user-centred viewpoint and principle-based approach, limitations of information systems and especially its users must be taken into account. Task flow is concerned with the way in which an application or system handles a unit of work. One major component of task flow is the processes identified directly in data flow diagrams. A second major component comes from business rules. Issues include privilege, authority and responsibility for particular tasks. A third major component is the user's working style.

Closure is the point at which a task or subtask has been completed, or when the saves work in progress or leaves a 'risky' area. Closure is so widely exploited in most areas of life that users expect an application to offer points of closure. Therefore the design of a user interface that provides visible, concrete and frequent points of closure will appeal to users. Such a design will emphasise the user's sense of security and satisfaction. To experience the sense of resolution associated with closure, the user also needs to receive affirmative feedback that closure has occurred. Points of closure can either be explicitly controlled by the user or in certain applications will be determined by business rules. Where business rules control points of closure, the user should be made aware of when closure points occur. Ideally these points should correspond to business units of work. When the two cannot coincide, it is the designer's responsibility to make the mismatch explicitly visible to the user.

Users often want to save work in progress even though they have not completed a system transaction. To meet this need, the design should allow the user to divide a long and complex task into smaller components, each of which can be closed individually. There are two major ways to structure support for work breakdown: serial closure and hierarchical closure. Serial closure provides saving points, by which the user can save work up to a given point. Hierarchical closure divides a task into a set of nested component tasks. Once each of these components is committed, the overall task can be closed. In general, serial closure is preferable to hierarchical closure because only the most recent task step has to be closed. It is also possible to combine both nested and serial closure points.

Once tasks have been identified, the designer must proceed to identify the degrees of freedom available. One approach is to examine dependencies between tasks. To identify dependencies, the designer needs to consider the following questions:

- *Does a process contain tasks that must be carried out in a particular order?*
- *Does the user need to suspend a task in progress to carry out a related task?*
- *Does the user need to suspend one task to carry out an independent task?*
- *Do tasks or sequences of tasks need to synchronise at particular points?*
- *Does a task contain subtasks?*

Therefore, to determine the degree of freedom to offer the user, the designer needs to identify technical and technological limits, uncover business and process dependencies and make allowance for user limits and expectations.

Restricting the freedom offered to the user can have the following benefits:

- Eliminates paths of execution that do not lead to closure
- Increases consistency of usage
- Reduces concentration demanded by the user

Also, the user interface must express explicitly every dependency that underlies it. This is also a strong instance of the visibility principle.

Modes are another issue that must be considered when task flow is determined. Modality can be defined in terms of user actions. If, within a given area of application, a user action has the same interpretation whenever and wherever it is executed, the given *space* is considered modeless.

There are some general rules that make modes more useful, regardless of the level of granularity:

- Modes should be used consistently. From the user's point of view, similar *spaces* should behave in a similar way.
- Modes should not be initiated unexpectedly
- It should be clear to the user that a mode has been entered. This can be achieved by providing highly visible feedback that is not obtrusive.
- It should be clear to the user how to escape from a mode.
- It should always be possible for the user to escape from a mode harmlessly e.g. using a Cancel button.

Modes operate at all levels of the user interface and can be a very powerful mechanism for guiding the user to perform correct actions in the correct order, but only if used in a way that respects the fundamental principles of visibility, feedback, mapping and cues.

There are however a number of conflicts that arise in designing the task flow. Two key benefits of GUI's are often in opposition, namely ease of learning and ease of use. Ease of learning suggests immediately intuitive design with plenty of visibility, a lot of guidance and simple, easily understood tasks. Ease of use implies providing the user with efficiency of expression, uninterrupted flow and powerful, complex commands available at the touch of a button. For the experienced user too much guidance becomes obstructive, too much feedback is perceived as instability and too much visibility overloads the user with information.

There are several mechanisms that can be used to achieve the goals identified. Most of these techniques have certain advantages and disadvantages and are appropriate in different circumstances. A typical application uses a selection of these techniques [Zetie, 1995].

- ***Guidance***
  These are opportunities presented to the user to be lead systematically through a complex process. These include the following: w*indows installation programs, wizards* of automated sequences that carry out complex tasks on behalf of the user and c*oaches and tutorials* can provide on-screen instructions to track the user's process and provide information to the user.

- ***Locks***
  Locks ensure that a valid sequence of actions occur in the right order. Strict lockouts prevent an illegal event from occurring; for example disabling a command until all information has been supplied. Interlocks guide the user to perform the right actions at a synchronised point and can be either implicit or explicit.

- ***Forcing techniques***
  These are blocking mechanisms that offer the user a narrow range of choices. Forcing can occur at the level of windows e.g. modal dialogs and alerts. The disadvantage is that they lock the user out of all other parts of the application. Mandatory or required fields prevent the user from moving forward until the information has been supplied. Commands and items can also be dynamically enabled or disabled to prevent the user from choosing inappropriate or unavailable options, to indicate visually that work has been completed by progressively enabling commands as information is supplied and to lead the user through a complex process.

- ***Work queues and task lists***
  One of the least forcing mechanisms is through work queues, where the user is presented with a list of tasks or steps to be carried out. The implementation can be highly generic and data-driven, the flow made explicitly visible to the user and can be made modal or modeless, the interface strongly supports closure and feedback and the model supports partial work by saving information entered so far.

- ***Navigators***
  For less structured, document-driven applications using a tree where branches can be expanded and collapsed under the user's control may be more appropriate.

- ***Customisation and macro languages***
  The user is given the ability to customise the user interface and macro languages provide keystroke recording and playback for frequently used actions or the user can write a program in the macro language. Customisation can be useful when a single application must meet the needs of different classes of users.

4.3.5 Dialog Design

[Zetie, 1995] describes a dialog as a set of related windows used to carry out all or part of a work task. A dialog may contain modal and modeless dialog boxes and other windows. A dialog is resolved when the user completes all actions in the dialog and dismisses the dialog box. Resolution is often a point of closure for the user. The possible paths or sequences of windows that a user passes through before resolving a dialog constitutes the dialog flow. Elements of dialog flow include the number, arrangement and modality of windows. Dialog flow serves two purposes, namely as a mechanism to control or manage task flow and to adapt the task flow to the realties of the interface.

Dialog flow needs to support the following aspects of task flow:

- ***Data Integrity*** ensures that fields satisfy validation rules or value integrity and related fields support referential integrity.
- ***Process flow*** manages the dialog flow to allow all sequences of actions identified in the task analysis, while protecting data integrity.
- ***Business rules*** combine both aspects, so that each action is valid for the current data, the current state of the system and the current user.

The first step is to identify the dialogs themselves. This consists of identifying related information and actions and grouping that information into a single dialog. These dialogs should be closely related to the tasks identified for the application. The point of resolution on a dialog should correspond to a natural point of closure for the user.

Next, the particular functions and information must be allocated to the window. Guidelines include presenting enough information and functions for the user to complete the task successfully, provide visible cues where more detail is available and never present an exceptional option so that the user must learn it.

The majority of windows fall into three categories: documents, modeless dialog boxes and modal dialog boxes:

- ***Documents*** are windows in which the subject matter of the application is presented. Typically, documents are modeless, resizable and minimisable. They may be maximisable if the amount of information presented to the user can be increased and vertically and horizontally scrollable.
- ***Modeless dialog boxes*** are generally used to implement 'tools' that are implemented alongside documents. Tool dialogs are not usually resizable, since components are shown at a fixed size.
- ***Modal dialog boxes*** are normally used for detail windows required to complete a command. They are used to ask users questions, collect details and confirm actions. Modal dialogs are rarely resizable.

In addition, the information presented should be separated, both for input and output, within each dialog into frequently used and rarely used sets. Frequently used information should be allocated to the first window of the dialog, other information to secondary windows. When a window becomes overcomplicated, canvasses can be used to reduce apparent complexity. Related fields should be kept together, but not to the detriment of the other goals.

A message box is a type of modal dialog box that requires special attention because it is the most common form of communication with the user. There are several simple rules that can be followed when it comes to message boxes:

- The message should be simple and make sense.
- Answers should make sense and be easily distinguishable from each other.
- Messages should be complete and contain relevant context.
- The message box must identify which application it came from.

There are four widely used styles of message boxes [Zetie, 1995]:

- ***Information messages*** should only be used when it is essential that the user read and acknowledges the message. Generally, an information message should have only one button, labelled 'OK'.
- ***Question messages*** should make answers distinguishable from another and make sense as a response to the question. Wherever possible, there should be at least one choice that is completely safe for the user.
- ***Warning messages*** can be used when an action cannot complete. Generally, a warning will have only one button acknowledging the message.
- ***Stop messages*** should only be used in the case of the severest errors, explaining the cause of the error, whether any consequential damage has occurred and how the user can continue.

Applications can be divided into two broad classifications: task-oriented and subject-oriented. Within a task-oriented application, the user generally begins with a list of tasks, from which each task is launched. As each task is completed, the user returns to the task list to choose the next. In order for the user to keep track of the steps and position within a process, the task list needs to be available to the user at all times. This can be done in a number of ways:

- The task list window and task windows are modeless
- Each task list window is modal with respect to the task list window
- The current process step and stack is summarised on each window
- The task list and current task are presented in a single window and canvas switching is used to show different tasks as required

Within a subject-oriented dialog, the user selects a subject from a list, works on it and returns to the list of subjects. Generally, the subject-oriented dialog includes a filter that allows the user to limit the documents presented in the list or order them in particular ways. Filters should come below the list in the dialog flow, so that the user can bypass the filtering process. Selection from the list should take place on a single window to allow the user a more efficient task flow. The filter in effect should be visible and frequently used filter options may appear directly on the list window.

4.3.6 Detailed Design

Detailed design should incorporate decisions made about the choice of conceptual model, the task flow and dialog flow. Detailed design incorporates issues of detailed layout, field organisation and graphical effectiveness, which will have significant impact on the ergonomics for the user. The visual design will also influence usage for the medium to long term.

4.3.7 Controls

A control is a component of the display that the user or program can interact with. Most controls are used for both input and output.

The following standard controls are widely used:

- ***Radio button groups***
  Radio buttons map well when a user must choose between a set of mutually exclusive settings. Only one item in the group can be selected at a time and selecting any item automatically deselects all others. Radio buttons are appropriate when the possible values are static and small in number.

- ***Check boxes***
  Check boxes are useful when a field can only have two Boolean values, On or Off. A check box is only appropriate when the value associated with the unchecked setting is obvious.

- ***Combo boxes and pop-down lists***
  Combo boxes are useful for displaying dynamic choices or when there are many choices available to the user. Replacing data entry fields with lists promotes recognition over recall and is useful when the user may have difficulty providing an exact match.

- ***List boxes***
  List boxes allow a user to select one value from many and permit more sophisticated forms of selection than combo boxes. A list box is good for presenting a list of subjects in a transaction processing application, or to represent tasks and work queues.

- ***Pushbuttons***
  Pushbuttons are used for small numbers of commands and actions. Buttons should be used for actions that are specific to a particular dialog or context, whereas menus are preferred for commands that are widely applicable.

- ***Menus***
  Menus are the most common mechanism for organising a large number of commands or actions. Sub-menus can be nested to any depth, although more than three levels should be avoided, since it becomes difficult to explore the menu structure. Because the menu is always available, the commands should be those that are broadly applicable in many windows and modes. Menus can be altered dynamically so that a different set of menu commands can be presented according to context. Menus can also be used to allow users to choose between options, usually large sets, although visibility is poor.

