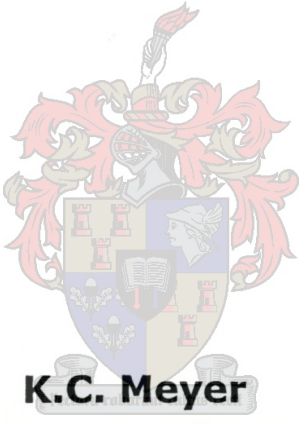
Development of a GIS for Sea Rescue



K.C. Meyer

Thesis presented in partial fulfilment of the requirements for the degree of Master of Arts at the University of Stellenbosch.

Supervisor: Prof. H.L. Zietsman
December 2003

AUTHOR'S DECLARATION

I, the undersigned, hereby declare that the work contained in this thesis is my original work and has not previously in its entirety or in part been submitted at any other university for a degree.

28 June 2003

.....

Kobus C. Meyer Date

ABSTRACT

Saving the life of another person cannot be measured in monetary terms. It is also impossible to describe the satisfaction of carrying out a successful rescue to anybody. However, the disappointment and sense of failure when a rescue mission fails and a life is lost, is devastating. Many rescue workers, including those of the National Sea Rescue Institute (NSRI), have experienced this overwhelming sense of failure. Rescue workers often dwell on failed rescue attempts, wishing that they could have arrived on the scene earlier or knew where to start looking for people. The fact that lives are still lost, despite the best efforts of rescue workers, points to the need to improve on life saving techniques, procedures, equipment and technology.

Providing the NSRI with a workable tool to help them manage and allocate resources, plan a rescue, determine drift speed and distance or create search patterns, may one day be just enough to save one more life.

With this goal in mind, a search and rescue application, called RescueView, was developed utilising ArcView 3.2a. This application was specifically designed for use by the NSRI, and it will be used as a command centre in all NRSI control rooms and for all rescue efforts.

OPSOMMING

Om die lewe van 'n ander persoon te red, kan nie in geldwaarde gemeet word nie. Dit is ook onmoontlik om aan enige iemand die bevrediging van 'n suksesvolle redding te beskryf. Die terleurstelling en gevoel van verlies is egter baie groot wanneer 'n reddingspoging misluk en 'n lewe verloor word. Menige reddingswerkers, insluitend dié van die Nasional Seereddingsinstituut (NSRI), het al hierdie oorweldigende gevoel van mislukking ervaar. Reddingswerkers tob dikwels oor onsuksesvolle reddingspogings en wens dat hulle vroeër op die toneel aangekom het of geweet het waar om vir mense te begin soek. Die feit dat lewensverlies steeds plaasvind, ten spyte van reddingswerkers se beste pogings, dui op die behoefte om lewensreddingstegnieke, -prosedures, -toerusting en –tegnologie te verbeter.

Deur die NSRI met 'n werkbare instrument te voorsien, wat hulle kan help om hulpbronne te bestuur en toe te wys, 'n redding te beplan, dryfspoed en –afstand te bepaal of soekpatrone te skep, mag eendag dalk net genoeg wees om nog 'n lewe te red.

Met hierdie doel in gedagte is RescueView, 'n soek- en reddingsapplikasie, deur middel van ArcView 3.2a ontwikkel. Hierdie applikasie is spesifiek ontwerp vir gebruik deur die NSRI en dit sal as beheersentrum in alle NSRI kontrolekamers en vir alle reddingspogings gebruik word.

ACKNOWLEDGEMENTS

I would like to thank the following people who have been an inspiration in completing this thesis:

Prof H.L Zietsman, for his continued support, guidance and friendship during the years and finally the motivation to complete this thesis.

My wife, for her continued support and motivation to carry on and complete my studies.

My sister, Alma, for her technical and editorial work, as well as my brother Henry, for assistance with publishing this document. The rest of my family, for the words of encouragement - it certainly helped.

I also dedicate this work to the men and woman involved in search and rescue throughout the world. Many people have paid the highest price trying to save and protect the property or lives of others. Still, there is the comfort of knowing that the experience gained and lessons learned were valuable in, later on, saving so many others.

Finally, to the men and women of the NSRI, with whom I have experienced the joy and sorrow of saving lives, I say thank you for being part of a team. I pray that your efforts will be rewarded through the knowledge that you have helped someone in need. Saving someone's life is a privilege and no number of 'thank you's can ever be enough. It is sometimes hard putting in all the effort and not getting recognition or a thank you. Just remember, there is someone out there who is thankful that you managed to save one of his or her loved ones.

ABBREVIATIONS

ATESS	Aerospace and Telecommunications Engineering Support Squadron
DGPS	Differential GPS
DEM	Digital Elevation Models
ECDIS	Electronic Chart Display and Information Systems
ER	Entity Relationship Diagram
ESRI	Environmental Research Systems Institute
GIS	Geographic Information Systems
GPS	Global Positioning System
GUI	Graphical User Interface
MN	Magnetic North
MRCC	Maritime Rescue Coordination Centre
NSRI	National Sea Rescue Institute
RDBMS	Relational Database Management System
RMSE	Root-Mean-Square Error
SASAR	South African Search and Rescue
SAR	Search and Rescue
SOP	Standard Operating Procedures
SFWMD	South Florida Water Management District
TN	True North
UN	United Nations
WTC	World Trade Centre

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 : INTRODUCTION

Public safety and city officials spend massive amounts of time preparing for emergencies, but they don't always happen as planned (Scott III 1998(a):44).

1.1 Introduction

Since prehistoric times, man has had to deal with nature and its forces. The human ability to survive has been continually tested by having to cope with natural disasters like floods, fires and hurricanes. In addition to these natural disasters, industrialisation and technological advances have lead to a whole new range of hazards such as motor vehicle, train and plane accidents. Ironically, although nature is the source of countless threats to humans, natural resources have aided human development tremendously.

However, through economic and engineering developments, man has survived and defied nature to the point that nature itself is now in jeopardy. The major sources of threats to humankind are those that were created by man through technological advancements, causing destruction of the environment and degradation on a global scale (Vorobiev 1998).

The following examples illustrate the threats posed by development. Obtaining fossil fuels has lead to the destruction of the natural resources that have to sustain these technologies, like electricity generation and fuel for vehicles. Gas emissions resulting from industrial processes and other developments have lead to global warming and severe climatic changes. Manmade disasters have even occurred in the search for sustainable energy sources, as was the case in the Three Mile Island and Chernoble nuclear disasters.

Uncontrolled deforestation, large-scale dam building, urbanisation in natural flood zones and the channelling of rivers through manmade waterways have lead to

numerous flash floods. Similarly, continued development and urban sprawl has lead to the settlement of communities within natural forests. To protect property and human lives, unnatural fire prevention without controlled burning has lead to the continued build-up of fuel (dry grass, dead trees or dried vegetation) This, coupled to the lack of rainfall due to climatic changes, has in turn lead to devastating wildfires, often with the loss of life and property.

Fortunately techonology can also be used to the benefit of humans. The field of emergency management developed in response to the natural and manmade threats facing the world today. This field consists of the combined use of computer technology, management principles, training and infrastructure to minimise the impact of emergencies on human lives and property.

GIS has been used in emergency services for several years. Emergency management and response has been revolutionised by the implementation of GIS, an information system that plays an important role in decision making, communication, response times and on-site support of emergency service personnel in vehicles, in the control room and in the effective use of their equipment (Pratt 2001).

Alhtough information on topics such as physical geography, population, political boundaries and infrastructure is available from various source, this information is not always readily available to rescue personnel. In most emergencies there is simply not enough time to gather and analyse information. Prior to the introduction of GIS, response decisions were based on experience of managers or rescue personnel. They often had no choice but to rely on their intuition, which did not always correspond with current information at scene of the emergency.

By utilising information generating tools like GIS, emergency workers can easily access information on current conditions and existing infrastructure. GIS is such a powerful tool that it has been introduced to all levels of emergency management, like

planning, mitigation, preparedness, and response and not only to response and recovery operations (Johnson 2001).

It is possible to have access to strategic information such as the location of the hazards, greatest risk and most valuable resources during the planning phase. With the visualisation capabilities of GIS, data integration and the integration of other technologies like GPS, information can be delivered in a format that can be grasped quickly (Anonymous 2000(a)).“In an emergency, time can be the deadliest enemy. Hours and even seconds can make the difference between saving or losing lives…” (Johnson 2001:11).

1.2 Objectives, approaches and methodology

Saving the life of another person cannot be measured in monetary terms. It is also impossible to describe the satisfaction of carrying out a successful rescue to anybody. However, the disappointment and sense of failure when a rescue mission fails and a life is lost is devastating. Many rescue workers, including those of the National Sea Rescue Institute (NSRI) have experienced this overwhelming sense of failure. Rescue workers often dwell on failed rescue attempts wishing that they could have arrived on the scene earlier or knew where to start looking for people. The fact that lives are still lost despite the best efforts of rescue workers points to the need to improve on life saving techniques, procedures equipment and technology.

This sense of failure, personally experienced by the researcher, lead to the belief that the use of GIS technology, spatial modelling and the right methodology can enable the NSRI and other similar organisations to improve real life rescue coordination, planning, analyses and post operational briefing.

The purpose of this thesis and the application that was developed in fulfilling the requirement of a masters degree is to save lives. Providing the NSRI with a workable

tool to help them manage resources, plan a rescue, allocate resources, determine drift speed and distance, organise search patterns, may one day be just enough to save one more life.

The application RescueView was developed utilising ArcView 3.2a and makes use of its own internal programming language called Avenue. All user interaction and analytical results are performed through the standard functionality of ArcView or by customised functionality specifically developed for this project by the researcher. By utilising this software and its fast programmable features, as well as the grant by Environmental Research Systems Institute (ESRI), the NSRI will be in a position to use this application in every control room and every rescue effort undertaken by them.

This application will become the standard operating procedure to track, manage and coordinate all rescues and will be implemented at all NSRI Sea Rescue stations in the future.

1.3 Research framework

The aim of this thesis is to highlight the importance of a geographically driven response to emergency management in search and rescue and to introduce new technology and methodologies to the current operational procedures undertaken by the NSRI.

Chapter 1 provided an introduction to the emergency management environment. The chapter started by briefly tracing the historical development path of geographic information systems (GIS) and explained what makes up a GIS system and the different data types available in such a system. The chapter also focussed on positional accuracy by introducing global positioning system (GPS), before giving a more detailed description of navigation and tracking movement with GPS. This chapter introduced computer aided dispatch systems, before taking an in-depth look at

GIS in modern emergency management. Emergency management, wildfire GIS Management and crime management through GIS were also analysed. The development of the marine use of GIS, followed by GIS and sea rescue made up the remainder of the chapter, closing with the objectives and methodology as can be seen in the case study diagram, figure 1.1.

Chapter 2 discusses the history of the National Sea Rescue Institute (NSRI) and examines the role that the NSRI plays as the official Search and Rescue organisation of the South African Government. A look into the organisation also sheds some light on its infrastructure, crew and the location of the NSRI's bases through South Africa. A critical look is taken at water sport trends and changes during the last couple of years and its effect on the NSRI call out statistics.

Chapter 3 concentrates on GIS in emergency response with emphasis on spatial management of time, resources and infrastructure, focusing on the new role of GIS and its place in modern search and rescue organisations. Attention is also given to applications and location based emergency management and the current technology and applications available to these organisations. It ends with a look at why GIS has come to the rescue of emergency services.

Chapter 4 deals with the fundamentals requirements of a GIS search and rescue system while also discussing the concept and design behind the GIS application developed for the NSRI, called RescueView. In this chapter nautical terminology is explained along with data requirements for such a rescue system. An in-depth look into application design concepts and integrated data analyses for rescue systems form an integral part of this chapter. Also dealt with in this section is the RescueView system design, detailing definitions and formulas used in the system, operational functionality, spatial time management and lastly the logging and report-back features of the system.

Chapter 5 is dedicated to the first operational use of RescueView and is a case study of how such a system can play an important role in the NSRI and search and rescue. The

lessons learnt were introduced into the system and also highlighted functional needs. The case study evaluates an aircraft crash simulation and looks at the scene set-up, role players, timeline and debriefing as done during the simulation case study.

Chapter 6 looks at the success of the system design and operation. It critically evaluates the systems goals and whether they were met. This chapter also deals with possible system functionality enhancements, as well as the possible integration of technology that may be undertaken in the future.

For an understanding of the research framework followed please refer to the detailed case study diagram (Figure 1.1) below:

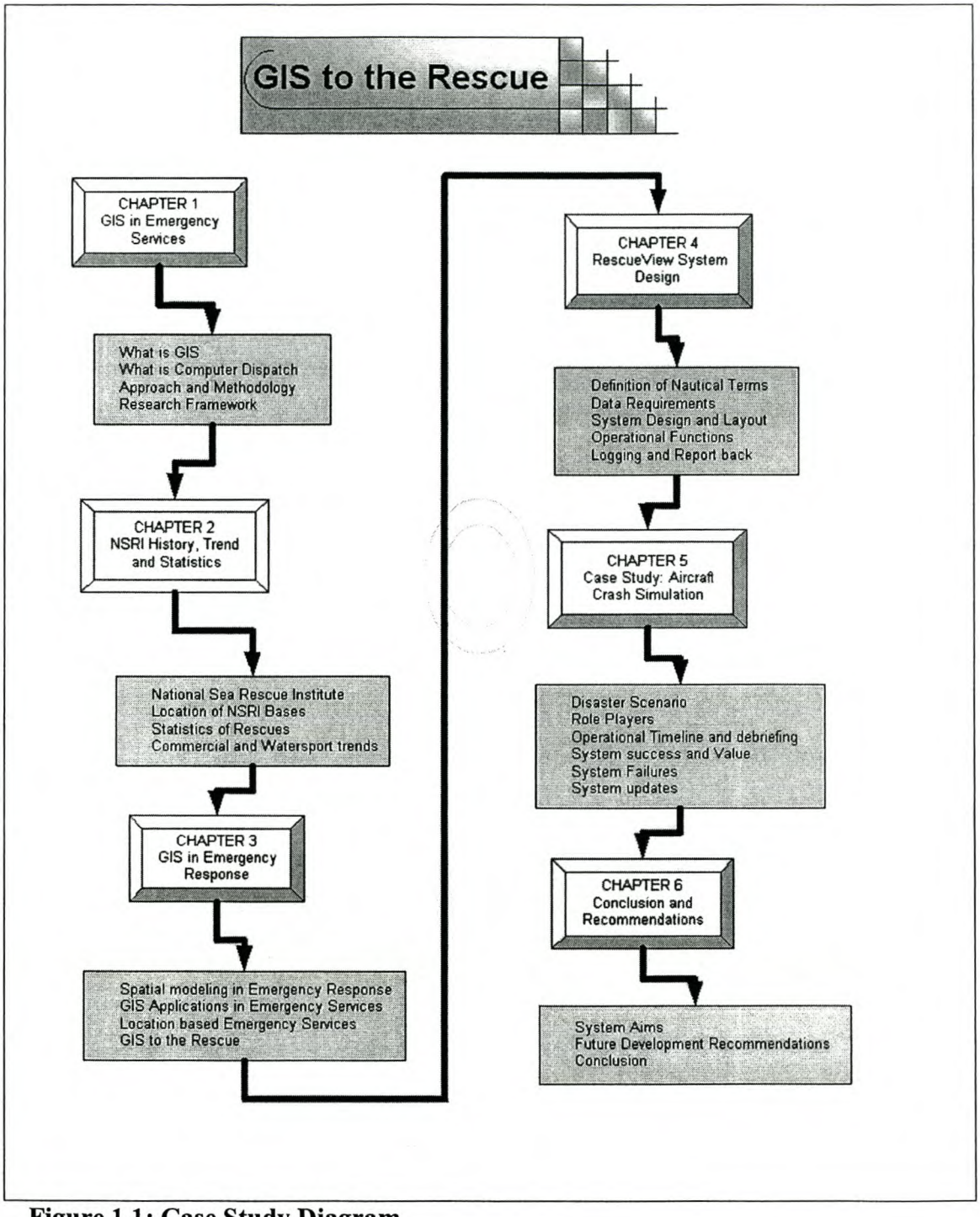


Figure 1.1: Case Study Diagram

CHAPTER 2 : EMERGENCY MANAGEMENT AND GIS

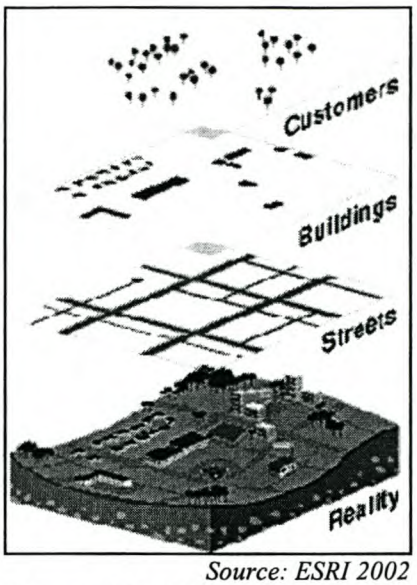
To develop and research a search and rescue application, it is important to understand the principles and building blocks of GIS. The utilisation or manipulation of the system and data needs to be well researched, planned and tested to ensure a successful and trusted system that can be counted apon when lives is at stake.