- ***Pop-up menus***
  Popup menus are usually invoked by selecting an object, then clicking the right mouse button. Commonly required actions for the object are then available. There is a significant visibility issue with pop-up menus, namely to know when a menu is available and how to invoke it.

- ***Icons***
  Icons are used in numerous ways in GUI applications: an icon represents an application on the desktop, a minimised window and most frequently pictorial buttons. An icon should instantly communicate its purpose to the user, be easily interpreted and icons must be easily distinguished from each other. Images chosen for icons should be clear in themselves and support the overall metaphor of the application. A group of icons, that represent related actions, should use a consistent visual language derived from a common pictorial vocabulary.

- ***Cursors***
  Cursors are used to provide feedback to the user, for example an 'I Beam' indicates a text field and an hour glass indicates that the user must wait for an action to complete. Cursor shapes are a visual indication, therefore different should be easily distinguished. Cursors can be used to indicate modes, invalid fields or a cut buffer containing material to be pasted.

- ***Tool bars and palettes***
  Toolbars provide one-click access to a range of commonly used actions, which are usually iconic buttons. A tool bar may also feature a form of check box, which when clicked remains down, in the selected state. There may also be a form of radio buttons, where clicking one in the group automatically deselects the previous selection. Tool bars serve three major purposes: they make global, frequently used functions quickly available, they make important functions easy to find and they can be visually attractive.

Custom controls should only be used when the new control significantly improves mapping, compared to a standard control; if the new control gives an adequate cue to the user and the gain offered by improving the mapping compensates for the unfamiliar cue.

4.3.8 Window layout and graphic design

[Siebert and Ballard, 1992] suggest that a good layout serves three basic purposes: it works, it organises and it attracts. A good layout communicates a particular message to a particular audience. The layout of the window maps out a visual path for the user to follow which affects task flow, productivity and accuracy. An attractive layout has a balance and flow that supports the other objectives.

Wherever possible the windows should be designed to show all controls without scrolling and controls that resolve a window should always be visible. Consistency should be a significant factor in window size and shape. The choice of position of a window should be close to the point of activation and not obscure fields that the user needs to see. Any window or dialog that has been moved by the user should be reopened in that position [Cooper, 1995].

In positioning controls, the designer needs to consider issues of ergonomics such as spacing and task flow, issues relating to fields such as field order, alignment and size and issues of consistency. Controls should be organised into groups that are related to the user's perception of the task. When choosing fonts the most important considerations are readability, clarity and then visual appeal. When choosing colours, at least four related aspects should be considered: colour is meaningful, ergonomic, operational and cosmetic. A single colour can have many meanings depending on the person and the context in which it is encountered. Since colour is rarely presented independently of some supporting context, the conceptual model should help reinforce the particular meaning.

4.3.9 Application design

Detailed issues that are applicable across the entire application include the use of language, the importance of consistency and broad-ranging issues in the use of the keyboard and mouse [Zetie, 1995]:

- ***Language***
  Language should be direct and clear, using the user's vocabulary

- ***Consistency***
  Consistency should not become an obstacle to usability goals. Most important is consistency with the user's expectations.

Mouse use can be effective is supporting the illusion of direct manipulation e.g. judicious use of drag and drop. Strong visual cues should be given to the user when mouse actions are available.
Keyboard equivalents for menu items should be provided for the most common commands, although it is more important that menu items be clear and easily distinguished.

4.4 Summary

The priciples and guidelines presented in this chapter are now used a basis for the GUI design of the system, as presented in Chapter 5. The GUI design takes into account all the elements of design and presents the design stages, from the conceptual model to the task flow, dialog design and detailed design.

CHAPTER 5

5. Design of the Emergency Response System

5.1 Overview

The goals and objectives of the *Emergency Response System* are to:

- Provide a graphical user interface within a web-based framework, the Web Computing Skeleton
- Integrate multiple types of information about a geographic area
- Display, manipulate, integrate and query geographic-based spatial data stored in a geographic database
- Read a variety of popular external formats and access attribute data stored in a number of commercial database management systems
- Execute queries about spatial relationships between drawing, geographic, and attribute data
- Analyse the attribute data to determine the shortest route between the nearest ambulance and the accident site, and the shortest route to the nearest hospital

The purpose of the GUI is to query geographic spatial data through the Internet to any number of databases on different servers. The user interface then launches GIS requests and displays the results as visual data. The visual interface works with both spatial and textual data.

The Java applet is embedded in a Web document with code downloaded on demand. Its' purpose is to provide the user with a user interface and display the results of distributed GIS queries. The Application Server which establishes the link to Java via native methods and provides the communication over the network handles communication with the Java applet. The Server processes the GIS database queries, while the client displays these results.

Initially the user interface is downloaded from a Web server and run on a browser. Requests are handled via Remote Method Invocation (RMI) within a remote framework provided by the Web computing skeleton. The Java applet communicates with the Application Server via generic GIS classes that provide a common interface to various GIS data sources.

A screen dump of the GUI is given in Figure 5.1.

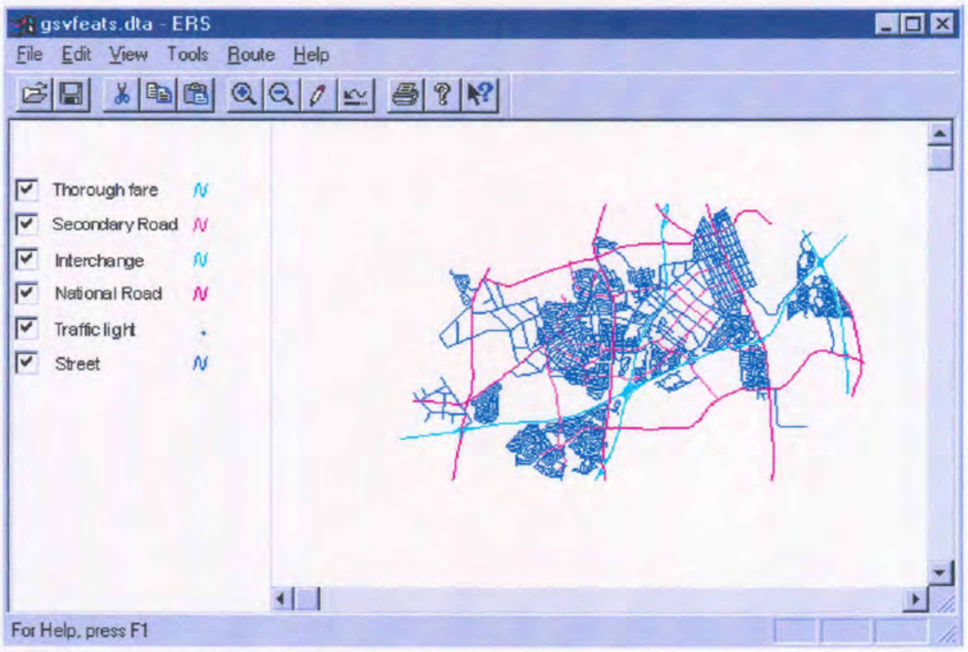


Figure 5.1 The GUI of the Emergency Response System

5.2 Conceptual Model

The unifying vision behind the *Emergency Response System* is a map that the user manipulates directly. All the actions are grouped around the map, to consistently enforce the idea that the capabilities of the system are carried to the map. Explained differently, the map can be seen as the painted area or image and the actions e.g. zoom, pan and annotate as tools that can be applied to the image. The individual actions all use the *toolbox* metaphor, to hide the representation of the data in the geographic database from the user. This abstraction allows the user to manipulate the spatial data without considering the implementation details.

This is an example of a data-centred model, where the model focuses on manipulating the spatial data. The reason for choosing a data-centred model is that it is the most logical expression of the purpose of the user interface within the Web Computing Skeleton. Within this framework, the data source is the most significant factor, where all business tasks relate to the data source.

This model for the *Emergency Response System* describes a symbolic domain of expression, although other forms of expressions are also used to cater for the differing needs of the users. A symbolic domain of expression means that actions have characteristics of the symbolic referent. For example, the toolbox acts as a symbolic referent for the map manipulation actions. Abstract expressions are used to provide short-cut keys for commonly used functions, for example keypress R rescales the map back to the default scale and keypress R refreshes the screen. These short-cut keys provide experienced users with rapid access to frequently used actions. The domain of expression caters for both input and output of data. All input from the user is handled by the symbolic referent, the toolbox, facilitating spatial display, querying and manipulation. All output, that is display and printing of maps and route analyses is handled similarly.

Transformations have to be made to the data in order to present the system information to the user. These transformations are divided into output, input, and those for data both entered and displayed. Output transformations used in the *Emergency Response System* include co-ordinate transformations, world to screen conversions and projections, which simplify the user presentation of the system information, without losing significant detail. Also, map features are divided into points, lines, areas and text. These features represent all geographic information. Input transformations include scaling and display factors, and routing weights, which map system values to a subset of possible user interface presentation.

5.3 Task flow

5.3.1 Overview

Task flow describes how the user carries out the business tasks, as units of work. To identify the business tasks, processes are isolated and displayed in terms of data flow diagrams. Section 4.3.2 illustrates the main processes identified. Section 4.3.3 describes how the business rules affect the software design, discussing issues such as privilege, authority and responsibility for particular tasks. Section 4.3.4 clarifies the effect of the user's working style on the GUI design.

5.3.2 Processes

Opening a map

To open a map in the Web Computing Skeleton, the Java application launches the Emergency Response System applet to display the map. The user can then change the view of the map by zooming or panning to see the exact portion of the map that is required. The map appears in the scroll pane window of the applet and the legend appears in the panel to the left of the map window.

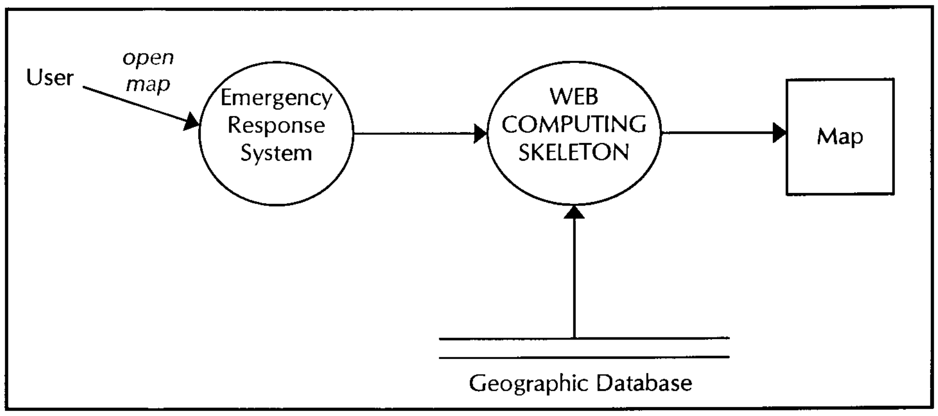


Figure 5.2 Opening a map

Altering the map view

To change the view of the current map, the user can zoom in and out on the map and move the map around (pan). This allows the user to see specified portions of the map as required.

- *Zooming in*

  The user can zoom in on the centre point of the map window. The map view will zoom in by a magnification factor of two on that point.

- *Zooming to a rectangular area*

  The user can use the mouse pointer to highlight a rectangular area on the map, then zoom into that area. The user right-clicks on the map and drags the mouse to draw a rectangle over the area to display. When the mouse button is released, the rectangle disappears, and the map zooms in to display just the area within the rectangle.

- *Zooming out*

  The user can zoom out from the centre point of the map window. The map view will zoom out by a magnification factor of two on that point.

- *Zooming out to the entire map*

  To display the entire map, the user can zoom out to the entire map. This will display the entire map in the original scale factor the map was captured in.

- *Zooming to a map scale*

  The user can specify the map scale factor to zoom to. The larger the scale factor in the scale ratio is, the more the map window will zoom out, which will display a larger area with less detail. The smaller the scale factor is, the more the map window will zoom in, which will display a smaller area with more detail.. A custom scale factor can also be used. The map will zoom in or out to the scale the user specified.

- *Moving up, down, left, or right (pan)*

  The user can slide the map around to display areas that were outside of the previous view, using the horizontal and vertical scroll bars of the map view.

- *Viewing annotations*

  The user can view the map object names or annotations for a selected map feature, for example if the map feature is a regional roads, the regional route name will be displayed next to each regional road. The legend indicates the feature type of the object selected. For example, blue lines could indicate regional roads, yellow lines could indicate national roads and black text could indicate route names.

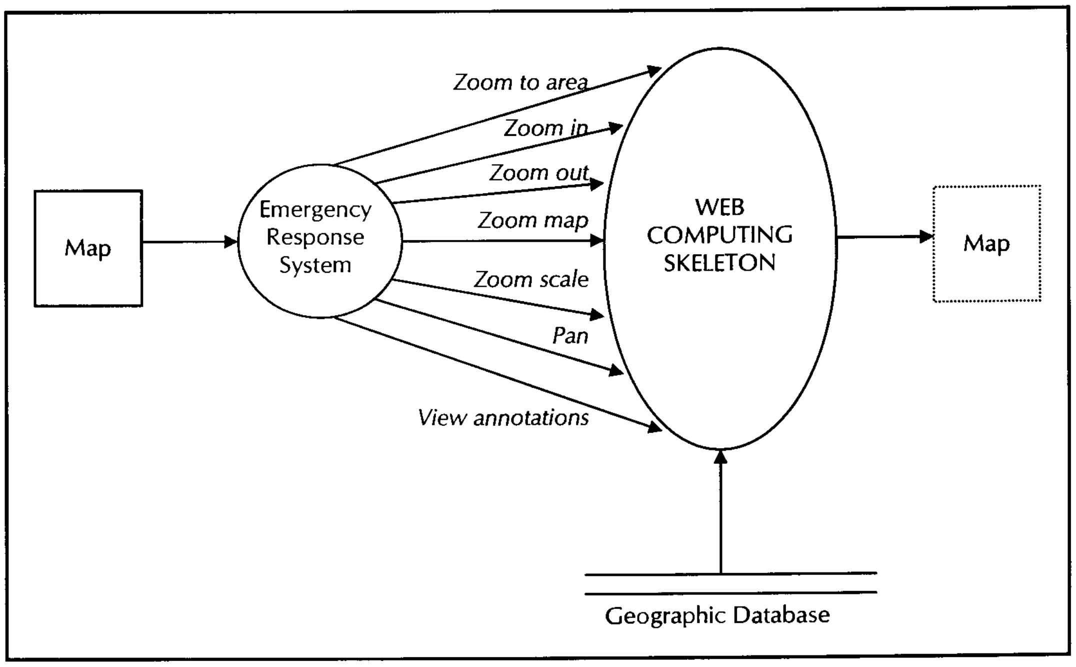


Figure 5.3 Altering the map view

Manipulating map objects

After the user has opened the selected map and moved it around to display the objects required, the user can select locations and weight factors for the route and calculate the shortest route.

To select locations for the route calculation, the user has to load the intersections and weights ASCII comma delimited text files. The intersection file format is specified as the number of intersections followed by each intersection's ID and (x, y) point co-ordinates, with each new intersection on a new line. The weight file format is specified as the number of weights followed by each intersection of the lines ID's and the weight factor, each on a new line.

```
<count>,
<ID 1>,<x co-ordinate>,<y co-ordinate>
<ID 2>,<x co-ordinate>,<y co-ordinate>
.
.
.
<ID n>,<x co-ordinate>,<y co-ordinate>
```

Figure 5.4 Intersection file format

```
<count>,
<intersection ID 1>,<intersection ID 2>,<weight factor>
<intersection ID 2>,<intersection ID 3>,<weight factor>
.
.
.
<intersection ID n-1>,<intersection ID n>,<weight factor>
```

Figure 5.5 Weight file format

The weight factor can be chosen by the user, for example the user could take into account the distance, traffic flow or calculate a route flow weight factor. If the file is not in the correct format an error will be generated. Once both the intersection and weight files have been selected, the map object intersections are displayed. The weight factors between intersections will also be displayed if viewing annotations is selected.

The user can then calculate the shortest route between a randomly generated accident site and the nearest ambulance, as well as the nearest hospital. The shortest route from the nearest ambulance to the accident will be displayed intersection by intersection. Then, the shortest route from the accident to the nearest hospital will be displayed intersection by intersection.

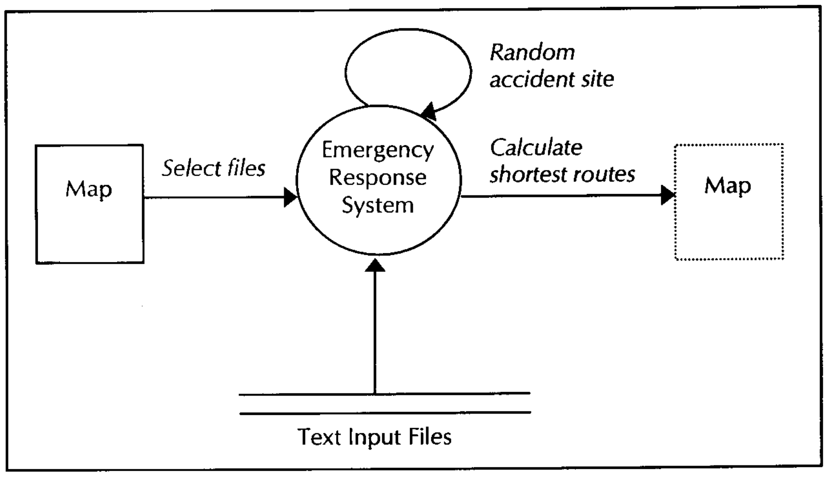


Figure 5.6 Manipulating map objects

Selecting map objects

When the user selects objects on the map, the user can adjust the map display to zoom to the selected objects. The map object is selected by the object's key in the attribute database. Therefore, if the user selects a map object that contains multiple points, each of which has the same key, each of those points will be selected. Similarly, if there are multiple objects in the map with the same key, selecting one of those objects will select all objects that share that same key.

The user can select individual map objects, by clicking on them, which will in turn deselect the previously selected objects.

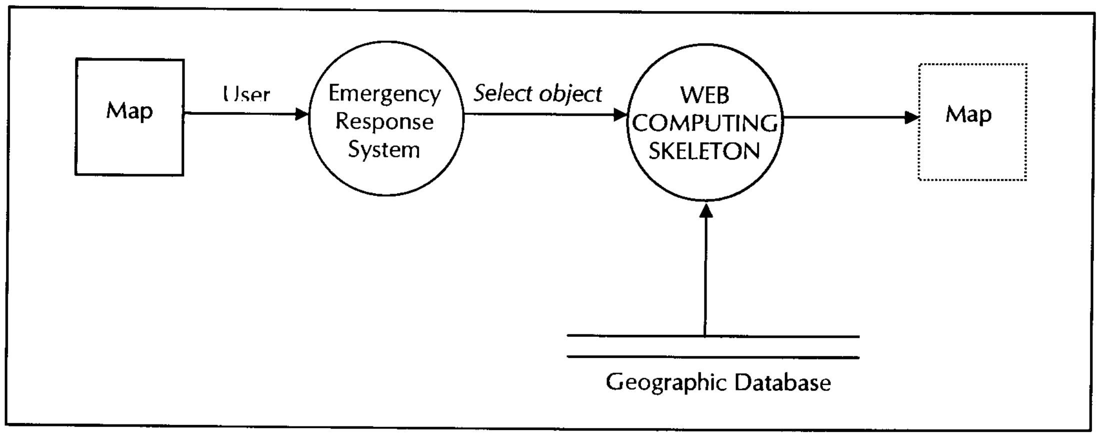


Figure 5.7 Selecting map objects

Manipulating map layers

The user can control the way layers are displayed. To display or hide a layer, the user turns the layer on or off. This is useful for focusing on exactly the objects the user wants to see. To turn on a layer, the user selects the check box next to the layer's name in the legend. The objects on that layer display on the map. To turn off a layer, the user clears that layer's check box. The objects on that layer disappear.

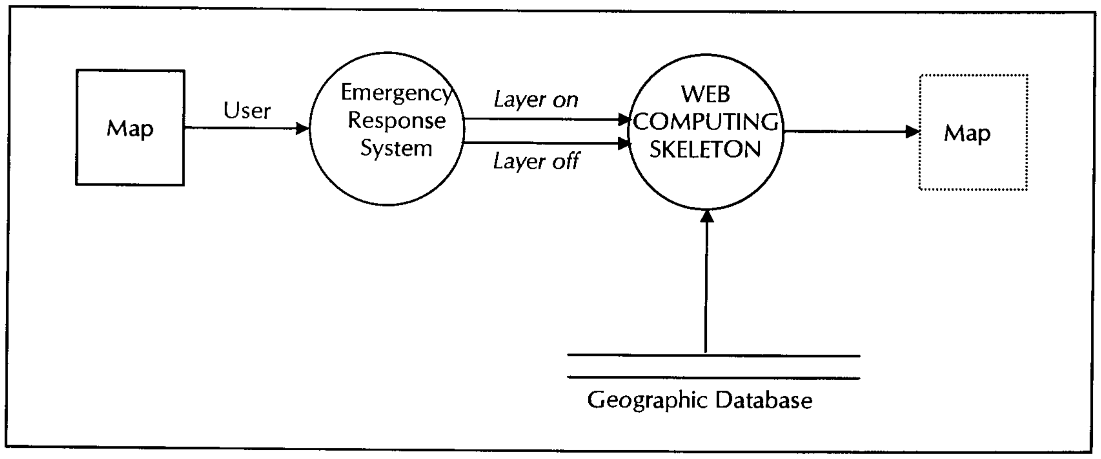


Figure 5.8 Manipulating map layers

Printing a map

The user can zoom or pan to adjust the view in the map window to the area selected to print. Before the user can print the current map view, however, the size of the page to print must be entered. The user can test the output by printing a sample page that shows rectangles of different sizes to determine the best printing size. The user can then use these dimensions to set the page size. Only after setting the page dimensions, can the user specify the print settings to use. The map will be printed according to the current page size settings. To print the entire page and not only the map, the user can print using the Web browser's print options.