2.1 What is GIS?

Roger Tomlinson developed GIS in the 1960s when he started using the technology to map out large quantities of information for the Canada Land Inventory. He realised that computers could be used to analyse and map huge amounts of data. These statistical and cost-beneficial tools were used throughout Canada to formulate management plans for the Land Inventory. GIS expanded throughout Canada and the rest of the world despite the initial high cost of computers and limited computing power (Wright 1999).

Special computerised systems and techniques were developed in the mid-1970s to process geographical information. These included techniques for converting information into a digital format and storing the digital data on computer disks or compact devices like CDs or other media storage devices. Techniques were also developed to automate geographic data analyses to search for patterns, measure, optimise routes or predict a scenario's outcome. Several techniques and technologies were also developed for the display of results in the format of maps or tabular data, followed later by advancements in printing and plotting techniques, imagery and multimedia technologies. GIS itself has become more than a software system for processing, storing and analysing geographical data (Bernhardsen 1999).

GIS can be seen as a presentation of layers that represent real world object as can be seen in Figure 2.1. Each layer individually or collectively represent a physical entity, that can be analysed individually or in relationship to others.



Source: ESRI 2002

Figure 2.1: GIS consists of Layers

GIS is a collection of people, data, software and hardware to manipulate, analyse, query and display the information that is linked to a particular spatial or geographic location (ESRI 2002). GIS can be seen as a form of information system that has been applied to geographic data, connecting entities and activities, which interact for a specific purpose. The process that is executed on a set of data produces information, which will be useful in decision-making. Geographically (spatial) referenced data as well as non-spatial data are used together to perform spatial analyses, which in turn solve complex planning and management problems. Geographic proximity between entities allows for the connection of information which otherwise would have been unrelated (Klinkenberg & Cowen 2002).

GIS also possesses the following functionalities:

- acquisition and verification;
- compilation;
- storage;
- updating and changing;
- manipulation;
- retrieval and presentation and
- analysis and combination.

These functions form the basis of a GIS system. Its most important characteristic is the spatial relationship that exists between geographical features, linking objects in the database to a specific location, and the attribute (descriptive) data that describes the properties of that feature (Bernhardsen 1999).

2.2 Map analysis and data

GIS analysis can be both complex and simple at the same time. The production of a basic GIS map can be seen as the simplest form of GIS analysis. Having mathematical data models simulating real world environments is more complex. The types of data used in map analysis can be categorised by its purpose and cartographic representation. Data can be grouped, for analytical purposes, into discrete features, continuous phenomena and features summarised by area.

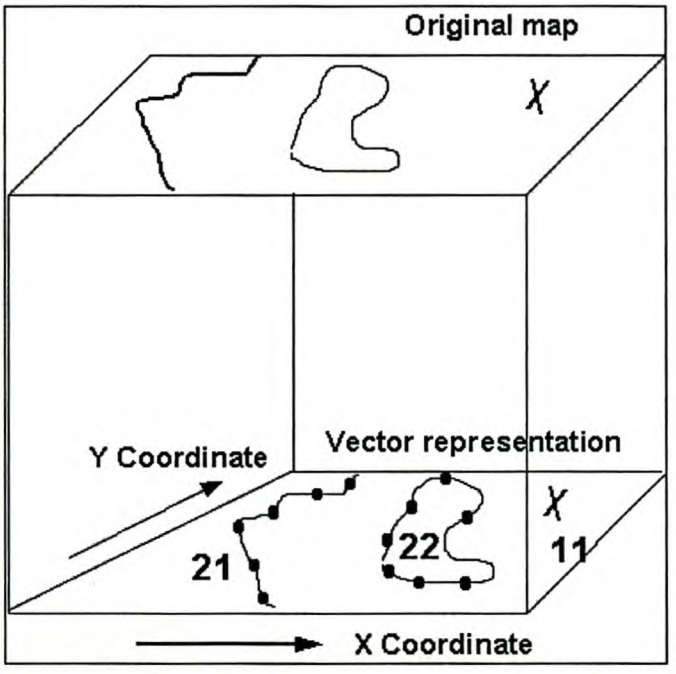
- *Discrete features*: Objects requiring pinpoint locational accuracy, where exact location is very important, can be classified as discrete features. Moving the object changes its location and therefore its relationship with others (Mitchell 1999). In an emergency response environment, location based services like vehicle and resource tracking is primarily based on discrete features. The use of location-based services has been greatly advanced with the help of GPS systems and its integration into modern day navigation systems and procedures.

- *Continuous phenomena:* Data depicted as surfaces, for instance elevation display, toxic plume dispersion or the thermal analysis of a fire can be assigned to this category. Continuous surfaces are created by a series of sample points that are either evenly spaced, for instance in a surface elevation model, or unevenly in surface monitoring stations for weather or by fireman applying thermal imaging systems. Continuous data can also be created by interpolating point data, where missing values are created based on their location relative to other values at known locations. Other data that can be represented by continuous phenomena is vegetation surfaces or soil types (Mitchell 1999).

- *Features summarised by area:* Data that represents a particular feature's density or count within certain set boundaries can be classified as being summarised by area. Data that falls into this category can be census information enclosed by suburb boundaries, or total stream length of a river system within a particular watershed. Totals or percentages are used to analyse data in this category (Mitchell 1999).

2.2.1 Data formats

GIS data is stored into two formats, either vector or raster. Vector data is represented by line type drawings like points (towns), lines (roads) or areas (countries). Raster data consists of image data types, which can include satellite imagery, aerial photographs or scanned background maps. It can even consist of cell based grid models representing digital elevation models, surface utilisation by land-use or vegetation cover.



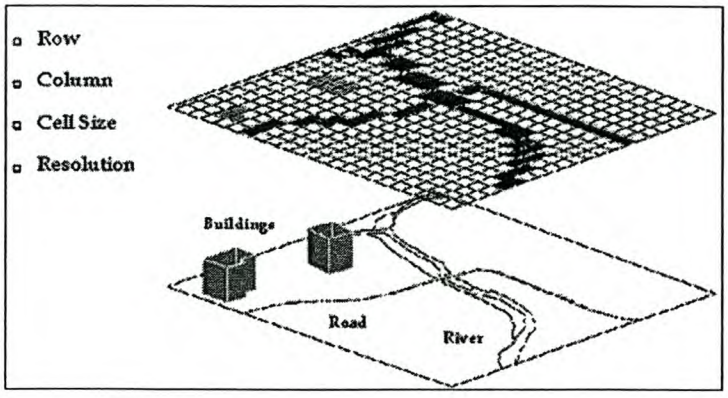
Source: UN ESCAP 2002

Figure 2.2: The world presented through vector data

Vector data, as seen in the Figure 2.2, is predominantly used in conjunction with values that can be attributed to specific features and are stored in associated database tables (attributes). Each feature can have multiple attributes assigned to it. Spatial analyses using vector data are done by comparing its unique spatial relationship properties with other data or datasets, mostly in conjunction with its attribute values. This is achieved by building topological relationship models within datasets to indicate its spatial relationship with other data features. Topological property rules consist of adjacency, intersections, shared boundaries, containment, within distance of, etc.

Another way of representing GIS data is by using raster data sets. Raster data sets consist of cell based (grid) data that is representative of a particular surface phenomena occurring continuously in space. Each grid cell represents a particular attribute value

in the dataset, as can be seen in Figure 2.3 below. Raster data is also determined by the cell size (resolution) allocated to that dataset - a big raster cell size leads to loss of information. Small raster data cells result in very big dataset file sizes, with increased processing time. This can, but does not necessarily lead to more information (Mitchell 1999).



Source: UN ESCAP 2002

Figure 2.3: The world presented through raster data

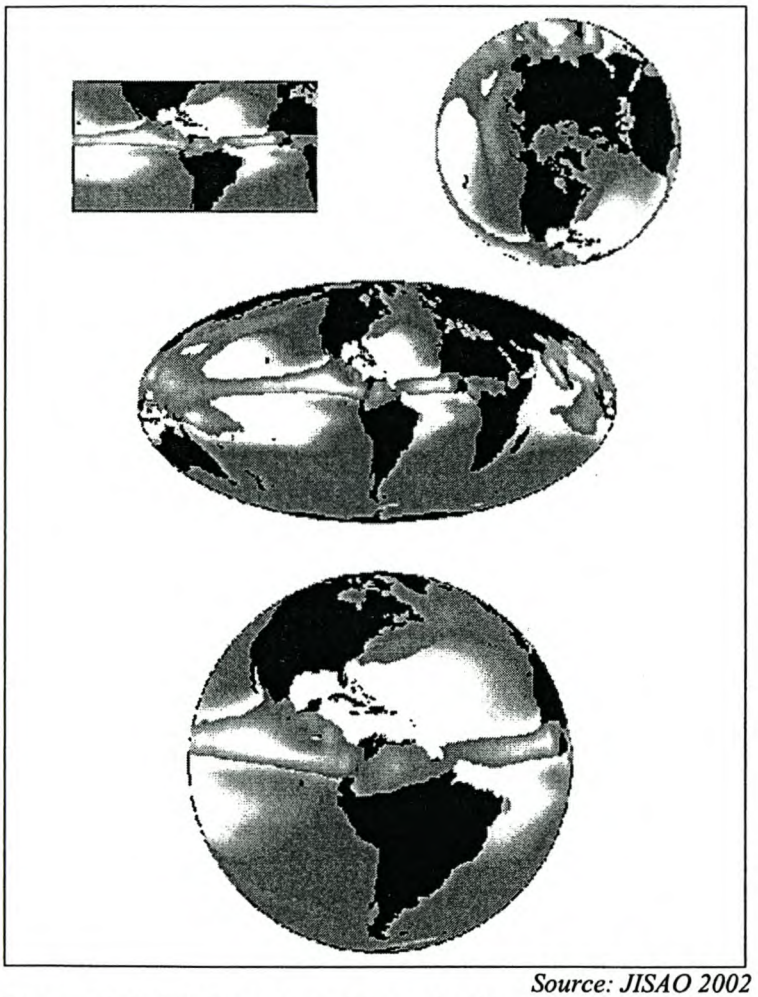
Most GIS datasets consist of a combination of both vector and raster datasets, since each of these have particular advantages and disadvantages. Representing real world features in a GIS has certain limitations, creating the need for a simplified way of storing and manipulating the data.

Reality is an informal (unstructured) system, a system of immense complexity and a system with an infinite amount of information. Representing such a complex and often chaotic system calls for a dataset whose data model and behaviour represent the real world as closely as possible (UN ESCAP 2002).

2.2.2 Coordinate and projection systems

In order for data to be analysed, indicated on a map, or to be evaluated in terms of its interrelationships, a common reference system needs to be employed. To achieve this, a common projection and coordinate reference system must be used to translate known locations onto a map. Figure 2.4 below demonstrates how projection systems will

distort data obtained from a sphere onto a flat surface. The smaller the area represented, the smaller the distortion will be. Coordinate systems are used to represent objects in two-dimensional space. Choosing the correct projection and coordinate systems will depend on the location and size of the area being represented (to name but a few of the factors that need to be taken into consideration) (UN ESCAP 2002).



Source: JISAO 2002

Figure 2.4: Different world projections

2.3 Positional accuracy

Accurate georeferencing and a common coordinate system are vital when determining an object's actual location. The smaller the map scale, the more accurate the position will be, compared to larger-scale maps. Inaccurate data can be costly in many ways, as inaccurate data leads to errors in data analyses, costly decisions, life threatening decisions, oversights of data trends, incorrect conclusions and distrust of other data. As all observations contain an element of uncertainty and positional data inaccuracy, georeferenced digital data is evaluated using statistical error theory by means of the standard deviation (also called the root-mean-square error or RMSE). The fidelity of digital data depends on the accuracy of the georeferencing process, as well as on the source of the digital data, which in turn also has inaccuracies stemming from the original map-making, surveying or digitising process (Bernhardsen 1999).

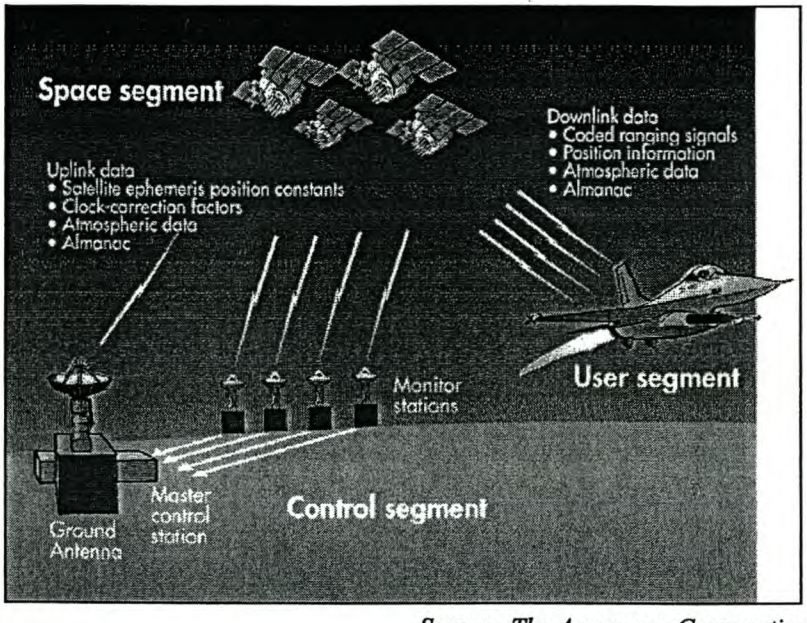
2.4 Global Positioning Systems

The use of Global Positioning Systems (GPS) in disaster and emergency response is vital. The accurate determination of someone or something's position has greatly improved since the inception of GPS systems in the mid 1980s. Originally developed for military use, GPS has taken the civilian market by storm. It is now used in mapping, navigation, surveying, agriculture, construction, vehicle tracking and recovery, archaeology, biological tracking, recreation and many other applications. GPS has developed into a broadband consumer product, and with the advent of personal and small GPS devices, has found its way into mobile phones, private vehicles and recreation. Tracking people, vehicles, planes, ships etc. through GPS has opened a new area of GIS analysis called temporal spatial analysis.