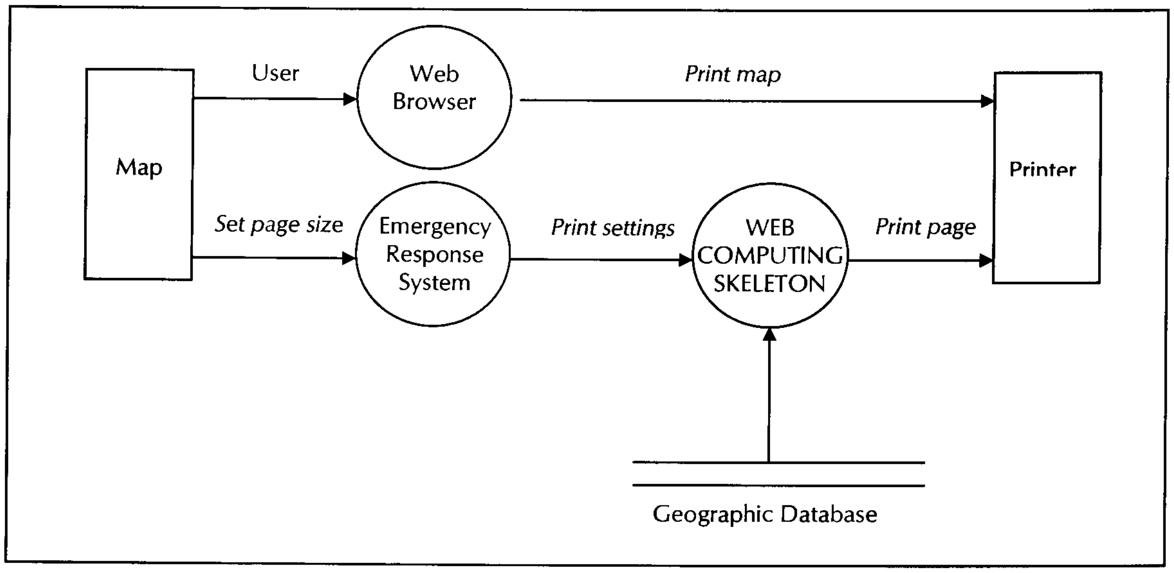


Figure 5.9 Printing a map

5.3.3 Business Rules

The Emergency Response System should allow the user to zoom in and out on a map, find specific objects, select objects that meet certain criteria, display information about selected objects, print the current view of the map, and more.

Unlike a typical map, which just shows spatial data such as roads, cities, and country borders, the GIS links attribute data like population statistics to the spatial data. This link between the map data and the attribute data makes the GIS a powerful tool to manipulate and display data. The user can manipulate the map and the data to specify what to display and print it.

The Emergency Response System must allow users to read maps in various external formats and access various types of attribute data, which could have been created in a variety of DBMS. The map author can create display themes that vary the display of map objects according to certain values. For example, one theme might display countries in a range of colours to indicate population, which allows the user to quickly locate the countries with the lowest or highest populations. The author also specifies the data the user can access and to what degree the user can zoom in and get detail. The system has to cater for maps that vary depending on how the author created the map.

The data in a map is organised into map layers. Layers are like transparencies. Each layer contains different drawings, but when they are stacked together, all of them can be seen at once as a single map. For example, there might be one layer that contains roads, another that contains cities, and another that contains state boundaries. The system must allow the user to turn on all of the layers to see a single map with all of the data: roads, cities, and states. To see the states and cities only, the user can turn off the roads layer to hide the roads.

Different types of map data can also be isolated into separate layers, and even separate different levels of detail into separate layers. Because the map data is on separate layers, users can select map objects from a specific layer without selecting overlapping objects from other layers. Thus, the organisation of map data into layers improves the ability of the user to work with the data while still displaying the data as a single map.

Issues such as privilege, authority and responsibility for particular tasks will be discussed. These issues are important because of their impact on the Internet environment, where access to the applet must have strict security controls.

5.3.4 User Style

The Emergency Response System's user interface provides visible, concrete and frequent points of closure. Each sub-task within a task's task flow has to be completed before the next task can be started. To ensure that each task reaches resolution, various techniques have been used.

- *Modal dialogs* - Modal dialogs are used where information has to be entered before a task can be completed, for example the scale factor to zoom to. The user receives affirmative feedback that closure has occurred by explicitly dismissing the modal dialog.
- *Enabling/disabling menu items* - Menu items are disabled where other tasks need to be completed first. This ensures that the user completes tasks in the order stipulated by the business rules. For example, all the zooming functions are disabled if a map has not been selected and opened. Resolution is implicitly indicated by enabling all the other menu items that can now be used.
- *Menu commands* - Popup menu commands are used to break down tasks into sub-tasks. For example, zoom in, zoom out and zoom all are all tool tasks, where no specific ordering is indicated. Resolution is reached implicitly when the command is executed by using a wait cursor to indicate that the system is processing the data.

Users can save work in progress, by saving changes made to the map at any time. This saves all changes to the current map, including any alterations to the map view. All business tasks have been broken down into smaller tasks which can all be closed as required. Serial closure is used, whereby only the most recent task step has to be closed.

The degree of freedom the user has depends on technical limits, business and process dependencies and user limits and expectations.

- *Technical limits* - The Java environment imposes several limitations on the degree of freedom. The most notable is printing, cutting, coping and pasting. Because Java is used as the development platform, the user cannot cut, copy or paste data displayed within the applet directly. Instead, the user has to use the Web browser's facilities. To print a map, the user first has to determine the page size and save the print settings.
- *Process dependencies* - Several process dependencies have been identified in the task flow that restrict the user's degree of freedom. These restrictions eliminate paths of execution that do not lead to closure.
- *User limits and expectations* - To simplify complex tasks and increase consistency of usage, tasks are divided into sub-tasks that can be accessed via popup menu commands.

Dependencies between processes that were identified in the task flow are expressed explicitly through use of modal dialogs and disabling and enabling menu. This increases the visibility of the user interface.

5.3.5 Dialogs and Dialog flow

The dialog flow of each dialog identified by the system will now be shown. The possible paths or sequences of windows that a user passes through before resolving a dialog constitutes the dialog flow. Elements of dialog flow include the number, arrangement and modality of windows.

Opening a map

Map data is displayed in a document window. The map window is modeless, resizable, minimisable and maximisable. The map window is maximisable so that the amount of map information presented to the user can be increased and vertically and horizontally scrolled.

To open a map, the Emergency Response System uses a subject-oriented dialog, where the user selects the geographic data or map, from a list, works on it and returns to the list of maps. The dialog includes a filter that allows the user to limit the geographic database files presented in the list. Selection from the list takes place on a single File Open window to allow the user a more efficient task flow.

The File Open dialog is a modal dialog that allows the user to select the directory containing the geographic data. The directory may be located anywhere on the hard drive or network. The user has to complete the action before continuing to ensure that a directory is selected, before any other actions can be performed.
In other words, the dialog is resolved when the user completes the selection and dismisses the dialog box. Resolution is the point of closure for the user.

If the data is in a supported, readable format, the map will be displayed in the map window and a legend will be displayed to the left of the map. A warning message will be displayed is the data format is not supported.

Changing the map view

Zooming in, zooming to a rectangular area, zooming out, zooming to the entire map and viewing annotations are all actions that occur within the map window. Resolution involves selecting the command from the menu bar or popup menu, after which the action is executed by the system. Panning is performed by scrolling the map window vertically or horizontally, to see different portions of the current map.

Zooming to a map scale, requires the user to specify the map scale factor to zoom to. The Zoom Scale dialog box is a modal dialog that contains a scale factor text box, where the user can type in the scale ratio to use. This information has to be collected and confirmed by the user, for the command to be resolved. The map is then displayed at the specified resolution in the map window.

Working with map objects

To select locations for the route calculation, the user has to load the intersections and weights ASCII comma delimited text files. The File Open modal dialog is used to select the file names of the files. The user has to complete both actions before calculating the shortest route. This is the point of closure before the shortest route from the nearest ambulance to the accident and the accident to the nearest hospital will be displayed in the map window.

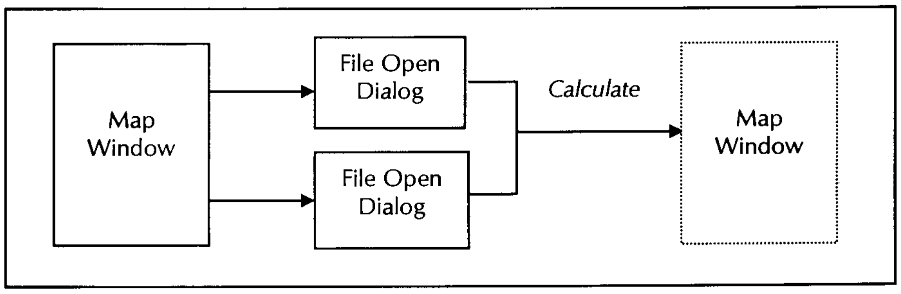


Figure 5.10 Dialog flow to calculate the shortest route

Selecting map objects

To select individual map objects, the point of closure is when the user clicks on the object. This selects the object and deselects the previously selected objects. To deselect an object, the user can select another map object.

Working with layers

To turn on a layer, the user selects the check box next to the layer's name in the legend. The objects on that layer display on the map. To turn off a layer, the user clears that layer's check box. The objects on that layer disappear. Resolution is reached at the point of selection.

Customising the display

To customise the appearance of maps, the user can select a scale factor to use. The Zoom Scale dialog box is a modal dialog box that collects data, the ratio to use, to complete the command. At the point of closure, the map will zoom in or out to the scale specified.

Printing a map

Before the user can print the current map view, the page dimensions and other print settings must be set. The Print Set-up dialog is a modal dialog that requires resolution before the user can continue. The Print dialog collects information about the map to print and prints the map according to the current print settings. The point of closure is when the Print dialog has been resolved.

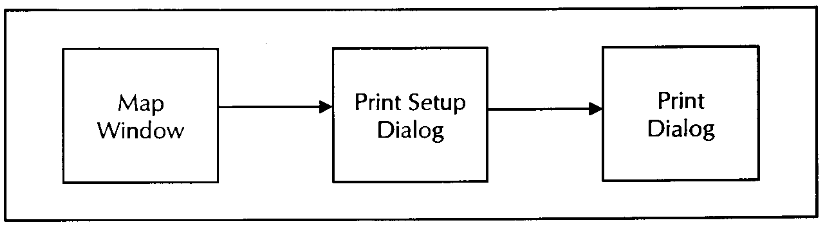


Figure 5.11 Dialog flow to print a map

CHAPTER 6

6. Detailed Design

6.1 Implementation of the Emergency Response System

The Java applet runs in the Web browser, providing a user interface that launches GIS queries and displays these results. The Java applet is embedded in a Web document with code downloaded on demand. Communication with the Java application on the Application Server occurs via an open communication channel using e.g. sockets. The server processes GIS database queries, while the client displays GIS database query results to the user. The communication that occurs between the client and the server must be reliable, no data can be dropped and it must arrive on the client side in the same order that it was sent by the server.

6.1.1 Using Java

[Flanagan, 1999] describes Java as a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi-threaded and dynamic language. Java aims to simplify the development of secure and highly robust applications, while being architecture independent. These features make Java ideal for programming on the Internet. The same Java program can run on all the different types of computers that are connected to the Internet. All kinds of systems can talk to each other, from smart cards to supercomputers, regardless of the underlying hardware or system software

The following are several reasons why the Emergency Response System's GUI is implemented in Java:

- *Java is object-oriented.*
  Most *things* in Java are objects, designed from the ground up. This means that the Object class serves as the root of the class hierarchy. The advantages of an object-oriented application: object-based modularization, abstract data types, automatic memory management, encapsulation, multiple inheritance and polymorphism, simplify development of the GUI.

- *Java is dynamic and interpreted.*
  Each program instruction is translated and executed before the next instruction. This means that classes can be loaded dynamically, a performance advantage for the GUI.