The Global Positioning System consists of twenty-eight earth-orbiting satellites that orbit the earth once every 12 hours at a height of 12 600 miles (20 280 km). In a process known as trilateration, a minimum of three satellites is concurrently used to pinpoint the position of an object on earth. This position is expressed in latitude and

longitude (Steede-Terry 2000). The first satellite was launched on 10 June 1989 and the latest on 30 January 2001. The satellites broadcast positional information on two L-band frequencies namely L1 = 1 575.42 MHz and L2 = 1 227.6 MHz (USNO 2002). A position is determined using three satellites (for position) and a fourth satellite to correct the clock discrepancies between receivers and satellites. With the addition of a fourth satellite, the ability to measure elevation becomes possible. Without it, time differences between inaccurate receivers and accurate GPS satellites can lead to gross errors. A difference of 1/100th of a second can lead to an error of 1 860 miles (8 980 km) without the use of a fourth satellite (Steede-Terry 2000).

GPS consists of three elements, as can be seen in the Figure 2.5 below, namely space (orbiting satellites), user (GPS receivers) and control centres (The Aerospace Corporation 1997).



Source: The Aerospace Corporation

Figure 2.5: Three elements of GPS operation

The control segment comprises five control stations (Hawaii, Kwajalein, Ascension Island, Diego Garcia, Colorado Springs), three ground antennae (Ascension Island, Diego Garcia, Kwajalein) and a master control station (MCS) located at Schriever AFB in Colorado, ensuring the continuous and correct operation of all satellites (USNO 2002).

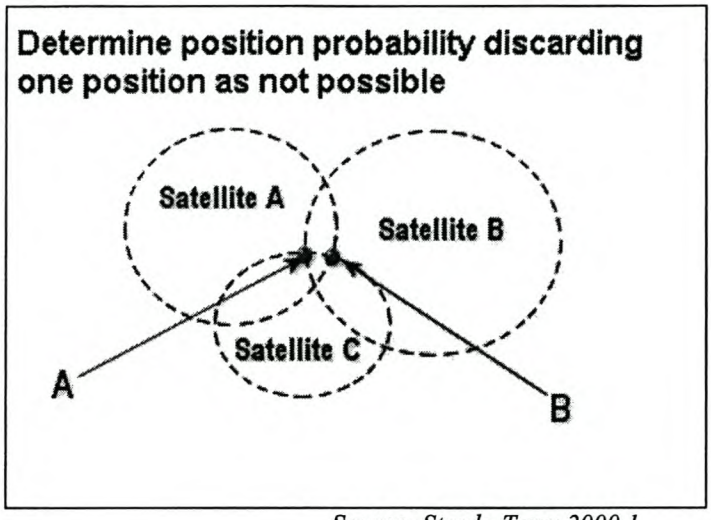
2.4.1 Measuring distance with time

A GPS receiver determines its position by calculating the time from when the satellite sent the signal until the GPS receiver received it, utilising the formula of *distance = time x velocity*.

The two known elements (time and velocity) enable the receiver to calculate its distance from the GPS satellite. Velocity is calculated using the speed of light, which is 186 300 miles (299 800 km) per second (Fowler 1996), which is also the speed the signals travels at. The time, in nanoseconds, is determined by calculating the difference between the signal leaving the GPS satellite and the GPS receiver receiving it. Each GPS satellite is equipped with an atomic clock and every GPS receiver has a quartz clock to ensure accuracy. To determine the time, GPS satellites transmit a pseudo random code with a length of 1 023 bits. The GPS receiver compares this code with its own and looks for inconsistencies, thus determining the time lag for the same code in nanoseconds (Steede-Terry 2000). However, a time discrepancy also occurs since light slows down when travelling through the earth's atmosphere. Small errors are also created through inaccuracy of the transmitted almanac. These two errors lead to an inaccurate distance (up to ±100m) called the pseudorange or pseudodistance (Pendleton 2002).

2.4.2 Calculating position through satellites

With each satellite broadcasting its own uniquely identifiable code (See Figure 2.6), the receiver can determine its distance from that satellite. When the receiver receives the broadcast from the first satellite, it can determine its distance from that satellite, but it is not possible to establish location. When the second GPS signal is received, the area is narrowed to anywhere between the two overlapping areas created by the distance radius of each satellite. A third GPS satellite signal is needed to calculate the intersecting radius of all three signals, resulting in one or two positions. The receiver can evaluate the two positions (A and B) and discard the least likely one (position B for example), positioned somewhere in space. Utilising a fourth satellite augments accuracy and also adds elevation to the latitude and longitude (Steede-Terry 2000).

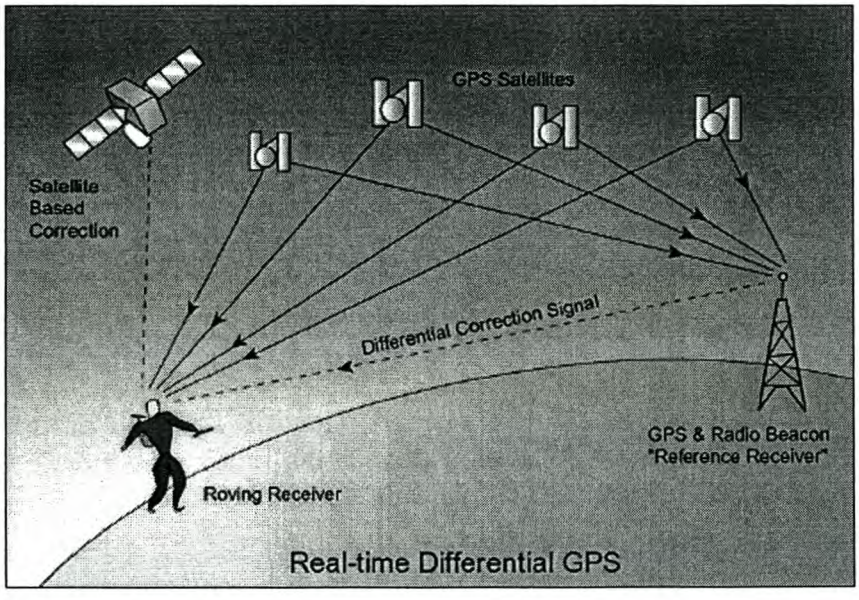


Source: Steede-Terry 2000 1

Figure 2.6: The world presented through raster data

2.4.3 Differential GPS

Differential GPS (DGPS) is achieved by using the known position of one GPS receiver (Base Station) to correct for the position of the roving GPS receiver (See Figure 2.7 below). During propagation of the signal, the effect of atmosphere on the signal can be amplified over the 20 000+ km propagation path. This may introduce errors in the range of metres or even tens of metres. Even a GPS time error, for example 0.1 microseconds in the satellite or receiver clocks, can result in coordinate differences of up to 30 meters. Since the position of both the satellite and base station receiver is known, the difference between the real range and the pseudorange can be calculated. The pseudorange correction for each satellite to the roving receiver is used to determine a more accurate position (Pendleton 2002). This pseudorange error is broadcast from the base station via radio links to the roving GPS and applied to the GPS position to achieve a corrected position, based on the error received.



Source: Pendleton 2002

Figure 2.7: Differential GPS

2.4.4 Post-processing

After DGPS coordinate capturing has been performed, a computer is used to compare the downloaded coordinates of the roving GPS against a file with the known positional errors at the time that capturing took place. This process is known as post-processing. GPS post-processing software is capable of calculating the differences between the satellite position signal and the base station's known coordinates. The GPS base station is capable of capturing raw GPS data and comparing its received position via satellite with its known position, and is thus able to calculate the difference or error between the two positions. The file includes satellite information and observations like GPS satellite identification numbers, pseudodistances, coordinates, GPS time and error in the latitude and longitude.

GPS errors

GPS receivers may also have other errors, which the user needs to compensate for:

- *Atmosphere*: Ionospheric and Tropospheric refraction can delay the signal leading to pseudodistance errors.
- *Multipath*: Reflecting or bouncing of signals not travelling directly to the antennae as a result of buildings, metal objects (railway carriages), tree canopies, mountain ranges etc., can cause pseudodistance errors.
- *Satellite Geometry* (Dilution of Precision or DOP): Poor satellite geometry results in positional errors. This is largely due to the satellite being low on the horizon, forcing the signal to travel through more atmosphere than a signal almost directly above, slowing the signal down and resulting in pseudodistance errors. Satellite geometry can be pre-planned and most receivers have the ability to weed out high DOP values through GPS preferences. DOP errors can be separated into vertical, horizontal, positional (3D) and geometric (with time) errors.
- *Selective Availability*: Selective availability refers to the US government's ability to degrade positional accuracy by slightly changing satellite clocks and the

broadcast satellite position. This feature was disabled on 1 May 2000, but the US government retained the right to reintroduce it without warning.

- *Anti Spoofing*: To prevent tampering with the signal by degrading the P-Code (position), the US government has introduced a Y-Code to replace the P-Code, creating an encrypted code that can only be demodulated through special hardware available only to the US government (Pendleton 2002).

2.5 Navigating and real-time GIS

Since prehistoric times various methods and tools have been used to navigate. Cavemen used stones, twigs and rock paintings to mark trails, and ancient mariners followed the coastline closely to prevent themselves from getting lost. Navigators later discovered that they could chart their course by using maps (See Figure 2.8) and following the stars. Ancient Phoenicians used the North Star to journey from Egypt and Crete and the goddess Athena told Odysseus to "keep the Great Bear on his left" during his travels from Calypso's Island.

Advances in navigation lead to the advent of the magnetic compass, the sextant and a shipboard chronometer. A sextant uses adjustable mirrors to measure the exact angle of the stars, moon, and sun above the horizon. Sextants and chronometers were used in combination to provide latitude and longitude information.

Source: The Aerospace Corporation 1997

Figure 2.8: Early example of map used for navigation

With the 20th century came radio-based navigation systems, which were commonly used during World War II. Ground-based radio-navigation systems had one drawback, however. Two systems generated radio waves at ground level. The one was quite accurate but it didn't cover large areas like high-frequency radio waves (UHF); the other had wide coverage areas and used lower frequency radio waves (AM), but was not very accurate. Launching high-frequency radio transmitters into space led to the start of the GPS era (The Aerospace Corporation 1997).

Paper charts were previously used as navigation tools, but the introduction of satellite position fixing, GPS technology and the use of affordable and small computers lead to Electronic Chart Display and Information Systems (ECDIS) and GPS being used as primary navigation systems. ECDIS systems also allow for colour coded information display to be superimposed on scanned images of nautical charts or others important information. At the same time it displays radar information, vessels' speed and course logs and compass headings coupled with navigational information aids. The system

can also sound alarms or warnings in the event of navigational hazards or possible collisions with other vessels.

A further advantage of ECDIS systems was that it replaced inaccurate positional accuracy maps that were sometimes outdated and inferior, and in conjunction with DGPS, it lead to accurate positional fixing and better collision avoidance, or route tracking and planning. Adding additional information, for instance tides, navigational weather aids or possible navigational hazards to ECDIS systems presented users with a more updated and accurate picture (Ward, Roberts & Furness 1999). With advances in technology and the introduction of mobile and wireless computing, users were able to add real-time weather and tide information while superimposing positional information of other vessels or resources in the area. It allowed for collision avoidance and users' resource optimisation.

GIS enables users to track emergency vehicles like fire engines, police units or ambulances through a busy street network in real time. In a marine environment, it also permits remotely operated and other vessels or objects drifting in the water to be tracked. Real-time GIS tracking provides a spatial context or reference in the ocean, in the absence of landmarks such as roads or mountains.

The value of GIS in visualising the progress and track of a vessel is apparent. It provides an informative "big picture" and with the help of previously captured or remotely sensed data. These base maps can be seamlessly combined to provide the user with an overall view and useful information. Point and click technology can be utilised in planning logistics and track lines, and in creating and managing contingency plans through using existing data. GIS provides an alternative to paper charts, dividers and parallel rulers and enables the user to point and click on the screen to retrieve co-ordinates, measure time and distance, thereby eliminating the time and labour intensive methods of the past.

Recently, real-time data access provided by off the shelf software has lead to features like instantaneous waypoint marking, accurate navigation and automated logging. Real–time data logging is not always possible, but the value of near real-time data is just as important. Software tools or customised applications can be used to import near real-time data and standardised views for members of the same organisation, or even on board a ship.

The continuous development of technology, size reduction of GPS tracking equipment and reduced costs will rapidly expand the use of GIS and GPS technology in marine applications. Advancements in satellite telephone technology also allow for the economical collection of data or vessel tracking from any location. With the addition of internet technology, operations requiring careful data collection and analysis can be managed from a single location through customised GIS technology, anywhere in the world (Hatcher & Maher 1999).

2.6 Tracking movements

GIS is used to track movement by showing the location of a feature at a specific date or time. This movement over time also allows for the path, incremental movement of a discrete feature or the location of geographic phenomena to be analysed. To map discrete or geographic phenomena, indicating the unique location over time and changing its colour to distinguish it from other similar features, allows for a unique image to develop.

To emphasise movement, a line joins discrete locations, showing the path and direction a feature has travelled. Short time intervals will closely reflect the actual path followed by a discrete feature. Using colour coded symbols or changing the symbol or symbol size can also indicate the status of such a feature (Mitchell 1999).

2.7 Computer aided dispatch

The purpose of a caller location system is to identify the location of any incident such as a fire, medical or law enforcement incident or an accident. The precise location of the incident is displayed on a computer aided dispatch system and incorporates several technologies like GIS, automatic location identification or even number location identification. These systems can also store information about the location, work out the shortest or best route to incidents or store related information, which can be retrieved when necessary and showed in conjunction with a map of the area. Information for these systems is mostly stored in a relational database, storing both the spatial and non-spatial information (Shin, Bae, Jeong, Hahm, & Ryu 1999).

Computer aided dispatch applications, incorporating GIS technology, match service calls to appropriate response and manage it. Such a system typically stores information about resources, personnel, deployment plans and event types. After a call is received, the caller location or location of the emergency is identified on a map. Location information is used to search the map and find the applicable location. Call takers question the caller while capturing the relevant information on the dispatch system for use by the emergency service personnel. An event type is coupled to the emergency location. Once the minimum required information is entered into the system, the call is dispatched to the appropriate authority or resource. An event type may also trigger a pre-planned event management and deployment plan.

The system can then be used to allocate available resources or personnel and assign them to a specific event. Once information is passed to the relevant response team, its status will be changed to "*dispatched*" and will no longer be available for other emergencies. The system will monitor the status of all resources throughout, always keeping the dispatcher up to date on its location and status. All information, status or location changes are stored in a database and time-stamped. Once at the scene, the dispatcher will change the status from "*dispatched*" to "*on scene*". Most dispatch systems allocate an icon to every resource and change the icon location and colour based on the resource's current location and status. The controller can determine the

status and location of any resource by simply looking at the map, which tells the whole story (Scott III 1998b).

2.8 GIS in emergency management

Hoetmer (in Cova 1999) defines emergency management as "the discipline and profession of applying science, technology, planning and management to deal with extreme events..." that influence the lives and property of people and disrupt a community. Many of the problems experienced in emergencies or disasters can be spatially categorised, and a spatial framework is invaluable in analysing the problems associated with emergency management. Emergency management GIS application is not an isolated field, but has its roots in other fields of technology, management, application design etc. Managing GIS emergencies is not managing something with a slow onset that one can prepare for, but rather the management of a disaster or emergency, due to a sudden impact or occurrence.

In order to design and examine the role of GIS in emergency management, it is important to have a solid conceptual framework to assist in managing the temporal nature of spatial information in a disaster scenario. It needs to have the ability to do comprehensive emergency management (see Figure 2.9) from risk mapping and hazard analyses (mitigation) to response (emergency activation) and reconstruction (assistance), in a continuous cycle (Cova 1999).

Coming to terms with major catastrophic events such as hurricanes striking densely populated places, earthquakes collapsing infrastructure, or even aircraft crashing into buildings, is difficult. It is always followed by an attempt to understand and make sense of what has happened. It is in this arena that Geographic Information Systems (GIS) has excelled. Its capability to visualise and consolidate data and to convey large volumes of data to many people in a short space of time is what is needed in the aftermath of a catastrophic event. GIS has become a fundamental part of the response effort, being integrated in the standard operating procedures (SOP) of traditional

disaster response. In many instances, GIS response to disasters and emergencies has moved beyond basic map-making into customised GIS applications for very specific tasks, utilised by specialists. These applications have also moved into the internet and wireless arena, allowing GIS capabilities to move beyond office and network connectivity limitations. In the aftermath of a disaster, GIS also plays a role in mapping information to assist in answering questions such as "What happened" and "Where can I help" (Greene 2002).

Jack Dangermond, President of ESRI, describes the role of GIS in disaster response as follows: "In a world increasingly beset by natural and man-made disasters, it's hard to imagine a more striking or beneficial use of GIS than in matters of public safety. When you stop to consider the number and kinds of disasters that happen at any given moment around the world, the elements and conditions that cause them, the factors and the forces that mix and collide as they occur – and the often indescribable chaos and misery that ensues – the idea of a comprehensive and effective system of response starts to look impossible, but that is precisely what GIS is" (Amdahl 2001: Preface). A great deal is learned about disasters through studying them. Public awareness and response are becoming more widespread and strategies for mitigation, response and recovery are being created along a wide front of human activities, like among others, response and service area analyses. Advances in computer technology make the extension and capabilities of GIS even more prominent through GIS applications dealing with emergencies and disasters. Whether working with thousands of fire fighters to manage a wildfire, or determining the best location for paramedic vehicles, or building an overview of community preparedness - GIS plays an integral role (Amdahl 2001).

Emergency management includes doing a hazard analysis, where dangers to the community are identified. This analysis involves the location and identification of potential hazards, monitoring of the hazard and evaluating the degree of vulnerability of the community. Emergency analyses consist of four stages: prevention and mitigation, planning, response and recovery. It is a continuous process (as could be seen in Figure 2.9 below), undertaken before, during and after any emergency.

Prevention and mitigation include activities that identify and prevent threats from becoming hazards to the community, as well as activities that reduce the potential effect of those hazards. In order to reduce and manage this risk, preventative measures can be established. Such measures include the creation of firebreaks, production of operating procedures or approval of relevant legislation.

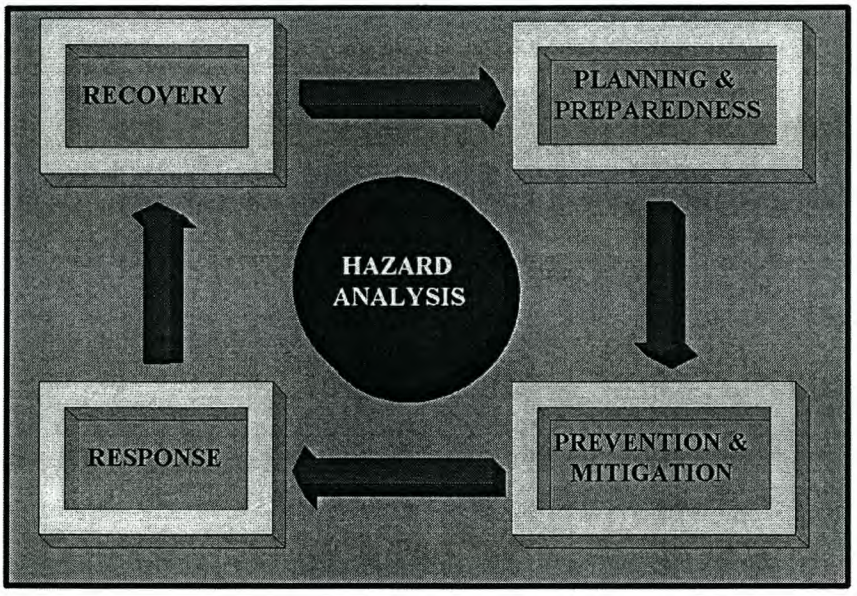


Figure 2.9: Comprehensive Emergency Management

In the event of a risk being unavoidable, proper planning, like the establishment of operating procedures and the placement of equipment and re-sources can ensure preparedness.

Contingency plans, response times, serviceability of an area and preparedness through adequately equipped response teams can further reduce this risk.

The purpose is to limit the impact and threat of the emergency by rapidly deploying trained and equipped response teams. Once on the scene, the proper management and control of the response teams are vital to ensure quick and effective results.

After the immediate hazard has been dealt with, and the risk to the community has been reduced or eliminated, the recovery process can begin. This will incorporate the ability of the community to deal with the events leading up to and including the

disaster. It also involves debriefings of emergency teams, analysing and scrutinising the response effort and its successes and/or failures (Clarsen 1993).

GIS has not met all emergency management needs. Coppock (in Cova 1999) lists five areas in hazard and emergency preparedness where GIS can be improved, namely:

- Lack of data availability, incompatibility and poor quality.
- Difficulty in developing and understanding GIS models and the errors imbedded therein.
- The incapacity and deficiency of off-the-shelf software and applications.
- Failure to meet emergency management user needs.
- Lack of infrastructure and leadership by industry organisations and associations to implement the technology.

The continued use of GIS, remote sensing and telecommunication technologies will lead to further advances in emergency management. Using GIS to inform and educate the public will play an even more important role during and after disastrous events. GIS in emergency management will continue to seek innovative applications and solutions to help solve and manage spatial related emergency management problems (Cova 1999).

2.9 GIS in wildfire fire fighting

The use of geospatial data for fire control and prevention has increased dramatically over the last few years. GIS, GPS and remotely sensed imagery provide tools for firefighters to aid their efforts in the field. These tools are used to map fire perimeters or assist with damage assessment after a fire has been extinguished. GIS analyses are used for high risk and high hazard fire planning and management where projects are implemented to curb or reduce the effects of large fires in specific areas. These

projects include public education, the construction of fire and fuel breaks, clearing of vegetation around buildings or structures or prescribed preventative burning.

The presence of fuel (e.g. vegetation), the location of properties that are at risk, the service levels provided by fire fighters for the area and the weather conditions conducive to fires are used to determine the severity or risk level. The level of hazardous fuel is mapped with GIS technology by dividing vegetation into different types. Data obtained from several agencies are often compared with remotely sensed data, like infrared and spot satellite imagery, for accuracy purposes. Digital elevation models (DEM) are used to determine the steepness of a slope. DEM data depicting the slope percentage is utilised to build fire models. The steeper the slope, the quicker the fire will burn up-slope (Walsh 2000).

One such example is the Marvin County Fire department in California, where they used GIS to map the available service levels during a wildfire. They used the "Behave" fire model to map out the behaviour of possible fires in their area. This model includes the potential loss of structures (homes), the travel times for emergency services to respond to the scene, the historical occurrences of fires and their resistance to control. The "Behave" fire model shows that a fire burning in annual or seasonal grass with a wind of 10 mph (16km/h), and a slope of 10 percent will consume 10 acres in five minutes. Within 10 minutes, the same fire can consume 50 acres of grass. For this reason, it is important to map and model response times using GIS. To illustrate this, Table 2.1 below lists some of the major wildfires in California, showing the number of acres burned and the total number of houses destroyed during that particular fire (Rowan 2000).

Table 2.1: Major wildfires in California

Fire Name	Date	County	Acres burned	Houses Destroyed
Tunnel	October 1991	Alameda	1 600	2 900
Jones	October 1999	Shasta	26 200	954
Paint	September 1990	Santa Barbara	4 900	641
Fountain	August 1992	Shasta	63 960	636
Berkley	September 1923	Alameda	130	584
Bel Air	November 1961	Los Angeles	6 090	484
Laguna	October 1993	Orange	14 437	441
Laguna	September 1970	San Diego	175 425	382
Panorama	November 1980	San Bernardino	23 600	325
Topanga	November 1993	Los Angeles	18 000	323

Source: Rowan 2000

Studies in Marin County, California also show that the most damaging fires occur during severe North-North East wind conditions. However, the complexity of wind models makes it difficult to accurately predict fire behaviour, so other methods are used to assist in building a weather assessment. These include areas above the altitude of the inversion layer, areas under tree canopies, a weather severity index and slope aspect analyses.

The inversion layer is mapped, since Marvin County borders the ocean and the area is often submerged in a dense fog layer, resulting in a smaller fire risk for areas below the inversion layer. Areas above the level of the inversion layer, away from the moisture content of the fog, tend to be very dry and thus present a more predominant fire risk.

Tree canopies shading the ground also produce vegetation fuel moisture, coupled with reduced wind speeds which lead to limited flammable vegetation. The direction and slope also determine the amount of solar heating an area is exposed to, which is why the south and southwest slopes in Marvin County are drier and more prone to burning. Using GIS, Marvin County mapped all major fires in their area from 1992 until 2000,

indicating response times, past ignitions, fire resistance and structure density, in order for them to plan and prioritise controlled burns in the area, thereby reducing the hazard to the community and its assets (Walsh 2000).

2.10 Crime and GIS

Police departments are continuously challenged by rising crime rates and a shortage of manpower. Many of them have specific task teams that focus on problem areas. This is where GIS can play an important role, but only if it can be integrated with existing information and procedures, enabling the spatial analytical capability of GIS to be used in crime analysis (Anderson 1990).

Through creating this kind of sophisticated crime analysis tool, police would be able to better inform officers of events in their area of duty. GIS could also provide management with a better understanding of resource deployment, ensuring the effective placement of officers in terms of high crime locations. It also provides an efficient way of integrating different databases inside the department (Campbell 1992).

The tools that are utilised most often in GIS crime analyses are overlays, address matching and network analyses. Overlay tools are commonly used to summarise point data (crime spots) within polygon areas (police precincts) by using data from independent sources within the same geographic location. Address matching information allows for address based crime information to be matched to linear features (streets) in order to create point features that could be used in overlay functions. This enables better vehicle routing and manages resources allocation in a crime area (Anderson 1990).

GIS can also be used as an accident management system to map and manage the location of traffic collisions along a street network. A transportation information system can be utilised for more than just basic road maintenance by adding

information from law enforcement agencies to the system, for example accident locations, thereby managing and promoting commuter safety. Using historical information (accident type, weather conditions, time of day) in conjunction with strategic information (road type, road condition, road markings) trends can be analysed and preventative measures put in place. It is important that the underlying base map information, on which other location placement data is based, is accurate, as incorrect base data will lead to inaccurate management data (Lithgow 1995).

2.11 Development of marine GIS

Although geography has been an academic subject for a long time, geographers have not shown much interest in the oceans beyond the coastal zone. A geographer, and not an oceanographer, Lt. Matthew Fontaine Maury, published the first book on this topic in 1855 while in the United States Navy's service and it was called "*The Physical Geography of the Sea*". It was only after the end of World War II that people started to take notice and study offshore resources, coastal population and the dangers to the environment caused by industrial growth.

When GIS came into being in the early 1960s, the U.S. National Ocean Survey embarked on an automated cartography effort to map the ocean floor. This lead to a series of matrices of depth values and the production of several nautical charts. Then, during the seventies and eighties, they took a more holistic approach and entered into research on a global scale. The continued development of GIS and other technologies lead to more sophisticated projects being undertaken for ocean data collection and the development of marine GIS.

It was during the nineties, with the advent of earth system science (ESS), that the traditional boundaries of geography were moved. Interdisciplinary cooperation started by focusing on systems like the atmosphere, oceans, ice, biospheres and others, on a global scale. In 1990, the first article written by Manley and Tallat appeared in the Oceanography Society's magazine and focused on the advanced role that GIS could

play in oceanography. The first marine GIS posters were presented at the ESRI user conference in 1991. In 1993, the first marine data sampler CD was released by ESRI. The Manley and Tallat article was followed in 1993 with a feature article in the Sea Technology magazine where the use of GIS was described in terms of its use in search and recovery. These and other oceanographic projects later lead to marine geography and ultimately to marine GIS, and entailed the development of several research initiatives and pioneering efforts.

Increased exposure to marine geography and the use of marine GIS has raised global awareness and concerns about marine pollution, global ecosystems, marine mining and exploration. Developments in marine ecology and the environment have lead to a resolution being passed by the United Nations (UN) in 1994 in terms of the Law of the Sea. During 1995, an entire issue of Marine Geodesy was devoted to marine GIS, which was becoming a mainstream topic at user conferences and feature articles in several marine publications. The UN declared 1998 as the International Year of the Ocean.

Marine GIS has become a well-established domain, and has lead to the application and use in several fields of oceanography or related industries. It also succeeded in taking a predominately land based system and applying its methodologies to a complex 3-dimentional environment. The overall success of marine GIS has been the utilisation of the technology, both data and software, which lead to new applications for modelling, simulation and research methods in marine GIS (Wright 1999).

2.12 GIS and sea rescue

As early as 1996, the Aerospace and Telecommunications Engineering Support Squadron (ATESS) of the Canadian Armed forces based in Trenton, Ontario, developed a GIS based search and rescue (SAR) application called SearchMaster. This system used by the Canadian Rescue Coordination Centres (RCC) utilised off the shelf GIS software, namely ArcView, and with the help of an Oracle database

managed to create a comprehensive SAR system. With ArcView's Avenue programming language and PowerBuilder database front-end development tools, they managed to create a system that was developed to user specifications - and it worked successfully (Aerospace and Telecommunications Engineering Support Squadron 1997).

CHAPTER 3 : NSRI HISTORY, TRENDS AND STATISTICS

The use of Geographic Information Systems (GIS) in emergency services is not new and has been in operation for several years.

3.1. National Sea Rescue Institute

Source: NSRI 2002 (a)

Figure 3.1: The first NSRI craft and crew

The National Sea Rescue Institute was founded on 29 May 1967 in Cape Town several weeks after a small fishing vessel sank of the coast off Stilbaai in the Western Cape. Seventeen crewmembers drowned and sparked a letter by Miss Patti Price to the then Port Captain of Cape Town, Capt. John Payne. Capt. Payne took this letter to the Master Mariners Society of Cape Town during their 22nd Annual Congress, which led to the formation of the sea rescue service.

The first base was located in Three Anchor Bay and was equipped with one 4.7m inflatable boat donated by the Master Mariners Society. The boat was manned by Capt. Bob Deacon and Mr. Ray Lant (See Figure 3.1). Since then, the NSRI has grown to 27 rescue stations, 60 rescue craft and 700 men and women on call 24 hours a day, seven days a week (NSRI 2002(a)).

Source: NSRI 2002(b)

Figure 3.2: A patient being treated by an NSRI volunteer

The NSRI is staffed by unpaid volunteers as can be seen in Figure 3.2. When their services are needed they are called out through a pager

system, leaving whatever they are doing and responding immediately. Highly skilled and competent crews train regularly in the areas of search and rescue, seamanship, navigation, radio operations, fire fighting and first aid (NSRI 2002(b)).

3.2. Location of NSRI bases

The NSRI started out in 1967 with only one station located at Stilbaai on the Cape Coast. By 2002, the NSRI had 28 stations covering the coastline of South Africa from Lamberts Bay on the Cape West Coast to Richards Bay on the Kwa-Zulu/Natal East Coast. The NSRI also has an inland rescue station on the Vaal Dam in Gauteng. Figure 3.3 below depicts what a typical NSRI rescue station looks like, displaying some of the craft allocated to that station.

Source: NSRI 2002 (c)

Figure 3.3: NSRI Station 12 located in Knysna

Volunteers staff all rescue stations. The South African Search and Rescue Organisation coordinates all search and rescue operations, involving other organisations like the South African Airforce, South African Navy, commercial vessels or aircraft. Based at Silvermine, near Cape Town, it acts as an International Maritime Rescue Coordination Centre (MRCC). The local Port Authority coordinates smaller operations where only the station and local rescue services are involved (NSRI 2002(c)).