- *Java is platform-independent.*
  Java programs can be run on any platform, without it being necessary to recompile the program with Java's Write Once, Run Anywhere™ capabilities. This eliminates multiple-operating system problems, such as version-control problems and long development times. Because the GUI uses an open spatial query mechanism and runs within a web browser, a distributed, heterogeneous environment, this feature is a necessity for the GUI.

- *Java is distributed.*
  Java provides plenty of support for networking, which is simple to use and effective. RMI allows a Java program to invoke methods of remote Java objects as if they were local. The environment in which the GUI functions is distributed, therefore this feature provides the necessary support.

- *Java is secure.*
  Java provides several layers of security controls that protect against malicious code and allows users to run untrusted programs. This means securely sharing important resources. This is important for networked environments, such as for use on the Internet.

Because the Emergency Response System runs on top of the Web Computing Skeleton [Šerbedzija et. al., 1997], using a distributed, open spatial query mechanism, both of which are implemented in Java, the GUI must also be implemented in Java. The Web Computing Skeleton and distributed open spatial query mechanism are implemented in Java to take advantage of the features described. Also, because the Web Computing Skeleton is implemented on the Web, a distributed heterogeneous environment, Java is an ideal choice.

6.1.2 Java's Abstract Window Toolkit

Java provides an AWT (Abstract Window Toolkit) to perform platform-independent GUI programming that supports event-driven programming. Java interfaces and classes are grouped into packages. This package provides an integrated set of classes to manage user interface components such as windows, dialog boxes, buttons, checkboxes, lists, menus, scrollbars, and text fields.

Features that are supported by Java include [Campione and Walrath, 1998]:
- Simple Types and Statements
- Arrays
- Classes
- Methods
- Interfaces
- Standard Libraries
- Addressing

Java has a large set of GUI elements [Sun Microsystems ^{C}, 2000]:

- *Object*
  Class Object is the root of the class hierarchy. Every class has Object as a superclass parent. All objects, including arrays, implement the methods of this class.

- *Component*
  The Component class is the abstract superclass of many of the Abstract Window Toolkit classes. It represents something that has a position, a size, can be painted on the screen and can receive input events. Primitive objects are directly derived from Component. Components for GUI design include buttons, lists and dialog boxes.

- *Menu Component*
  Menu Component encapsulates the menu design of the GUI. Menu containers are not required to be fully-fledged Container objects.

- *Layout*
  Layout is used for geometry management and controlling the layout of components within their container objects. A layout manager is a class for laying out the components of a Container.

- *Event*
  Event encodes all the event information within the Java environment. Event is a platform-independent class that encapsulates user events from the local Graphical User Interface (GUI) platform.

- *Container*
  Container is the abstract superclass representing all components that can hold other components. Each container may be associated with a LayoutManager instance that determines the position of each of the container's sub-components.

- *Peer*
  Peer objects are used to give the native implementation of that object.

Figure 6.1 illustrates the relationship between the GUI elements within the hierarchy.

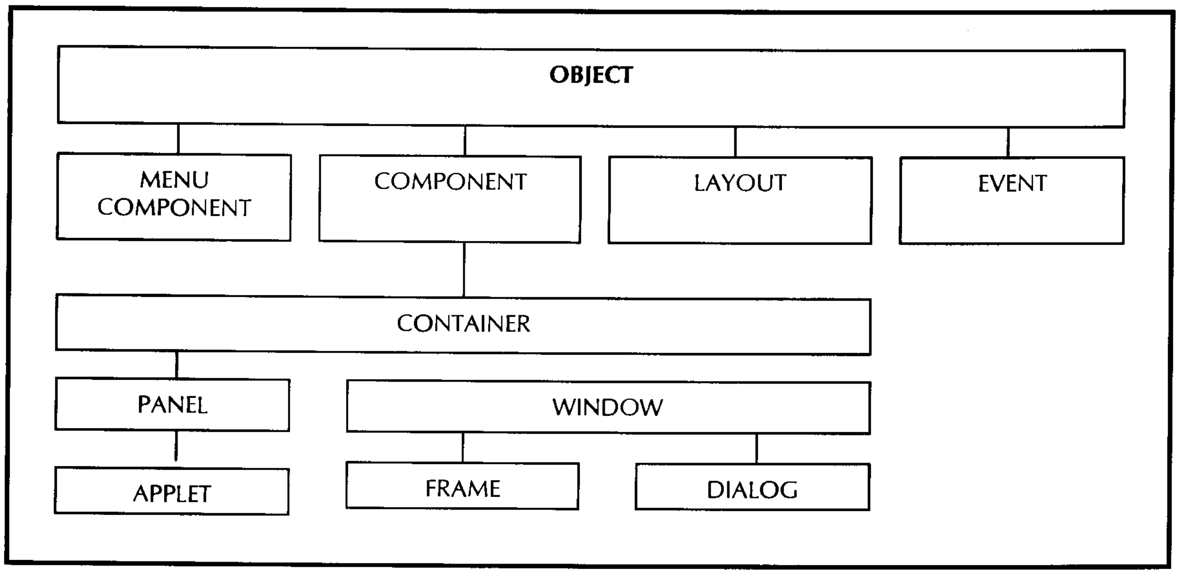


Figure 6.1 GUI hierarchy

The commonly used container subclasses are:

- *Panel*
  A panel is the simplest container class. It provides space into which an application can attach any other component, including other panels. Panel is a container that does not have its own window and is contained within another container. The AWT sends the panel all mouse, keyboard, and focus events that occur over it.
  - *Applet*
    Applet extends the *Panel.* An applet is a mini-application designed to run on a Web browser. It differs from *regular* applications in several ways. One of the most important is security restrictions on what applets are allowed to do. For example, an applet may not access local file systems because these can contain untrusted code. Also, applets have a single entry point – the main () method – from which the program starts running. Also, the applet is not in control of the thread of execution, the applet simply responds when the browser tells it to. An HTML file must reference the applet, for the applet to be displayed.

- *Window*
  A Window is a top-level window; it has no borders and no menu bar. It could be used, for example, to implement a pop-up menu. The AWT sends the window all mouse, keyboard, and focus events that occur over it.
  - *Frame*
    A frame is a top-level window with a title and a border. A frame can also have a menu bar. The AWT sends the frame all mouse, keyboard, and focus events that occur over it.
  - *Dialog*
    This class represents a dialog window, a window that takes input from the user. Dialogs are intended to be temporary windows. They present specific timely information to the user, or they allow the user to specify options for the current operation. The AWT sends the dialog window all mouse, keyboard, and focus events that occur over it.

Java provides a Graphics class to handle the definition of colours, fonts and images. Graphics objects are the keys to all drawing. A disadvantage of Java in this respect is that Java provides no optimisation or three-dimensional capabilities. The Graphics class is the abstract base class for all graphics contexts, which allow an application to draw onto components or onto off-screen images. Most graphics operations performed with a graphics context only modify bits within the graphic context's clipping region.

The Java AWT supports two basic kinds of drawing:
- *Primitive graphics*
  Primitive graphic objects include lines, rectangles, ovals, arcs, polygons and text.
- *Images*
  Images represent all graphical images.

Exception handling is provided by the class Exception. This class and its subclasses are a form of *Throwable* that indicates conditions that a reasonable application might want to catch. *AWTExceptions* are thrown when an Abstract Window Toolkit exception has occurred. An Error is a subclass of *Throwable* that indicates serious problems that a reasonable application should not try to catch. Most such errors are abnormal conditions that should never occur. *AWTErrors* are thrown when a serious Abstract Window Toolkit error has occurred.

6.2 Classes used by the Emergency Response System

6.2.1 The Scroll Pane

The scroll pane handles all output to the screen. Output includes messages, text and spatial data. The scroll pane handles the display of styles, shapes and points. Lines are areas, all of which require world to screen conversion. Other functionality's include setting, getting the scale factor and redrawing.

Also, a custom function implemented by the scroll pane is reading file input contained in text files to calculate the shortest route. The data includes weights (in distance) per node pair and intersection point co-ordinate pairs per node. These are inputted into an array and used to set-up the graph.

The graph set-up includes calculating the nearest intersection to each ambulance and hospital. Using the traffic generator, an accident is randomly generated and the graph then determines the nearest ambulance to the accident (in terms of distance) and the nearest hospital to the accident. The shortest path from the ambulance to the accident is then calculated and displayed dynamically. Next, the shortest path from the accident to the hospital is calculated and displayed.

Functions of the Scroll Pane include:

- *Displaying Maps*

The GUI design centres on the display of spatial data retrieved from the geographic database. This business task requires a sequence of processes that need to be executed. Firstly, before a connection to the geographic database (geobase) can be established, the driver for the proprietary geobase must be retrieved. Before a driver to a certain geobase can be used, it must be loaded. Although this implementation detail is hidden from user, the user does need to select the geobase to be used, which automatically triggers these processes. To display the legend in the proprietary styles, the style library needs to be loaded and initialised. The style library, however, forms part of the geobase and the user does not need to explicitly select it. Once the styles have been loaded the legend will automatically be displayed using the appropriate styles. The spatial data will then be displayed as areas, points, lines and text. Figure 6.2 depicts these processes.

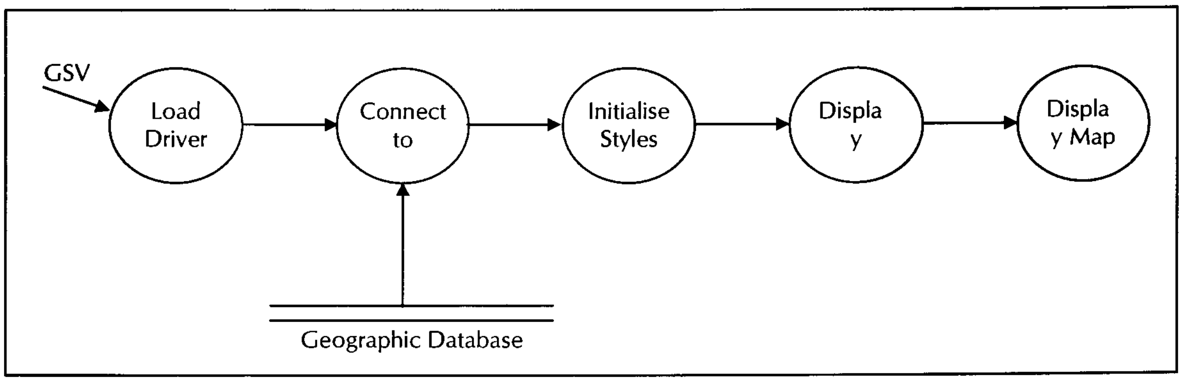


Figure 6.2 Displaying spatial data

Further detail on how the legend and map are supplied in figures 6.3 and 6.4. Although these are not part of the user's task, these processes run in the background when spatial data is displayed on the scroll pane. The legend is made up of the feature class names and displayed by type. Point, line, area and text feature classes are distinguished by the style associated with their type. The feature class colour is obtained from the style.

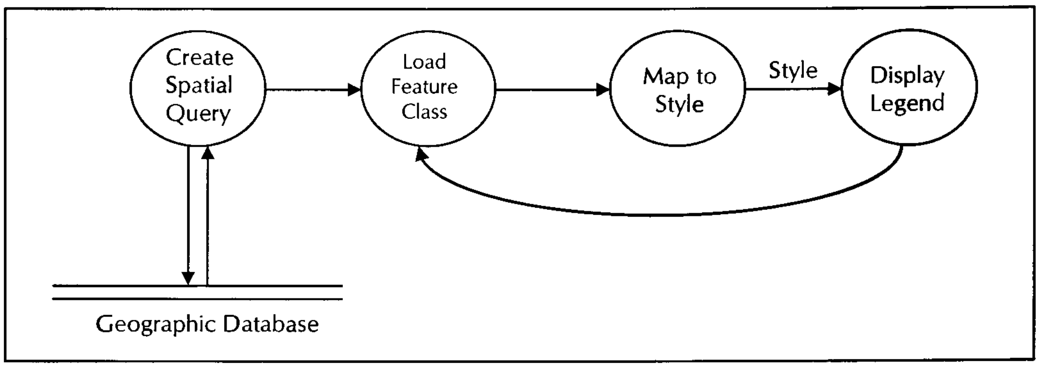


Figure 6.3 Displaying the legend

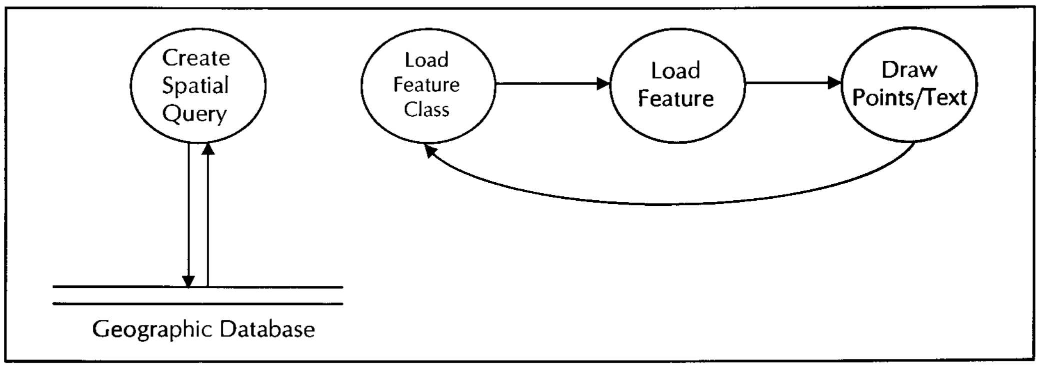


Figure 6.4 Displaying the map

- *Calculating the Shortest Route*

Figure 6.5 shows the calculation of the shortest route in more detail. The user must select an input file containing the node and weight co-ordinates of the ambulance and hospital intersections. An accident site is randomly generated and the shortest route graph is calculated. From the graph, a shortest path array is initialised, that plots the path from the accident to the nearest ambulance and from the accident to the nearest hospital. These paths are then displayed to the user.

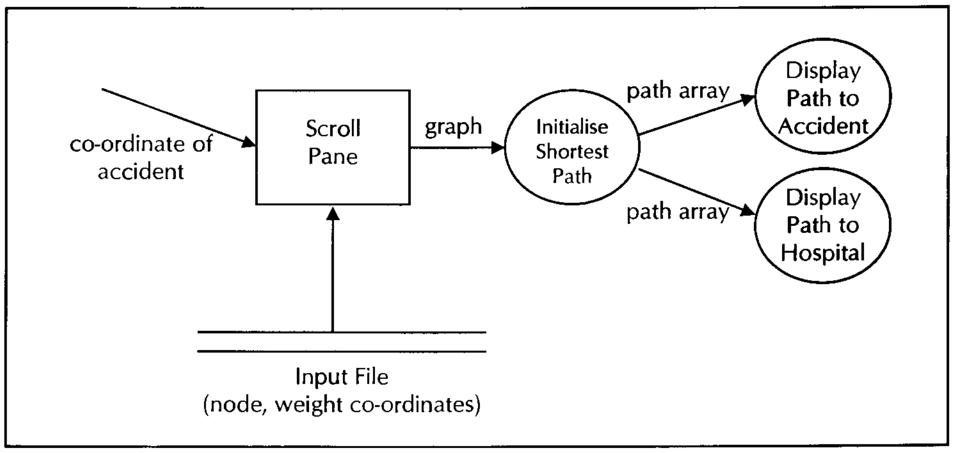


Figure 6.5 Calculating the shortest route

6.2.2 The Control Panel

Figure 6.6 depicts the activities that are controlled by the control panel. These are the business tasks that manipulate and analyse the spatial data. Action listeners handle the user's input when a control is activated. Zoom handles all processes involved in zooming, i.e. zooming in, zooming out, zooming all and custom zooming. Zooming increases or decreases the scale factor of the display by 0.2 or the selected custom scale factor. The user can explicitly refresh the display when an action has caused a redraw event to take place. Also, if the spatial data has been annotated, the user can activate or deactivate these annotations. Once the spatial data has been sufficiently manipulated, the shortest route can be calculated from the nearest ambulance to the randomly generated traffic accident. The shortest route to the nearest hospital can then be traced.

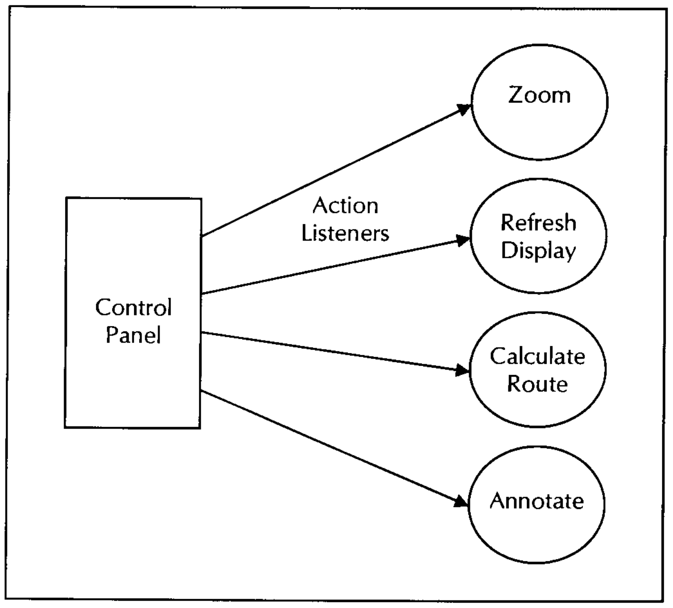


Figure 6.6 The control panel

6.3 Using the GIS Spatial Query Language

The GeoDriver class loads the specific version of the GIS class to be used. The geobase specific DLLs for the driver are loaded, including any proprietary API DLLs. A connection is the established using the GeoConnection class that represents a session to submit queries within the specific geographic database.

The established connection retrieves the meta data on the geobase to which the connection was established. Information includes the version numbers of the geobase, the name and physical location of the geobase, extents, feature classes, styles and shapes. This information is used to initialise the styles to be used for the legend. The legend consists of all the represented feature class names, the feature class types (point, line, area or text) and their styles (colour and font).

To draw the features contained in a feature class, a query is submitted. The GeoQuery class creates the query on the geographic database. The extents of the query are set to ensure that only features that fall within the extents are returned in the result set. The GeoResultSet class accesses the data returned by the GeoQuery class. The feature classes are then traversed and displayed. This method of traversal is faster than index queries. For each feature class, the features are queried and returned in the result set. Each feature class is drawn in the display style associated with the class e.g. fill colour, text font, and text size and font size.

The GeoFeature class represents each feature within a feature class. The features returned in the result set are then traversed. If the selected feature has valid data associated with it, the items within the feature are traversed. Class GeoD3Items represents the geometric items of a feature. A list of GeoD3Items makes up the feature data. The feature data is presented as a GeoObjectList of GeoD3Items.

2D and 3D point items are displayed as points. Ambulance sites and hospitals are determined by their feature class name and represented according to their shape. Shapes are used in styles to specify how point type features are displayed. The list of shapes for each geobase is accessible through the meta data. Lines are treated as a series of point items and are displayed in sequence.

Text items are represented as a point co-ordinate at which the text is defined and an angle at which the text is to be displayed. The text string associated with the item is then displayed at the point position.

CHAPTER 7

7. Performance

7.1 Displaying Spatial Data

To measure the performance of displaying spatial data (or maps), two main factors are considered:

- Processing speed, and
- Display quality

7.1.1 Processing Speed

There are a number of factors that can affect speed. The following are all factors that have influenced the GUI design. Known issues with specific Web browsers (such as Microsoft Internet Explorer), were not considered.

Factors that affect the processing speed when displaying spatial data include:

- The rate of data transfer from the generic GIS classes.
- The rate data is downloaded over the network.
- The speed of execution of the drawing.

The Rate of Data Transfer from the Generic GIS Classes

To graphically display the spatial feature data, display queries are launched, which often involve a large number of features in the geographic database. The number varies according to display variables, such as zoom scale and panning area. The feature data that is needed does not include feature-related information, such as the feature key and class, but does include the geometric data. For display queries, features in the result set are not traversed one by one; instead, one of two types of traversal mechanisms can be used, namely traversing features as linked lists or traversing features as byte streams. Both these traversal mechanisms result in one or less method calls per feature in the result set. Both the linked list and byte stream contain geometric data of more than one feature.

[Coetzee and Bishop ^{A}, 1998] concluded that converting the internal buffer into a linked list takes longer than converting the internal buffer to a Java stream. This means that traversal of features using byte streams is more efficient than traversing features as linked lists.

The Rate Data is downloaded over the Network

Using experimental data to measure the time taken to traverse features using linked lists and byte streams in a geographic database using the distributed open spatial query mechanism with and without RMI, [Coetzee and Bishop ^{A}, 1998] concludes that byte streams require less space than linked lists.

This means that where network traffic is a consideration, traversing features using byte streams provides a performance advantage over using linked lists. The byte stream takes up less space than the linked list, which means that where the distributed open spatial query mechanism is used with RMI a performance advantage for byte streams is gained.

Therefore, byte streams are a more efficient way to traverse the result set where RMI is used, since the smaller byte streams result in less network traffic.

The Speed of Execution of the Drawing

Three further optimisation techniques are used to improve the speed of execution of displaying spatial data, namely displaying spatial data using two-dimensional integer co-ordinates, optimising the pixel distance and redrawing spatial data using bitmapped images.

- *Integer Co-ordinate Optimisation*
  For maximum precision, the co-ordinates of features are stored as three-dimensional floating point co-ordinates in the geographic database. However, the z-co-ordinate stored in a geographic database is not necessary for displaying spatial data on a two-dimensional screen. Also, the x- and y- axis of a two-dimensional screen can be presented as integers, since the three-dimensional floating point co-ordinates retrieved from the result set are converted to screen co-ordinates before they are displayed on screen.

  Using methods of the GeoQuery class to set the ratio between the co-ordinates in the geographic database and the co-ordinates on the display screen before the query is submitted to the geographic database, both linked list and byte stream traversal of the features can be optimised. In both cases, the three-dimensional floating point co-ordinates are converted to two-dimensional integer co-ordinates at the same time that the internal buffer is converted into a linked list or byte stream. The integer co-ordinate optimisation requires extra processing time for the conversion from the three-dimensional to two-dimensional integer co-ordinates, but results in smaller linked lists and byte streams. [Coetzee and Bishop ^{A}, 1998] illustrates that when the network is involved in the distributed open spatial query mechanism with RMI, there is a clear performance advantage of the smaller linked lists and byte streams containing two-dimensional integer co-ordinates instead of three-dimensional floating point co-ordinates.

  The fact that now these feature co-ordinates do not have to be converted to screen co-ordinates by the GUI further improves the speed of execution of drawing.