The NSRI is equipped with state-of-the-art boats (Figure 3.4) and equipment (navigation, location and communication systems), ranging from 3m to 7m inflatable craft and up to 13m self-righting all-weather rescue craft. This equipment can cost anything from R25 000 to R3 million. Small craft are intended for inshore rescue, providing a fast response to swimmers, windsurfers, divers and small boats.

Source: NSRI 2002 (a)

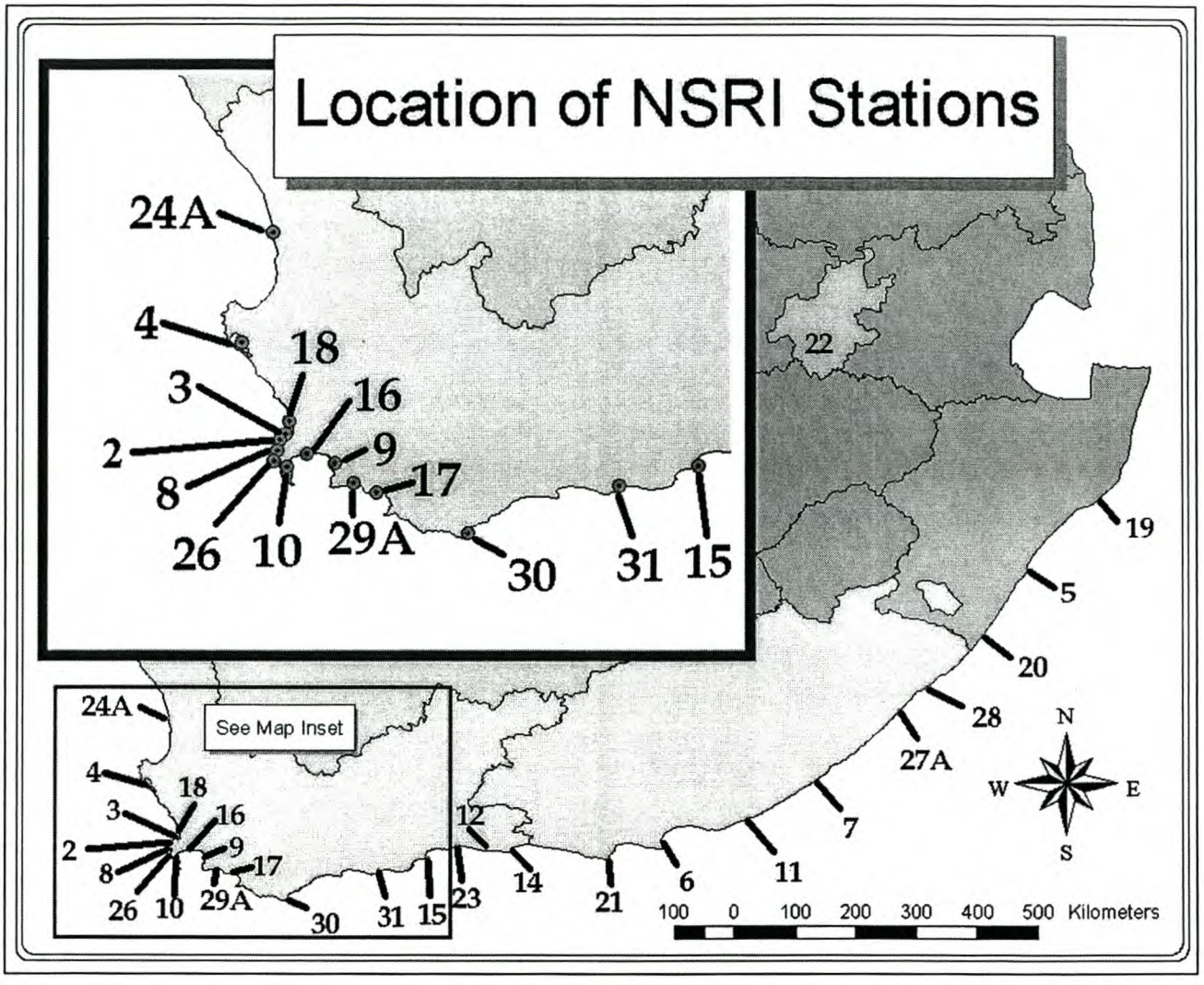
Figure 3.4: Rescue 8 - Hout Bay

Table 3.1 and Figure 3.5 contain a complete listing and map of all twenty-eight NSRI stations currently active within South Africa, with Station 22 being the only inland station, located on the Vaal dam near Johannesburg (Hudson 2002).

Table 3.1: Location of NSRI rescue bases

No.	Name	Location	No.	Name	Location
24A	Lamberts Bay	Western Cape	15	Mossel Bay	Western Cape
4	Saldanha	Western Cape	23	Wilderness	Western Cape
18	Melkbosstrand	Western Cape	12	Knysna	Western Cape
3	Table Bay	Western Cape	14	Plettenberg Bay	Western Cape
2	Bakoven	Western Cape	21	St. Francis Bay	Eastern Cape
8	Hout Bay	Western Cape	6	Port Elizabeth	Eastern Cape
26	Kommetjie	Western Cape	11	Port Alfred	Eastern Cape
10	Simon's Town	Western Cape	7	East London	Eastern Cape
16	Strandfontein	Western Cape	27A	Coffee Bay	Eastern Cape
9	Gordons Bay	Western Cape	28	Port St. Johns	Eastern Cape
29A	Kleinmond	Western Cape	20	Shelly Beach	KwaZulu/Natal
17	Hermanus	Western Cape	5	Durban	KwaZulu/Natal
30	Struisbaai	Western Cape	19	Richards Bay	KwaZulu/Natal
31	Stilbaai	Western Cape	22	Vaal Dam	Gauteng

Source: Hudson 2002



Source: NSRI 2002 (a)

Figure 3.5: Location of NSRI bases in South Africa

3.3. NSRI rescue statistics

Since its inception in 1967 up until the end of August 2000, the NSRI has been involved in 11 227 operations. The operations included the saving of 1 865 lives and also assisting 18 141 people in need of help. During the same period 3 787 boats where towed and a further 2 045 were assisted. A complete summary of these statistics can be viewed in Table 3.2 below.

Table 3.2: Total Operational Statistics

Record of Operations 1967 - August 2002	
Operations	11 227
Lives saved	1 865
Persons assisted	18 141
Boats towed	3 787
Boats assisted	2 045

Source: Jacobs 2002

Examining the statistics for the last five years, 1997 till 2002, it is clear that the NSRI has had an increase in operations, resulting in more people and boats being assisted. From the statistics it is clear that the NSRI has had an increase in operational calls. In fact, 29,24% of the total operational call-outs took place only in the last five years (1997 - 2002). Examine Table 3.3 for complete statistics on the period 1997 – 2002 (Jacobs 2002).

Table 3.3: Five year operational statistics

Record of Operations 1997 - 2002	
Operations	2 383
Lives saved	350
Persons assisted	4 769
Boats towed	686
Boats assisted	550

Source: Jacobs 2002

3.4. Water sport trends in South Africa

In this researcher's experience as a member of the NSRI, water sport trends and thus also rescue trends, have changed significantly for inshore rescue. In the inshore rescue environment, which typically represents all water activities that take place within one nautical mile from the beach, the type of rescues have changed significantly in terms of call volumes and the types of calls received.

During the past six years the researcher has witnessed the change in the popularity of certain water sports . Cost as well as fashion determine which water sports are predominant at a certain stage. In the 1980s windsurfing was one of the most popular sports, which was followed in the early 1990s by motorised inflatable boats called "Rubber Ducks". Towards the end of the twentieth century, sea kayaks and surf skis started appearing on the water in bigger numbers, but it was soon replaced by personal watercraft or jetskis, as they are commonly known. At the time of writing it seems as if kite surfing is the latest trend. Water sport tends to draw international visitors, as South Africa has become better known as a quality international destination.

The popularity of South Africa as a tourist destination and water sport paradise, as well as the dangers that lay within the sea, was evident in a rescue performed by the NSRI in December 2001. Two tourists from Northern Ireland braved the rough seas around Sea Point on a rented jetski. They were thrown from their shared jetski and were swept onto the rocks where they sustained several injuries before being rescued by an NSRI & Metro rescue helicopter. The helicopter had to be used since it was too dangerous in the rough seas for the rescue boat to pluck them from the rocks (Anonymous 2001(f)).

As the water sport fashion changed, so did the type of calls and also the degree of difficulty in performing successful rescues. Equipment used in a sport also plays a role in rescue efforts. The larger the watersport equipment involved, the easier it is to find it. Inflatable craft and windsurfers tended to create a large enough "footprint" on

the water to spot from the air or even the water. Sea kayaks, surf skis and especially kite surfers tend to be the most difficult to find.

Searching for and rescuing kite surfers are difficult for several reasons. When a kite surfer is in trouble, he or she is mostly submerged in the water, with only the head or a small portion of the upper body exposed. With even a small swell, it is very difficult to spot someone in the water unless you are in his or her immediate vicinity. Kite surfers also tend to be far away from the kite or even, if the lines have broken, totally separated from it, especially when the kite gets blown out to sea or away from the scene.

Staying with the equipment or boat after it has capsized can potentially save a life. On many occasions being able to stay on top of the boat or equipment has enabled people to stay afloat and alive, giving rescuers a better chance of finding them (Laganparsad 2002). Some people tend to leave their equipment in the water, trying to swim ashore. This is a huge mistake. In most cases, with currents and swells on a windy day, they don't succeed and end up in even more trouble, compared to those that stayed with their board and kite equipment, making a larger footprint to search for.

Bathers also get into trouble. The NSRI has been called to the scene many times, but with rough seas along the coastline (Peer 2002) and difficult sea currents experienced by swimmers, it is very difficult to find a person, even when using a helicopter (Ajam 2002). Sometimes alcohol plays a role, as several people have drowned or have been killed as a result of alcohol abuse (Govender 1998).

3.5. Rescue of commercial fishing vessels

Since the ocean off our coast is not only used for recreation, but also as a source of income, the NSRI is faced with calls from countless ski- and commercial fishing boats along the shoreline. This has also led to multiple rescues when ski boats lose their

engines and are capsized. During March 2001, this was evident in a rescue performed by Hout Bay's Station 8, when they where called out by a skipper of a ski boat with his cellular phone. The skiboat had lost one of its motors en route to Hout Bay harbour. With the sea and wind picking up, the skiboat later lost the other motor too, and was capsized by the huge swells running off the coast.

The NSRI responded to the scene, but found it difficult to locate the skiboat in the darkness. Fortunately all lives were saved when the boat washed onto the rocks and the men managed to climb to safety, but the boat was destroyed behind them (Kemp 2001). Some cases take a more tragic turn. In October 1998 a father, his son and another crewmember drowned off Gordon's Bay when their skiboat capsized. The NSRI managed to rescue one survivor and only found the body of the father (Anonymous 1998(g)).

3.6. NSRI and Information Technology

It is clear from the operational statistics listed by the NSRI and from articles that appeared in the press that there is a constant need to improve search and rescue techniques to be able to save lives. The use of technology is not new to search and rescue organisations, and neither is it to the NSRI. The implementation of GPS technology has improved the situational awareness of crew to their surroundings. Most GPS systems in use by the NSRI include moving map displays, local waypoints for added situational awareness and even information on the depth of the water.

What is needed in addition to this and other equipment is the use of analytical tools like GIS technology. This would not only improve onscene command and control, but also search pattern planning, location analyses, drift calculation, response times and crew and media debriefing. The use of GIS and in particular the search and rescue application developed for the NSRI will be discussed in the following chapters.

CHAPTER 4 : GIS IN EMERGENCY RESPONSE

Source: GIS Development 2002

Figure 4.1: WTC after being struck by an aeroplane

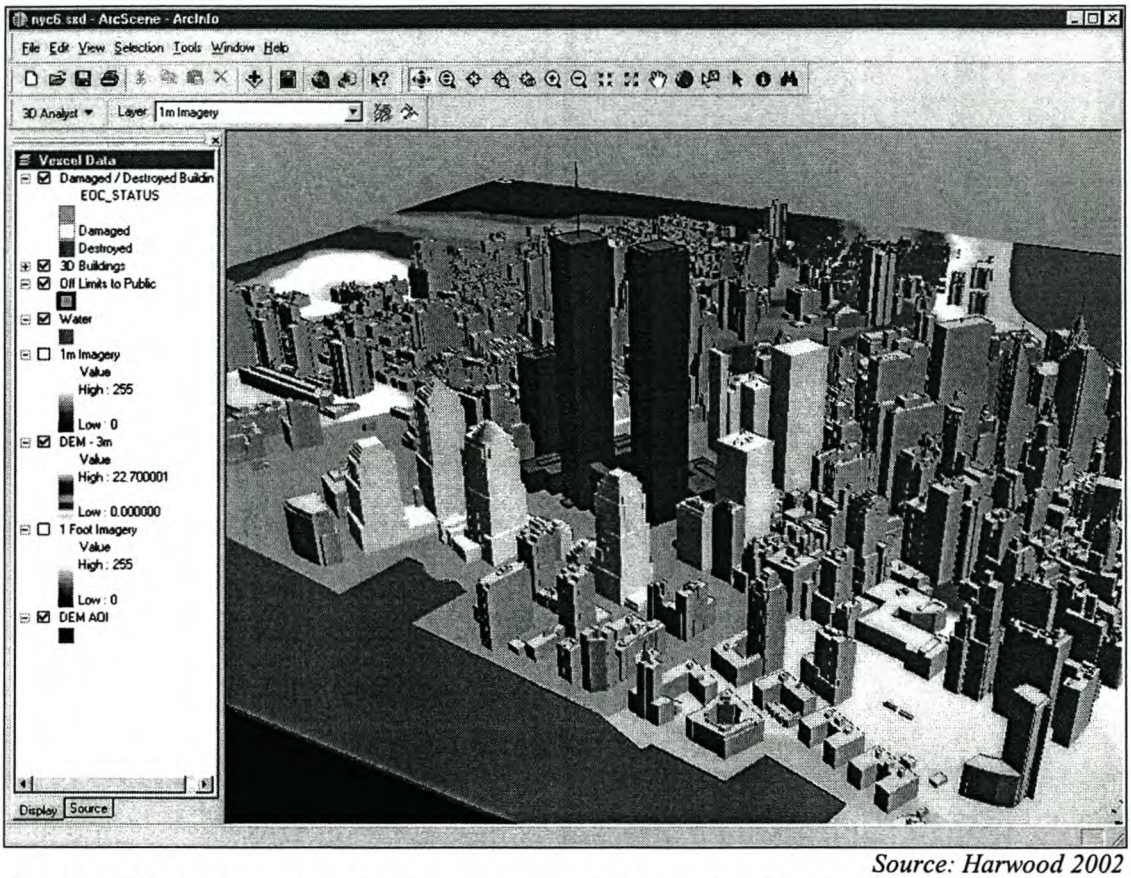
"On 11 September 2001, New York City suffered the largest disaster in United States history when terrorists hijacked two airliners and crashed them into the North and South Towers (Figure 4.1) of the World Trade Centre (WTC), causing the collapse of both towers and killing thousands of people – not only those who worked in the WTC but also fire fighters, law enforcement personnel, and others who initially responded to the incident. Coordinating immediate rescue efforts and long-term rescue recovery plans required sharing of information among responding personnel. " (Pratt 2002).

4.1. Spatial modelling in emergency management

It is not easy to implement GIS solutions for man-made disasters to overcome the crises created by such an event, as was demonstrated with the World Trade Centre attack and its subsequent collapse (GIS Development 2002). The value of GIS was soon demonstrated as it was used to first turn out simple maps of the disaster area followed by more complex and detailed maps as can be seen in Figure 4.2. Ultimately spatial analytical techniques lead to information that could be supplied to rescue workers, city officials and the public (Harwood 2002).

According to Pratt (2000) "Spatial modelling technology is growing like wildfire within the emergency management community". The emergency management community is using spatial technology more and more to help with emergency

management. Real world phenomena and events are simple to understand, predict and manipulate through use of GIS models.



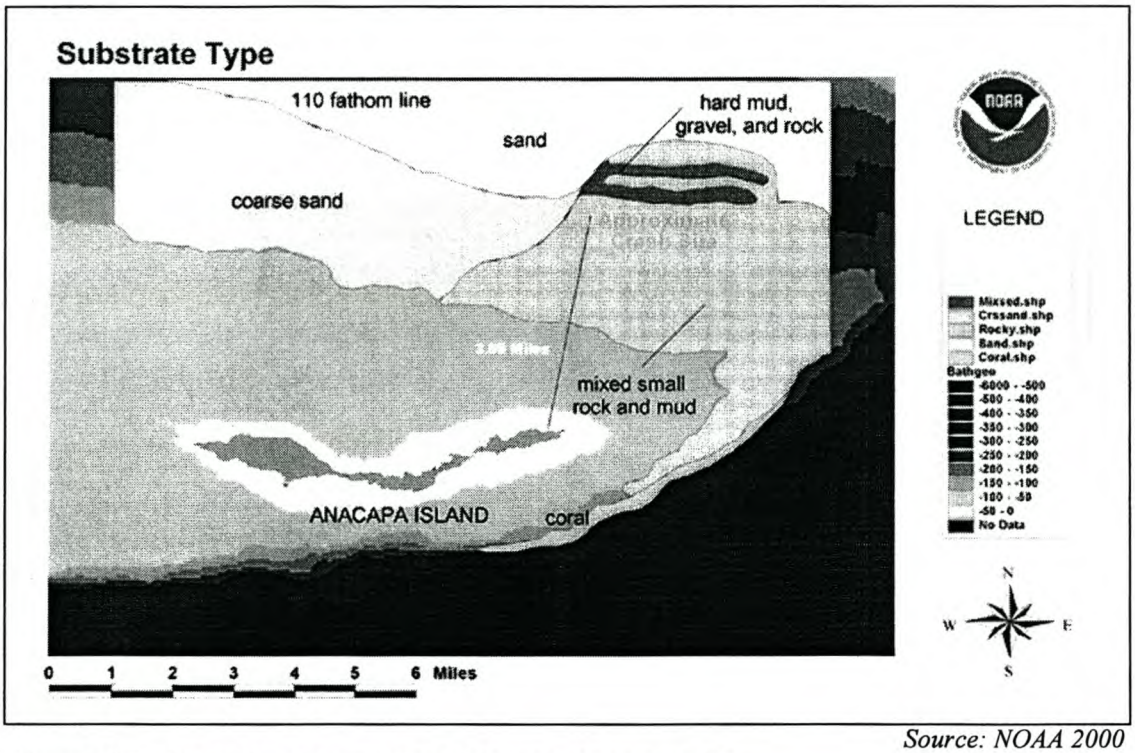
Source: Harwood 2002

Figure 4.2: An example of a 3D model of the WTC created by GIS

A powerful methodology, using data integration and viewing capabilities for spatial modelling, is available through GIS. Data derived from one model can be used as input for another model, using different parameters, thus allowing geographically different located data to be analysed using the same model parameters. Testing "what if" scenarios allow emergency responders to identify specific areas and the formulation of mitigation strategies (Spalding 2000).

On 31 January 2000, Alaska Airlines flight 261 went down in the Channel Islands National Marine Sanctuary. The U.S Coast Guard responded to the incident, and with the help of the sanctuary staff, they utilised GIS for the search of survivors and later for recovery. The data used consisted of bathymetry and topographic, sea surface and marine oceanic data, as can be seen in Figure 4.3 below.

Utilising GIS and sea surface data, they were able to predict where people and debris would drift. Apart from quickly sending maps to people, one of the most important aspects of GIS is its visualising capabilities. To support the emergency services, the sanctuary staff produced a series of maps such as the one in Figure 3.3 below (NOAA 2000).



Source: NOAA 2000

Figure 4.3: Sample map produced by sanctuary staff

4.2. GIS applications for emergency services

The effect of GIS, even in its basic functionality, has a profound impact on emergency response efforts. Basic GIS functionality integrates large quantities of data from varying sources, map scales and map content. These in turn help manage, integrate and display a variety of data sources while maintaining reliability and accuracy of information (Haack-Benedict 1998).

Search and rescue applications utilising GIS and GPS technologies are numerous. In scenarios where search and rescue need to be performed, it is important to show which areas have already been searched and where additional searches have to be done.

GIS-based dispatch applications are numerous and can be found in emergency medical services, police patrol vehicles, public transport services, taxi cabs and express delivery vehicles, to name but a few. They all take advantage of GIS technology for purposes of dispatch and tracking. Most of these systems do not only have a location based capability, but also routing capabilities, allowing for more effective use of the vehicle or resource. By monitoring the current location of resources, the closest resource can be identified quickly so that emergency services can immediately respond to an emergency location. It is important that the routing algorithms used are effective and accurate and that the information it generates is sent out quickly (Chou, Rudd & Pennington 1998).

When a search and rescue team returns from a search area, the rescue coordinator must determine the effectiveness of the search. This is called probability detection. The coordinator's function, using probability detection, must determine the effectiveness of the search team to find a missing person in the area that has been covered, given the conditions at the time.

In 1996 search and rescue dogs were fitted with GPS tracking devices to show the effectiveness of tracking movement by search and rescue teams. Most of the time, the dogs work out of sight and the handler uses approximation to determine a dog's location. In the past, once such a dog had found a missing person, it reported back to the handler. This often resulted in several re-finds by the dog before the missing person was found, often wasting valuable time, time that may have saved a life. By fitting a dog with a GPS tracking device, it allows the handler to track the dog's movements, sometimes through rugged terrain, building up a strategy for optimal search coverage given certain conditions. It also helps to locate the missing person by studying the dog's movements and areas where it may have found the missing person. In this way, assistance can be dispatched to that person at a much earlier stage. Eloise

Anderson, a search and rescue dog handler, says that "I can think of a lot of instances where this might have been a big help. This is really the future of search and rescue work. A big future lies ahead through the use of spatial information" (Michelsen Jr. 1997).

On 11 May 1996 ValueJet Flight 592 crashed into the Florida Everglades, killing 110 people on board. To assist with response efforts, the South Florida Water Management District (SFWMD) utilised GIS technology to help determine the best way to recover the bodies and cockpit of the plane. The SFWMD emergency operations centre conducted analyses and reports on the crash site to help with on-site logistics and better decision-making (Anonymous 1996(b)).

When Hurricane Bertha hit Hanover County, NC in the spring of 1995, GIS was used to trace incoming calls for help. Once a call was received, the system would automatically zoom to the area where the call originated, displaying predetermined information based on the type of emergency, allowing the dispatcher to quickly and accurately identify the caller location (Anonymous 1996(c)).

4.3. Location based emergency services

As far back as 1996, the United States Federal Communications Commission had already implemented legislation which stipulated that all cellular phone signals must be detectable up to 125 meters from their location. This is to assist emergency service dispatchers in sending an emergency service to a caller's location, in the event of the caller being unable to inform the dispatcher of the emergency address location. Often these calls are made from locations that are difficult for the caller to describe or give an address for. With the spatial ability provided through location services, the call can be traced and intercepted. Location services not only allow the user to "pull" information from the network, but also to "push" information back to the network, allowing others to benefit from it. GIS provides the geographic context required by location-based services. By incorporating GIS into everyday life, location-based

services can be seen as the foundation of a societal GIS community (Anonymous 2001(d)).

Applications related to vehicle tracking and routing allow dispatchers to identify in real time the status and location of vehicles such as ambulances, fire engines, and police vehicles. Information provided by these tracking applications is important in quick response decision-making. Such a system's capability is enhanced when location-based information is linked to geographic information by means of GIS. Location-based information, although it may be accurate and timely, can be useless to interpret, unless linked to other geographic-based information sources (Chou, Rudd & Pennington 1998).

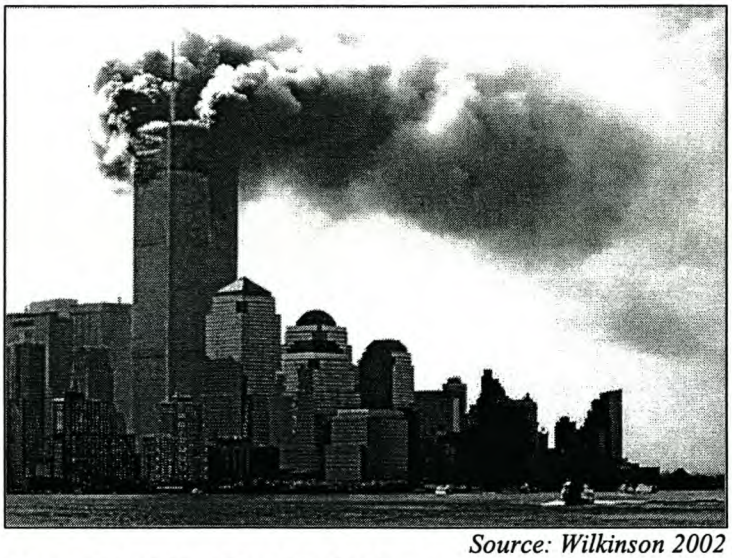
In May 2000, the US Government unscrambled the GPS signal, allowing for better positional accuracy previously only available to the US military. The selective availability (SA) deliberately introduced an error that degraded the signal accuracy of all civilian based GPS systems. By discontinuing this practice it allowed normal GPS receivers to pinpoint their positional accuracy up to 10 times more accurately than was previously possible. With SA being discarded, positional accuracy improves from 100 meters to about 10 – 20 meters, depending on the GPS receiver (Anonymous 2001(e)).

4.4. GIS to the rescue

Emergency management personnel demand rapid data processing from multiple data sources, while under the stress of bearing the legal or moral obligation for the protection or saving of a life. Advanced techniques of data searching, processing and presentation provide the potential of significantly improving the capacity of an organisation, reducing the risk associated with an emergency, and enabling rescuers to coordinate the response more efficiently (Comfort & Chang 1995).

For emergency management to be effective, it requires assimilation and dissemination of real time information and pre-planned, historical or other source data. In order to carry out the required activities and objectives, this information needs to be understood and relayed in the shortest amount of time. Communication channels between different organisations and resources need to be open at all times. What is even more difficult is that this communication and sharing of information must often occur in very hostile environments such as storms, fires or accidents. It is usually extremely time sensitive, requiring immediate attention and not leaving much room for delayed or faulty communications.

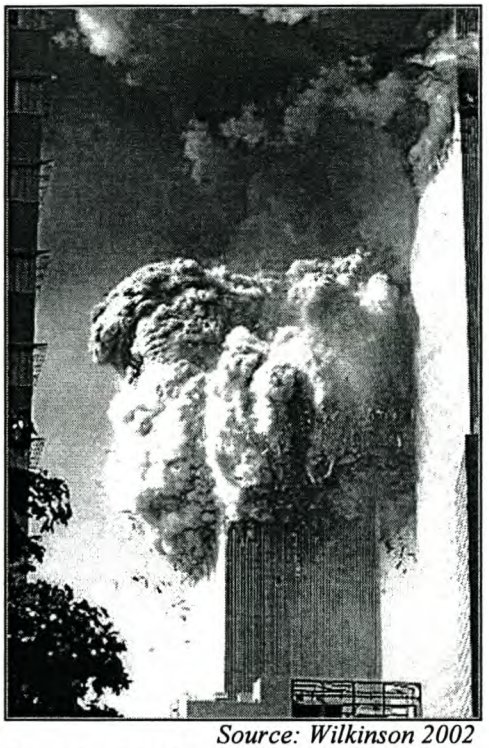
With the addition of GIS to dispatch and control systems, GIS is being used to plan, dispatch, and recover emergency situations, providing accurate information where and when needed. It gives emergency personnel the ability to safely assemble vast amounts of information and to analyse and use the information in an intelligent and efficient manner, saving time and lives. Scott says "Linked with an extensive database that provides capabilities for real-time command and control, GIS transforms disaster response into emergency management" (Scott III 1998(a):50).



Source: Wilkinson 2002

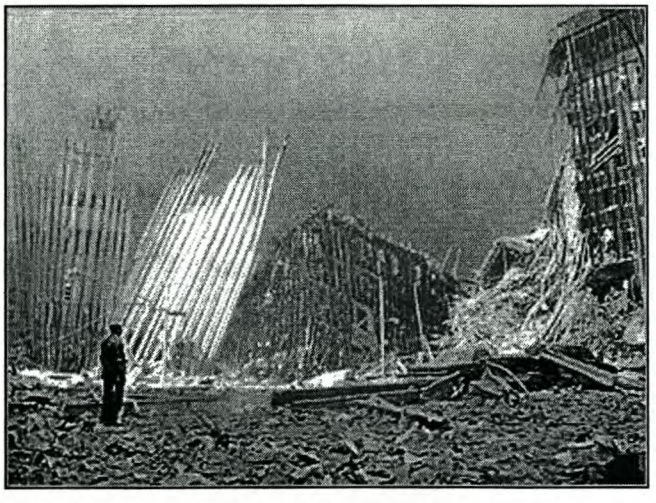
Figure 4.4: The New York skyline being changed forever

Due to the 11 September attack on the World Trade Centre in New York, the public's confidence in emergency response averting future tragedies, may change and improve when they know what role spatial technologies, particularly GIS, played in restoring order from the chaos. It allowed many emergency services, relief agencies, private companies and the media to communicate, collaborate and leverage disparate assets and resources in real time with efficiency and effectiveness as the whole world (unwittingly) watched how an important role spatial technologies played (Cahan & Ball 2002).



Source: Wilkinson 2002

Figure 4.5: The collapse of the WTC



Source: Wilkinson 2002

Figure 4.6: The remains of a famous landmark

The chain of events leading to one of the biggest disasters ever known to man, the collapse of the World Trade Centre, can be seen in Figures 4.4 – 4.6.

11 September 2002 lead to one of the biggest GIS cooperation efforts in emergency management.

CHAPTER 5 : RESCUEVIEW SYSTEM DESIGN

In order to have a better understanding of the requirements of RescueView, it is important to look at some of the fundamental issues in navigation. Thereafter system requirements and functionality will be described.

5.1 Definition of nautical components

5.1.1 Variation

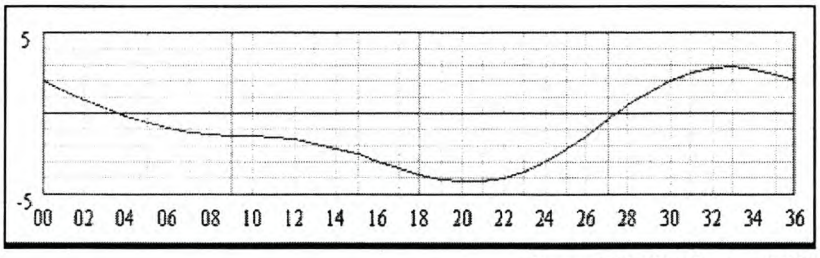
The earth's magnetic poles are not directly opposite one another. The earth's magnetic properties also do not coincide with the geographic (true) north and south poles. To complicate matters even further, the magnetic poles tend to shift their position continuously, making the magnetic meridians irregular. The magnetic displacement from true north to magnetic north is called variation (also known as declination), and is defined as easterly variation, when it is to the right of true north or westerly variation when it is to the left of true north. The amount of variation also differs from region to region depending on the latitude (Oram 1995).

5.1.2 Deviation

The compass needle of any craft is also affected by other influences causing additional errors to the magnetic heading as displayed on the compass. Several installed components on a craft, like electrical wiring, the loudspeaker magnet or any other metal object on the craft can have an effect on the deviation of a craft's compass. Even leaving an object with magnetic properties close to the compass can have a profound error on the compass bearing as displayed at that particular time. Each vessel normally has a deviation chart (See Figure 5.1), which is created when the compass has been

swung in order to determine the extent of deviation present on all compass directions (Oram 1995).