- *Pixel Distance*
  Another way to reduce the size of the linked lists and byte streams is to specify the pixel distance in the geographic database represented by two pixels (in mm) on the display screen. Since this is the upper limit for display granularity, the result set of the query will contain only co-ordinates further apart than the specified pixel distance. [Coetzee and Bishop ^{A}, 1998] shows that the time (in ms) to traverse all the features in a geographic database, using both traversal mechanisms decreases as the pixel distance is increased. This means that there is a definite performance advantage in using the larger pixel distance.

- *Using Bitmaps*
  To reduce the time taken to display the spatial data when a redraw notification is sent, bitmaps can be used. Instead of redrawing all the features in the geographic database every time a redraw notification is sent, the displayed spatial data can be saved as a bitmap. When a redraw event occurs, the bitmap can be displayed much faster than the time required to redraw every feature in the geographic database. This means that storing and displaying bitmaps is far more efficient than redrawing the spatial data.

7.1.2 Display Quality

Two factors influence the spatial data display quality:

- *Pixel Distance*
  The pixel distance set in the geographic database represented by two pixels (in mm) on the display screen affects the display quality, since the pixel distance is the upper limit for display granularity. This means that setting the pixel distance is a trade-off between the processing speed and the display quality.

- *Bitmap Quality*
  Bitmaps are raster images that are made up of pixels or cells. Raster images display differently from the vector (line and point-based) drawing data in the map, such as roads and cities. To improve the raster image quality, the bitmap resolution can be reset to display more pixels per centimetre.

To summarise, the significant performance improvements include:

- Use of buffered data to download from the network.
- Use of two-dimensional instead of three-dimensional data to decrease the amount of data traffic.
- Drawing execution is improved by using bitmaps for redrawing sections of the screen and for scrolling.

7.2 Calculating the Shortest Route Algorithm

7.2.1 Dijkstra's Depth-First Search

To calculate the shortest path a modified version of Dijkstra's depth-first search was used. Dijkstra's algorithm uses a graph data type. A graph consists of a collection of distinct vertices and a collection of vertex pairs, the edges. In each edge the two vertices must be different. Adjacent vertices are called neighbours. A path is a list of vertices in which each successive pair is an edge. The path length is the number of edges in the path. A graph traversal is an algorithm that accesses each vertex in the graph exactly once. A depth-first traversal is used [Carrano, Helman and Veroff , 1997].

Dijkstra's algorithm finds the shortest path between any 2 vertices. In a connected network, a path between 2 vertices is shortest if the total weight of all the edges in the path is minimal. Dijkstra's algorithm does more than find the shortest path between any pair of vertices, for a given vertex it finds the shortest path to all other vertices. The given vertex becomes the root in a shortest path tree. A shortest path tree contains the shortest path of all the paths in the network from the root to each non-root vertex.

NodeCalculate is where almost all of the dynamic path calculations take place. **NodeCalculate** calculates the shortest paths between one selected node and all other nodes in the graph. It also keeps a record of the first node on the path between each of the two nodes in each of the shortest paths calculated. When **NodeCalculate** has been run on all of the nodes, these nodes (last) comprise a comprehensive retention of the nodes along each shortest path: each node's 'last' for another particular node will lead the computer along a path of 'lasts' until it arrives at its target node. At this point, **NodeCalculate** will also have calculated dynamically all of the shortest paths in the graph.

From the node (k) specified, **NodeCalculate** first interprets the paths leading directly from k, updating **dist** (distance) and last values as it goes. It then adds one node for consideration at a time, checking to see if that node represents a better path to k than the best definite path yet encountered. If it does, **NodeCalculate** updates all of the other best paths to incorporate this new improvement.

The following code extract illustrates the implementation of the modified version of Dijkstra's depth-first search.

```
public void detailsAlg (int i, int j)
{
    // check edge between node i and node j is amongst the edges to
    // choose from during this step of the algorithm
    // check if node j has the next minimal distance to the startnode

    if ( (finaldist[i]!=-1) && (finaldist[j]==-1) )
    {
        if ( (dist[j]==-1) || (dist[j]>=(dist[i]+weight[i][j])) )
        {
            if ( (dist[i]+weight[i][j])<dist[j] )
            {
                changed[j]=true;
                numchanged++;
            }
            dist[j] = dist[i]+weight[i][j];
            if ( (mindist==0) || (dist[j]<mindist) )
            {
                mindist=dist[j];
                minstart=i;
                minend=j;
            }
        }
    }
}

public void endStepAlg()
{
    if ( ( performalg ) && ( mindist == 0) )
    {
        if ( algrthm != null )
        {
            algrthm.stop();
        }

        int nreachable = 0;
        for (int i=0; i<numnodes; i++)
        {
            if (finaldist[i] > 0)
                nreachable++;
        }
    }
}

public void nodeCalculate()
{
    mindist=0;
    minnode=MAXNODES;
    minstart=MAXNODES;
    minend=MAXNODES;

    for( int i = 0; i < MAXNODES; i++ )
    {
        changed[i] = false;
    }

    numchanged = 0;
    neighbours = 0;

    // calculate all edges
    for ( int i = 0; i < numnodes; i++ )
    {
        for ( int j = 0; j < numnodes; j++ )
        {
            if ( weight[i][j] > 0 )
            {
                // if algorithm is running then perform next step for
                // this edge
                if ( performalg )
                {
                    detailsAlg( i, j );
                }
            }
        }

        // finish this step of the algorithm
        if ( performalg )
        {
            endStepAlg();
        }
    }
}
```

7.2.2 Computing Intersections

Intersections are computed using Autodesk World's™ Application Programmer Interface (API) in Visual Basic (VB). The geographic database should be the currently active geographic database, containing all spatial data i.e. area, line, point and textual data. The line features represent a street, secondary road, interchange or highway. The point features represent traffic lights. Textual data constitutes road names.

The steps to calculate the intersections between the line features are described next. Firstly, the active geographic database is retrieved and its line features are recursed to extract all the line features start and end points. A custom filter is then set on the spatial data set using a spatial expression to find all intersecting features for the currently selected feature. The intersecting feature's points are then checked for intersection within a bounding rectangle of the selected feature and the point of intersection is calculated. All intersecting points matching the criteria are then added to the node array that will be used as the graph to calculate the shortest path. Distances are also calculated incrementally as the intersection features are found. This is then used as the weight for the shortest path graph. Weights may be altered to reflect a more realistic traffic flow.

All intersection points and the respective distances between 2 consecutive nodes is then output to a two text files. These are used as input to the Java applet. Using the traffic generator, accidents can then be generated at random intersections and the shortest path can then be calculated between the accident and the nearest ambulance as well as between the ambulance and the nearest hospital.

7.2.3 Processing Speed

Dijkstra's algorithm is a "greedy" algorithm with items on a priority queue that are ordered pairs of intersection points. Priorities are based on the total weights rather than individual edge weights. An array maintains the total weights. When execution of the algorithm has ended, the array will have the correct final values for all indices.

The time required for Dijkstra's algorithm is estimated approximately [Collins, 1992].

The queue size is: $\boxed{E/N}$

where: **N** is for the loop, where a pair is dequeued from the priority queue and neighbours are enqueued
E is for the total number of pairs on the queue

Using a weight list the time to investigate all of vertex k's neighbours is **2E/N** iterations on average. Therefore the total time is approximately:

$$\boxed{N * [O(\log E/N) + 2E/N]}$$

In other words, time is: $\boxed{O(E)}$

where: **E** can range from 1 to N - 1

By substituting, time is: **O(N) to N * (N-1) / 2**

That means, the maximum time is: **O(N)**

This is more efficient than using either a weight matrix or heap-based design which both take **O(N)** time always.

CHAPTER 8

8. Conclusion

8.1 Results

This dissertation has shown the design and implementation of a web-based graphical user interface to display spatial data based on the research of geographical information systems (GISs), graphical user interface (GUI) design and properties of spatial query mechanisms. The GUI was presented as a case study for the Web Computing Skeleton [Šerbedzija et. al., 1997] using the distributed open spatial query mechanism [Coetzee and Bishop ^{A}, 1998] to show how browser-enbled clients can access spatial data worldwide within a heterogeneous distributed environment across a network, thus allowing GIS applications to become usable to a wider audience. Varoius issues were addressed to optimise the performance of the GUI to calculate the shortest route and display the spatial data.

8.2 Future Work

The design of the GUI to display spatial data provides for the range spatial query language properties. The extent of this project includes only the implementation of a subset of these properties, as illustrated in Chapter 5. Further research could implement all spatial query mechanisms to query and analyse spatial data, including thematic maps and a macro language for the user to enter customised queries. This implementation could then be compared with the spatial query criteria in terms of functionality and design principles in terms of efficiency, ease of learning and ease of use.

The main purpose of using the distributed open spatial query mechanism from the Internet is to submit queries to a geographic database to display geographic information. Therefore, the design of the mechanism does not make provision for any updates to a geographic database. This means that the application cannot make provision for the user to update the geographic database, with the exception of specific user settings, such as the zoom scale factor. To enable the development of such an application the design and implementation of the GUI must be extended, using either spatially enabled applications currently available on the Internet and extending the distrbuted open spatial query mechanism.

Also, to provide for a fully functional emergency response system, various additional features can be included. Among these, a fire brigade response sub-system that locates the site of a fire emergency and notifies relevant fire stations of the optimal route plan; or a traffic control centre that contains data on traffic flow problems that can be cleared using an alternate route plan. Further, a shift planning sub-system can be implemented to also allocate various personnel currently on duty within a given region to respond to an emergency situation. The opportunities for extending the application are numerous, especially with the ever increasing popularity of the Internet and World Wide Web as a source of information and communication.

With the increase in the number of spatially enabled applications, the rising popularity of geographic information systems and the ever-increasing demand on graphical user interfaces, the future of these applications seems set.

References

Bernhardsen T, **Geographic Information Systems: An Introduction**, 2nd Edition, John Wiley & Sons, 1999

Campione M and Walrath K, **The Java Tutorial Second Edition: Object-Oriented Programming for the Internet**, Addison-Wesley, 1998

Carrano F, Helman P and Veroff R, **Data Abstraction and Problem Solving With C++: Walls and Mirrors**, 2nd Edition, Addison-Wesley, 1997

Coetzee S and Bishop J [A], *Distributed Spatial Query Mechanisms*, http://www.cs.up.ac.za/Polelo, 1998

Coetzee S and Bishop J [B], A New Way To Query GISs on the Web, *IEEE Software*, May/June 1998, pp. 31 - 40

Collins W J, **Data Structures: An Object-oriented Approach**, Addison-Wesley, Longman, 1992

Cooper A, **About Face: The Essentials of User Interface Design**, IDG Books Worldwide, 1995

Deok-Yoon K, *Geographic Information Systems*, http://dbserver.kaist.ac.kr/NEW/research/doc/gis.html, 1995

Egenhofer M J and Herring J R, Querying a Geographical Information System, *Human Factors in Geographical Information Systems*, Belhaven Press, London, 1993, pp.124 - 135

Flanagan D, **Java in a Nutshell: A Desktop Quick Reference**, 3rd Edition, O'Reilly & Associates, 1999

Frank, A. U. and D. Mark, Language Issues for GIS, *Geographical Information Systems: Principles and Applications*, Longman Scientific and Technical, Volume 1, 1991, pp.147-163,

Maguire D J, An Overview and Definition of GIS, *Geographical Information Systems Principles and Applications*, Longman Scientific & Technical, Essex 1991

Martin A and Eastman D, **The User Interface Design Book for the Applications Programmer**, John Wiley & Son, 1996

McCluskey G, *Remote Method Invocation: Creating Distributed Java-to-Java Applications*, http://developer.java.sun.com/developer/technicalArticles/RMI/CreatingApps/index.html, 2000

Šerbedzija N, Botha L, Abbott A, Bishop J, *Web Computing Skeleton: A Case Study*, Proc. ICSE 97, ACM Press, New York, 1997, pp. 188 - 197

Siebert L and Ballard L, **Making a Good Layout**, 1st Edition, North Light Books, 1992

Sun Microsystems [A], *JDBC™ Technology - Drivers*, http://splash.javasoft.com/jdbc/jdbc.drivers.html, 1 February 2000

Sun Microsystems [B], *JDBC™ Data Access API*, http://java.sun.com/products/jdbc/index.html, 1 February 2000

Sun Microsystems [C], *Java™ 2 Platform, Standard Edition Product Family*, http://java.sun.com/jdk, 24 November 1999

Tognazzini B, **Tog on Interface**, Addison-Wesley, 1995

Tomlin C D, **Geographic Information Systems and Cartographic Modeling**, Prentice-Hall, Englewood Cliffs, 1990

USGS Web Team, U.S. Geological Survey, *Geographic Information Systems*, http://internet.er.usgs.gov/research/gis/title.html, 1997

Zetie C, **Practical User Interface Design: Making GUIs work**, McGraw-Hill International (UK) Ltd., Cambridge, 1995

Glossary

AWT (Abstract Window Toolkit)
A Java toolkit that supports platform-independent GUI programming with event-driven programming.