"Swinging the compass" is a process used to determine the deviation of a compass. It is established by rotating an object through each of eight cardinal and inter-cardinal compass bearings. For each of the eight points the bearing is taken and any discrepancy between this compass bearing and the known chart bearing is noted, this producing the deviation card.



Source: SailingIssues 2002

Figure 5.1: Deviation chart (example)

5.2 Data requirements

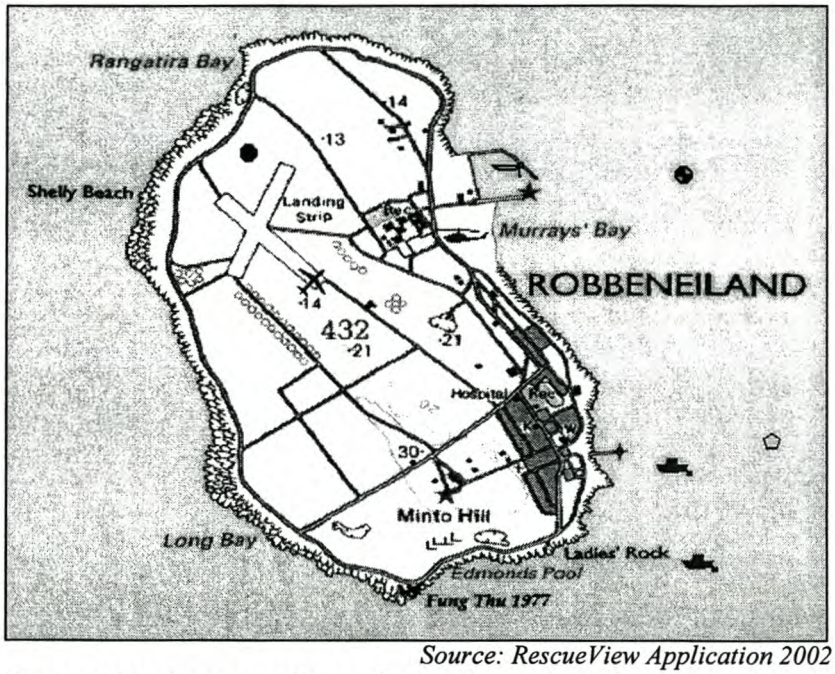
To have a successful system the following data components are necessary.

- Scanned raster maps (Topographic or marine maps)
- Shapefiles (Stores resource track log to shown route taken)
- DBF files storing event data like GPS waypoints
- INFO Database to store log file information

For situational awareness, the system utilises 1:50 000 topographic maps that were scanned, georeferenced and then used an as backdrop in the system as can be seen in Figure 5.2. This allows the rescue control centre to visually verify that information obtained via radio is indeed within the general area where the rescue craft or vehicle should be operating. This feature alone will filter out positional errors when relayed by crew, sometimes under tremendous pressure and danger. A topographical map was

chosen instead of a marine chart because it contains more information. Topographical maps are of a larger scale than marine charts (which are typically produced at a 1:250 000 scale). The application however allows the user to load any other scanned image like a marine chart.

Spatial and non-spatial information are stored in RescueView as Shapefiles (shp) or Dbase files (dbf). Information that is entered into the system is captured in a logging file system, and stored in an INFO proprietary data-base. This log file can be exported to a text file (txt) and printed as a hard copy of the events that had taken place.



Source: RescueView Application 2002

Figure 5.2: A scanned map is used as a backdrop

The system also allows the user to import and display waypoint information and utilises ArcView's built-in functionality (Event Themes) to display the information.

5.3 Application design concepts

To effectively design a database driven application, three phases need to be completed. The three phases are firstly, *conceptual design*, to ensure that user requirements are met; secondly, *logistical design*, where the design has to consider the software's capabilities and thirdly, *physical design*, to accommodate hardware limitations (disk space, memory requirement etc.).

5.3.1 Conceptual design

The usability of any GIS system depends on the application and database design and the type of data that is used. For the application to be effective, the application design process and database design need to be structured. Processed data must be useful and applicable to the task to ensure a meaningful system. In order to design a successful rescue management application, various functional tools need to be present, for instance:

- Location a particular object
- Listing attributes of specific objects
- Listing attributes of related objects
- Identifying an object that falls within the area of another object
- Finding the shortest route between two objects etc.

For the system to have the required functionality, its data content should be carefully planned and structured. This data structure is also dependent on the data entry methods, which will guide the design of the database and application and should be considered independently from software and hardware requirements.

To determine data requirements and design the application, the functional area of the application should be clearly defined. Real world scenarios should be analysed to identify and design individual database components, before they are treated as a collective entity. Constraints to the system will be identified when analysing specific tasks and listing the requirements of each task in order to obtain a functional design. Task oriented functionality, given specific design constraints and requirements, will result in a successful application.

Once the individual objects have been identified, assigning priorities to these objects based on their data requirements will ultimately guide the final design. Some objects

and associated attribute information are more important than others, and in order to be successful, objective criteria needs to be applied to determine the inclusion or exclusion of objects. Once objects have been chosen and prioritised, they should be divided into object types, defined clearly and assigned attributes. In this design process, it is important to ascertain which information is necessary, as well as what is realistically achievable.

Rules and the basic geometric elements that will represent objects also need to be defined. These rules will define the type of data to be used, in terms of raster of vector data (points, lines and polygons). Additionally, rules will also determine the object representation in terms of line types, colour or symbology types. The inter-relationships between objects will determine how tasks will be defined and assigned to a particular object data type. As an example, a physical location and its composition (e.g. a line with a start and end point) will determine the connection line between a resource's current location and its last known location, in order to draw a track (path followed) line.

Specific user-based needs will determine relationships between objects, but if no clear relationship based on topology is defined, the object descriptions can be included as attribute information. In order for objects to from part of a system, their data quality requirements such as geometrical or attribute accuracy should be analysed to determine the data resolution, timeline, and consistency with other objects and the level of information.

The best way to describe the functionality and database design of an application is to draw up an entity relationship (ER) diagram as can be seen in Figure 4.3 later in this chapter (under the heading of RescueView UML design).

5.3.2 Logical design

The logical application and database design is software independent and relies on the geographical or delineated design of the data. The size of the database will be determined by the relevant geographical areas and may consist of a single database where data is seamlessly integrated. Alternatively, the design may contain several database designs, each representing uniquely located geographical data sets. The design will be defined by the size and storage mechanism of a database or the method of updating the data.

Database design as part of a GIS application is mostly organised by thematic layers or data with similar geographical characteristics and may be stored in a similar way. Whether all linear objects will be contained in one layer or as individual separate thematic layers will be determined by the user's functional requirements. The database structure is also governed by the coordinate system used, and the tolerance (accuracy and precision) at which individual data objects were captured. The database must also have a flexible enough design so that new objects or attribute types can be added.

5.3.3 Physical design

The physical design of the application and database is dependent on the hardware limitations. The database size, storage medium, number of users and access speed will determine the design limitations.

The storage medium and accessibility of the database, either via networks, CD-ROM or mobile storage capabilities, will also have an impact on the design. The major constraint on the physical design capabilities of any system is the institutional limitations imposed by the organisation for which it is being developed (Bernhardsen 1999).

5.4 Integrated data analyses

In order for an application to perform effective and accurate data analyses, an integrated data analysis process should be followed throughout specific tasks. Some of these tasks' properties should consist of the following:

- To have a successful analytical solution, it is important to state the problem and delineate the problem solving methodology. Buffering objects, overlaying objects or determining their topological relationships could pre-determine the analytical techniques that should be used to solve a problem. Specific attention needs to be given to the data quality and its use, to determine whether the appropriate analytical technique is being used.
- It is also common practice to modify the data to comply with the analytical needs of the system. Projecting data, extracting relevant attributes or converting measurements from one unit to another (for example degrees to nautical miles), is sometimes necessary in order to meet the design criteria or input data sources.
- Performing geometric operations like overlaying or intersecting data will also create new data. Each operation performed through geometric operations will result in a new data set.
- It is also sometimes necessary to change or update attribute information as a result of a specific data analysis process. Changing attribute information based on its relationship with a specific object, geographic location or operational status is common. Attribute information should be flexible enough to accommodate the database design rules without compromising the data quality or jeopardising the data analysis process.
- Analysing attributes also uniquely defines the data characteristics and separates data record sets from other records sets based on its ability to satisfy the requirements of the search selection criteria.
- Evaluating results in terms of their relevancy, accuracy and content is important. The effort is meaningless untrustworthy information is created. It is

therefore important that the system's data and procedures are accurate and current.

- The advantage of GIS in processing large data sets and complex computations is that it allows for the early identification of undesirable results. The user is able to re-evaluate and modify the analysis process in order to achieve better results.

The results of any analytical process are best displayed on a map, accompanied by statistical information or automated reports (Bernhardsen 1999).

5.5 RescueView UML design

The UML diagram below in Figure 5.3 gives a design overview of the basic functions available in the RescueView application. The design aim was to have a system that is easy to use for a non-GIS literate person while still being able to give it advanced spatial analytical functionality. Having the user follow a few easy steps, filling in information in dialog boxes, and selecting options from drop down menus, allowed for an easy to use and intuitive system.

Having both data accuracy and database security in mind, the system has been designed to minimise user errors where possible, automatically capturing time spatial information in a database log file, allowing for post-analyses of whole operations. This feature does not only allow the generated data to be reviewed by emergency managers during a debriefing, but also allows the information to be used during expert witness testimony, if needed in a court of law.

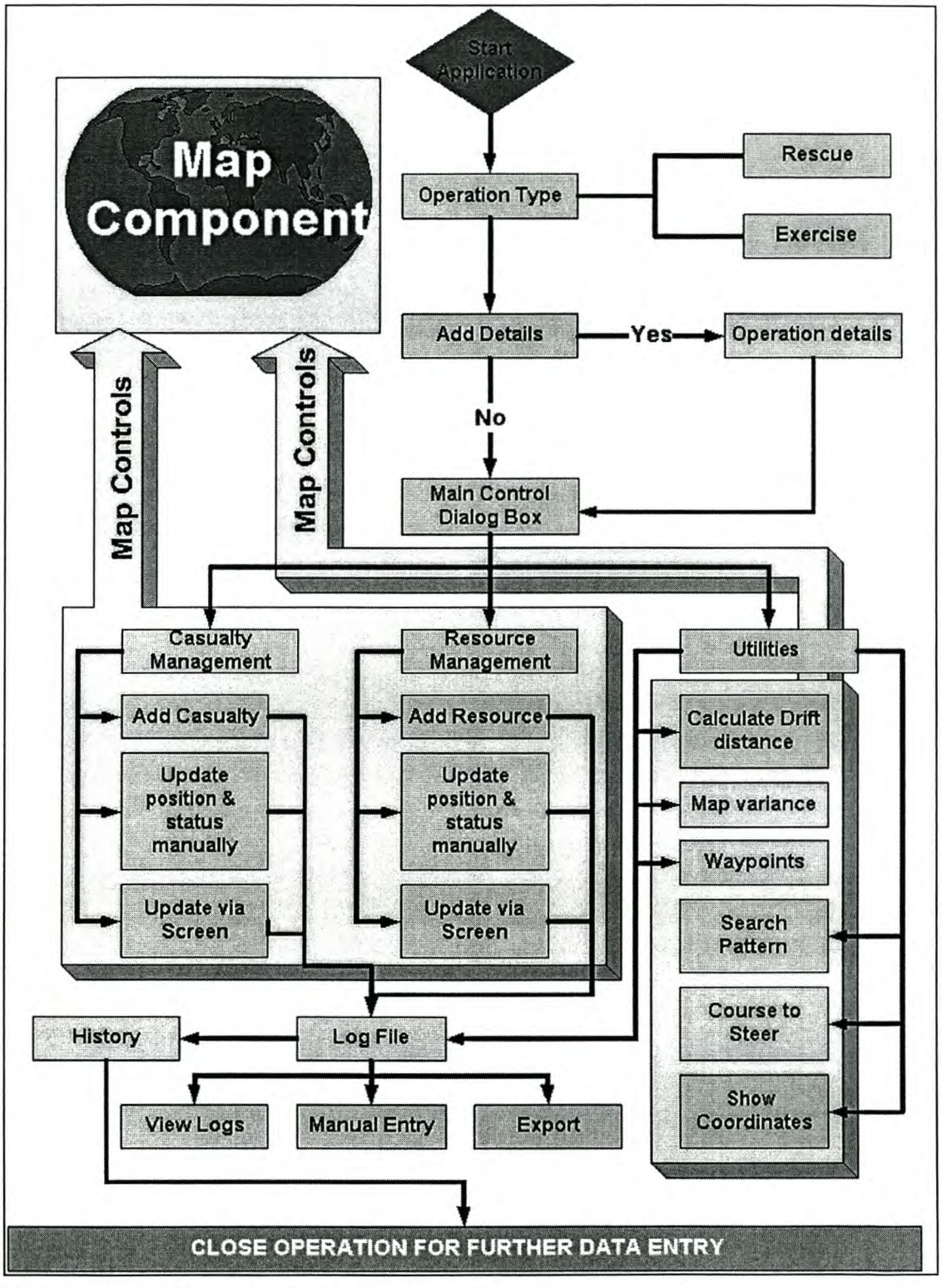


Figure 5.3: RescueView UML Design diagram

5.6 RescueView system design

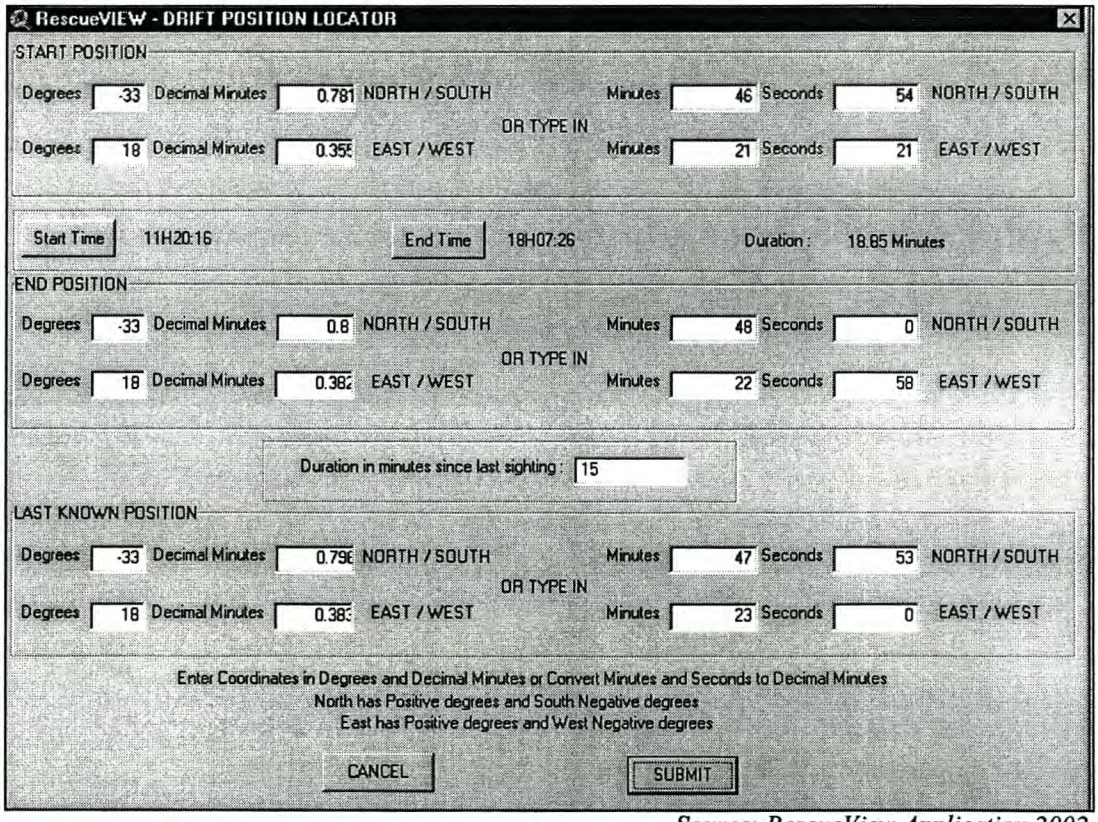
System design can be described as knowing how a system will be used and forms the basis of how information is stored, and how it is used in terms of its technical, organisational or legal requirements. The design is thus a method of determining what changes an organisation has to make so that an information system will serve its purpose. In designing such a system, attention needs to be given to issues like procedural changes, standardisation of data, training, functionality and organisational processes (Huxhold & Levinsohn 1995).