API (Application Programming Interface)
A formally defined programming language interface that is between a program and the user of a program.

ASCII (American National Standard Code for Information Interchange)
The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

DBMS (Database Management System)
A computer program that manages data by providing the services of centralised control, data independence, and complex physical structures for efficient access, integrity, recovery, concurrency control, privacy, and security.

DLL (Dynamic Link Library)
A file containing a dynamic link routine (DLR) that is linked at load or run time.

GIS (Geographical Information System)
A computer system capable of assembling, storing, manipulating, and displaying geographically referenced information, i.e. data identified according to their locations.

GUI (Graphical User Interface)
A computer interface, consisting of a visual metaphor of a real-world scene, often a desktop.

Internet
Internet refers to the global information system that is logically linked together by a globally unique address space based on the Internet Protocol (IP); is able to support communications using Transmission Control Protocol/Internet Protocol (TCP/IP) or other IP-compatible protocols; and provides high level communication services.

ISO (International Organisation for Standardisation)
An organisation of national standards bodies from various countries established to promote development of standards to facilitate international exchange of goods and services, and develop cooperation in intellectual, scientific, technological, and economic activity.

JDBC (Java Database Connectivity)
Java's standard SQL database access interface. The Java equivalent of ODBC that provides Java developers with a common interface to heterogeneous database management systems.

JDK (Java Development Kit)
Sun's Java programming language and development environment.

ODBC (Open Database Connectivity)
A Microsoft developed C database application programming interface (API) that allows access to database management systems using callable SQL, which does not require the use of a SQL preprocessor. In addition, ODBC provides an architecture which allows users to add modules called database drivers that link the application to their choice of database management systems at run time. This means

applications no longer need to be directly linked to the modules of all the database management systems that are supported.

Operating System
The software that controls the running of programs. An operating system may provide services such as resource allocation, scheduling, input/output (I/O) control, and data management.

RMI (Remote Method Invocation)
RMI enables developers to create distributed Java-to-Java applications and remote procedure calls (RPC).

SQL (Structured Query Language)
An established set of statements used to manage information stored in a database. By using these statements, users can add, delete, or update information in a table, request information through a query, and display the results in a report.

Universal resource locator (URL)
The identifier used by the World Wide Web for the names and locations of objects on the Internet.

VM (Virtual Machine)
A functional simulation of a computer and its associated devices. Each virtual machine is controlled by a suitable operating system.

APPENDIX A: *Features of the Emergency Response System*

Opening a map

To open a map the user selects OPEN from the FILE popup menu, which displays the map, if the format is supported, from any location on the hard drive or network.

Changing the map view

- *Zooming in*
  The user selects ZOOM IN from the TOOLS menu and the map view will zoom in by a magnification factor of two on that point.

- *Zooming to a rectangular area*
  The user can use the mouse pointer to highlight a rectangular area on the map, then zoom in to that area. The user right-clicks on the map and drags the mouse to draw a rectangle over the area to display.
  When the mouse button is released, the rectangle disappears, and the map zooms in to display just the area within the rectangle.

- *Zooming out*
  The user selects ZOOM OUT from the TOOLS menu and the map view will zoom out by a magnification factor of two on that point.

- *Zooming out to the entire map*
  To display the entire map, the user selects ZOOM ALL from the TOOLS menu. This will display the entire map in the original scale factor the map was captured in.

- *Zooming to a map scale*
  The user can specify the map scale factor to zoom to. The user selects CUSTOM from the TOOLS menu to display the Zoom Scale dialog box. In the scale factor text box, the user can type in the scale ratio to use. For example, to zoom in to twice the scale factor, type 2.0 in the text box. The map will zoom in or out to the scale the user specified.

- *Moving up, down, left, or right (pan)*
  The user can slide the map around to display areas that were outside of the previous view, using the horizontal and vertical scroll bars of the map view.

- *Viewing annotations*
  To identify map objects, the user can select ANNOTATE from the TOOLS menu. This displays the map feature's object names. The legend indicates the feature type of the selected object.

Working with map objects

To select locations for the route calculation, the user has to load the intersections and weights ASCII comma delimited text files, by selecting SELECT from the ROUTE menu. If ANNOTATE is selected from the TOOLS menu, the weight factors between intersections will be also be displayed. When the user selects CALCULATE from the ROUTE menu the shortest route from the nearest ambulance to the accident and the accident to the nearest hospital will be displayed.

Selecting map objects

To select individual map objects, the user can click on the object. This selects the object and deselects the previously selected objects. Therefore, to deselect an object, the user can select another map object, by clicking on the object.

Working with layers

To turn on a layer, the user selects the check box next to the layer's name in the legend. The objects on that layer display on the map. To turn off a layer, the user clears that layer's check box. The objects on that layer disappear.

Customising the display

The user can use the Custom command to customise the appearance of maps. The user selects CUSTOM from the TOOLS menu, which will display the Zoom Scale dialog box. The user can enter a selected scale factor or ratio to use. The map will zoom in or out to the scale the user specified.

Printing a map

Before the user can print the current map view, however, the size of the page to print must be entered. The user can test the output by printing a sample page that shows rectangles of different sizes to determine the best printing size. The user can then use these dimensions to set the page size. Only after setting the page dimensions, can the user specify the print settings to use.

To print a map the user selects the PRINT command from the FILE popup menu. Using this command will print the map according to the current page size settings. To print the entire page and not only the map, the user can select the PRINT or PRINT FRAME command of the Web browser.

Java provides no mechanism for copying maps to the clipboard and depending on the user's system, the quality of the colours displayed may vary. Java will recognise changes made to the Regional Settings or Input Locales for the system, as long as the default options are used. Customising the number, time, date, or currency options, however, will not be recognised by Java.

Accessing Commands

The user can access commands in the Emergency Response System using either the toolbar buttons or the map window popup menu. The toolbar buttons appear across the top of the map window:

Toolbar button summary

The following table shows each of the buttons menu commands, and a description of each. The buttons are described in the order they appear on the toolbar, from left to right.

COMMAND	DESCRIPTION
FILE > OPEN	Opens a map in the supported format from any location on the hard drive or network.
FILE > SAVE	Saves all changes made to the active map.
TOOLS > ZOOM IN	Zoom in by a magnification factor of two on the point that you click.
TOOLS > ZOOM OUT	Zoom out by a magnification factor of two on the point that you click to display a larger area of the map.
TOOLS > ANNOTATE	Show or hide the annotations, if any, of the active map.
ROUTE > SELECT	Opens a routing information file for the active geobase from any location.
FILE > PRINT	Prints the current map view to the selected printer.
HELP > HELP TOPICS	Displays the contents and index of help topics available.
CONTEXT SENSISTIVE HELP	Displays context sensitive help where available.

Table A.1 Toolbar button summary

The map window popup menu appears when the user right-clicks anywhere in the map window. The popup menu displays frequently used commands that are context sensitive to the location of the pointer. The user can click on one of the items in the menu to choose that command. Certain menu commands open a secondary popup menu.

Popup menu command summary

The following table describes the commands that may appear in the popup menu when you right-click in the map window.

COMMAND	DESCRIPTION
Print	Print the map.
Print Setup...	Display the Page Setup dialog box, where you can specify settings for printing the map.
Zoom in	Zoom in by a magnification factor of two on the point that you click, or zoom in to the rectangular area you draw with the mouse.
Zoom out	Zoom out by a magnification factor of two from the point you click to display a larger area of the map.

Zoom all	Reload the current map at the current zoom scale.
Annotate	Show or hide the annotations, if any, of the active map.
Select...	Opens a routing information file for the active geobase from any location.
Calculate	Calculates the shortest routes from the randomly generated accident to the nearest ambulance and hospital.
Help Topics	Display the Emergency Response System help contents and index.

Table A.2 Popup menu command summary

The following table describes the commands that may appear in the main menu when the applet is launched.

COMMAND	**DESCRIPTION**
FILE > OPEN	Opens a map in the supported format from any location on the hard drive or network.
FILE > SAVE	Saves all changes made to the active map.
FILE > SAVE AS	Saves the active map with a different name to any location on the hard drive or network.
FILE > PRINT	Prints the current map view to the selected printer.
FILE > PRINT PREVIEW	Displays the current map view as it would appear on a printer.
FILE > PRINT SETUP	Edit the printer and page settings.
EDIT > UNDO	Undoes the previous command, where possible.
TOOLS > ZOOM IN	Zoom in by a magnification factor of two on the point that you click.
TOOLS > ZOOM OUT	Zoom out by a magnification factor of two on the point that you click to display a larger area of the map.
TOOLS > ZOOM ALL	Display the entire map in the current window.
TOOLS > ANNOTATE	Show or hide the annotations, if any, of the active map.
TOOLS > CUSTOM	Customise the scale factor settings for the active map.
ROUTE > SELECT	Opens a routing information file for the active geobase from any location.
ROUTE > CALCULATE	Calculates the shortest routes from the randomly generated accident to the nearest ambulance and hospital.
HELP > HELP TOPICS	Displays the contents and index of help topics available.
HELP > ABOUT ERS	Displays the licensing, copyright and system information of the Emergency Response System.

Table A.3 Main menu command summary

Events handled by the Applet

In the event model of Java 1.1, different classes of events are represented by different Java classes. Most commonly used events are AWT events. Every event has a source object and a type value (this value is used to distinguish between the various types of events of the same event class).

The event-handling model is based on an "event listener". An object interested in receiving events is an event listener. An object interested in generating events maintains a list of listeners to be notified when events occur. They also provide methods to allow listeners to add and remove themselves from this list. When the event source generates an event, it notifies all listener objects that the event has occurred.

- *Mouse down, drag, up:*
  By dragging the mouse the scale factor of the geographic data can be altered. With the release of the mouse button the data is redrawn using the new scale factor. This facility can be used to zoom in or out of a section of spatial data. Data outside of the plot is clipped.
- *Keypress R*
  The plot is rescaled back to the default scale.
- *Keypress r*
  Repaints the data, which is useful when scrolling or rescaling.

Iconic buttons can be used similarly.