The RescueView system was designed with simplicity in mind. The strategy was to create a system that needs very little user training, has few buttons, is quick to navigate through, and most important of all, does not require GIS literacy. All "ArcView" menu commands and graphical user interface (GUI) buttons not used in RescueView were removed. By providing a master control, all functions were grouped together in one place, with easily readable GUI buttons.

A further important requirement was that the system needed to log the entire operation, whether it is a rescue or a training scenario. For the system to go into logging mode, the user has to start a new operation, choose the type of operation and enter a brief description of the operation to be undertaken. For this purpose, a time-stamped logging system was created which captured every change to the coordinates of the rescue craftor the victim. This function was created with several purposes in mind.

- Firstly, it can be used as a reference system to "go back" in time, allowing the user to view the chain of events that led up to that point.
- Secondly, a hard copy time-based events log can be produced and
- Thirdly, the system can be used to walk through the entire operation for crew debriefing or crew training.

Several system additions and enhancements were made to include functions like drift capture (See Figure 5.4), course to steer, on-screen clicking to obtain positional information, rescue craft or casualty status, search pattern generation, compass rose generation to triangulate a location and trace history of rescue craft. The system also allows the user to save the entire operation as a project. Once an operation has been completed, the user closes the event and the log file gets time stamped. This prevents any further information being logged. This feature allows the system-stored information to be used in the event of a legal inquiry or as evidence in a court of law.



Source: RescueView Application 2002

Figure 5.4: The drift locater dialog window

5.7 Operational functions

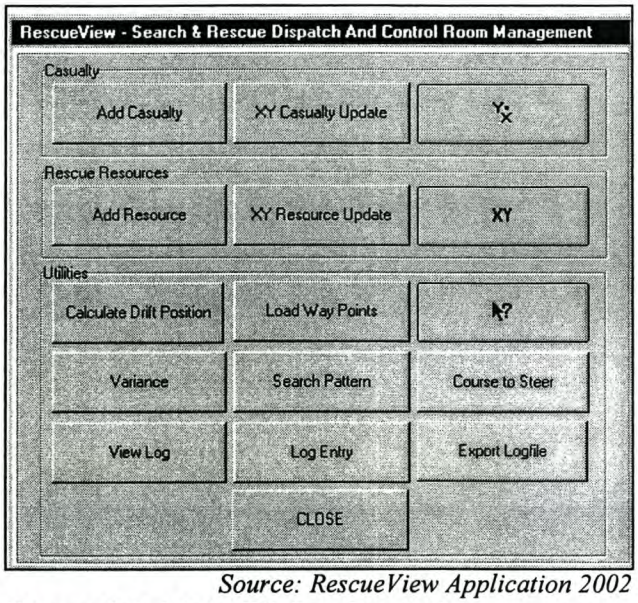
The operational flow of the system has been set up in such a way that the user will start at one point and manipulate all of the system functions in the same way.

5.7.1 Capture operation type and description

The user will create an operation by starting the system, whereafter the system will prompt for the type of operation (exercise or rescue). The next step will be to add descriptive information regarding the details surrounding the operation to the system. This functionality will be described in more detail in the logging and report back section that follows.

5.7.2 Main control windows

Once the operation has been registered with its descriptive information, the system will enable the main control dialog window (See Figure 5.5). From this dialog, the user can enter the location of the victim or vessel and update its location or status through several user-friendly buttons. The addition of resources is dealt with in the same manner, shortening the learning curve of the system for a new user. The user also has the option to either update information, based on relayed coordinates that have been called in via radio, into a coordinate dialog window or to manually pick the coordinates from the screen. Picking coordinates from the screen is particularly handy when information obtained via radio refers to a known location. With the user having the background information available via a scanned map, he or she can easily find the location and update the system with the site's coordinate information.



Source: RescueView Application 2002

Figure 5.5: Main Dialog Control

5.7.3 Calculate drift direction and distance

The most important tool in the RescueView system is the one used to calculate drift direction and distance. To calculate the drift speed and -distance the following information is needed:

- starting coordinate;
- end coordinate; and
- duration.

Using this information the distance can be calculated by creating a line programmatically. The functionality in ArcViews' Avenue scripting language allows for the creation of a line with a start and end point. Creating two points, and then creating a line from the two points achieves this, as can be seen in Figure 5.6. The resulting shape length is calculated in order to return the shape length, thus returning the total distance the boat or diver has drifted from drop off till pick up time.

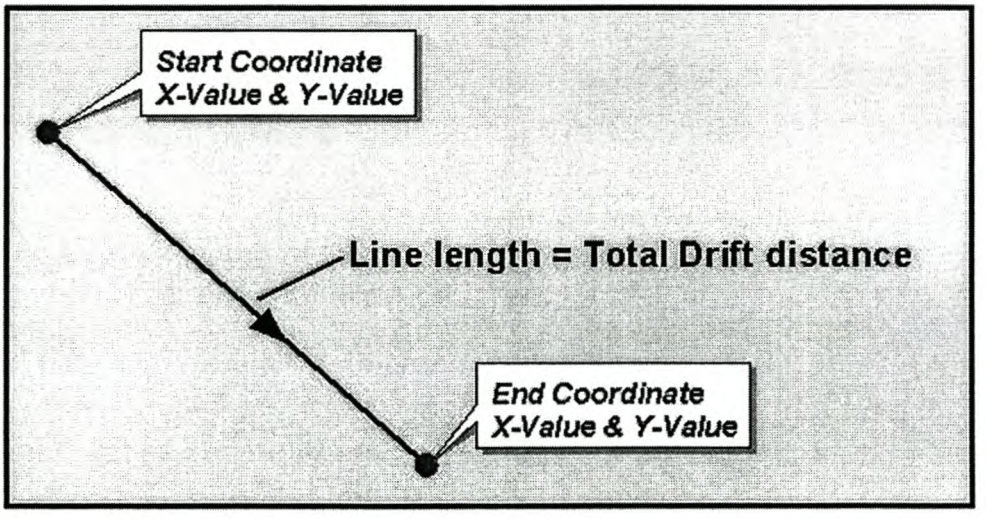


Figure 5.6: Calculating drift distance

The total duration is obtained by capturing the start drift time of an object (diver or boat) and then the pick-up time as illustrated in Figure 5.7. Once the total distance and time is calculated it is possible to determine the drift speed and direction.

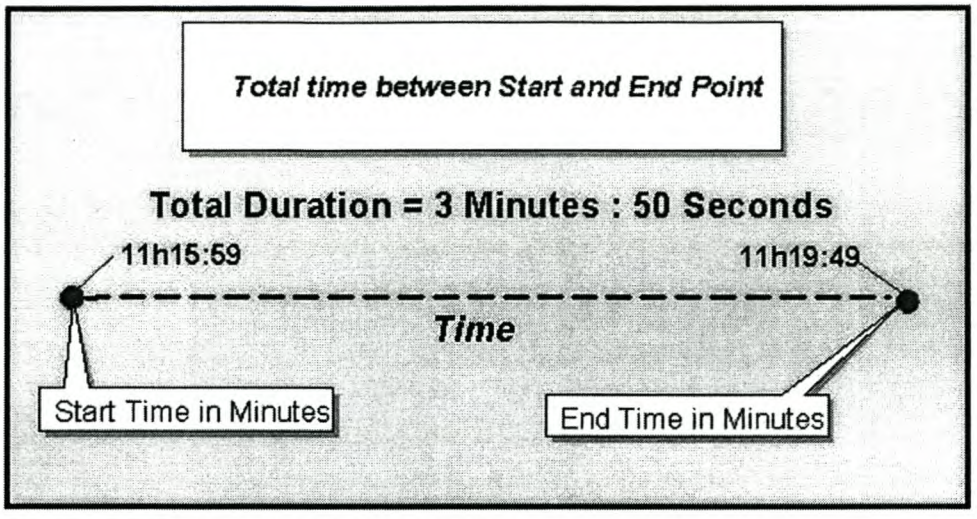


Figure 5.7: Calculating drift duration

Calculating the drift bearing is accomplished when an Avenue script calls the "*A_RESCUEVIEW_DRIFT_BEARING*" functions while passing it the Start and End point coordinates in decimal degrees. An extract of the "*A_RESCUEVIEW_DRIFT_BEARING*" script below, adapted from the script by Kimball (2000), illustrates in more detail.

```
'------------------------------------------------------------------------
'calculate the angle using  trigonometry :
'------------------------------------------------------------------------
theDistance = thePoint1.Distance(thePoint2)          'theDistance  = the distance
between the two points (hypotenuse)
dX = theX2 - theX1                                   'dX = difference in xcoords
dY = theY2 - TheY1                                   'dY = difference in ycoords

'------------------------------------------------------------------------
'theAngle   = calculate angle between two points (Clockwise from North in
Degrees):
'------------------------------------------------------------------------

theAngle = 90 - ((dX / theDistance).ACos.AsDegrees)
if (dX < 0) then

  if (dY < 0) then

'------------------------------------------------------------------------
'use negate avenue function to turn negative value into positive value
'------------------------------------------------------------------------

    theAngle = 180 + theAngle.Negate
  else
    theAngle = 360 + theAngle
  end
else
  if (dY < 0) then
```

```
    theAngle = 180 - theAngle
  end
end
'-------------------------------------------------------------------------
'return the angle as a number between 0 and 360:
'-------------------------------------------------------------------------
return theAngle        ' return value to program that requested function
```

Once the bearing angle has been calculated, it is used by the calling scripts to extrapolate a new coordinate where the casualty may be found. Using the angle and drift distance the new coordinate is calculated using the mathematical expression below.

An extract from the "*A_RESCUEVIEW_DRIFT_DISTANCE*" script demonstrates the use of the *Units.Convert* and the *AsRadians* functions available in Avenue to calculate the new X- and Y-coordinate. The *Units.Convert* function is used to convert the nautical distance into decimal degree distance.

```
'-------------------------------------------------------------------------
'*** CONVERT DISTANCE FROM NAUTICAL MILES TO DECIMAL DEGREES
'-------------------------------------------------------------------------

theProjectDistance = Units.Convert (theLostDistance,
#UNITS_LINEAR_NAUTICALMILES, #UNITS_LINEAR_DEGREES)

theTanValue = 1
thePie = 4 * theTanValue.aTan


newx = ( ( ( ( thePie/2 ) - theBearing.AsRadians ).Cos ) * theProjectDistance ) +
theLastXValue
newy = ( ( ( ( thePie/2 ) - theBearing.AsRadians ).Sin ) * theProjectDistance ) +
theLastYValue
```

The above function is applied to the casualty's last known location and from that point a new possible location and total drift distance is estimated.

5.8 Calculate course to steer

This function allows the user to select the current position of a rescue craft and then to select the position of a target. The system will determine the magnetic course to steer, as well as the total distance in nautical miles. This function takes the local magnetic variation into consideration, when calculating the course to steer from true north to magnetic north. The formula used to calculate magnetic north is shown in Figure 5.8 .

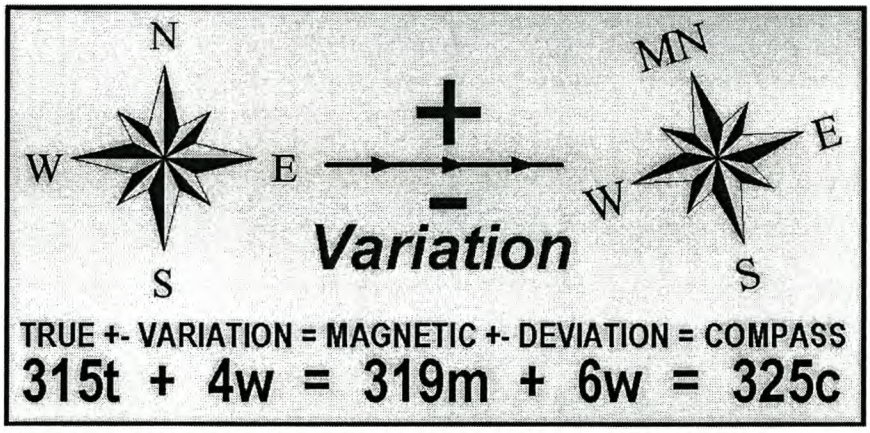


Figure 5.8: Calculating Magnetic North using Variation

To calculate the Magnetic North the formula, T ± V = M is applied.

True North heading (T) ± Variation (V) = Magnetic heading

Deviation was not taken into account, as all rescue vessels are equipped with GPS as their primary navigation tool. If a vessel has a compass deviation card, they can apply the deviation to the magnetic heading to get a compass heading.

To calculate Compass heading, the formula, M ± D = C is applied.

Magnetic heading (M) ± Compass Deviation (D) = Compass heading (C)

5.9 GIS spatial time management

Historically speaking, GIS has not been designed or developed to address time management, since initial interest was primarily focussed on only current GIS data and operations. Although GIS databases were updated and modified over time, the spatial

dynamical changes were not recorded. Handling temporal spatial changes in GIS has received much needed attention in recent years. GIS systems will have to accommodate these theoretical and practical changes to provide better reality-based information.

To fulfil its role as a decision-making tool, GIS has to represent both space and time phenomena to reflect human actions in geographical space as closely as possible. The purpose of temporal data is to record and reflect changes over time. These changes can either be reflected as a continuous collection of events or consist of a particular time slice representing a particular event. Space–time modelling can be defined as the modelling of the change of state and/or location of an entity and can therefore be classified as:

- *continuous* - being repeated at a set time interval;
- *majorative* - being present most of the time;
- *sporadic* - occurring at some or other point; or
- *unique* - occurring as a single event and never again.

The duration and frequency of changes are important parts of temporal spatial changes and their pattern can be chaotic, steady or sporadic. Individual events mostly create episodes that can form part of a specific cycle, while a location change can be either gradual or sudden. Entities can be created, changed and terminated over time, while their identities are maintained. For spatial analytical purposes, these events can expediently be grouped into discrete time units, although time and space are considered to be continuous in nature.

Location-based spatial data can only be viewed as temporal snapshots as represented by individual spatial layers. The spatial layers do not store data as a time-continuous data set, but rather as a representation of the data at a specific time. The period between data changes is not necessarily uniform, and the time difference can vary, based on whether the entity needs to be updated or not.

To effectively use spatial-temporal data, a user should be able to analyse changes on the basis of time. This would enable a user to retrieve spatial-temporal locations based on an event, and thereby identify possible spatial patterns that could be grouped together. The most important aspect of spatial-temporal data, however, is that it presents the opportunity to analyse the chronological order of the data in space (Peuquet 1999).

Versioning can be used to manage spatial-temporal data using the most commonly available relational database management systems (RDBMS). Versioning is accomplished by taking a snapshot of the complete database before any changes are made to the database. Every change to that specific record entity is recorded in the database, and is stored either as a new record, a change to the current record or a deletion of an existing record. Depending on the database design, these changes can be reflected either as changes at specific time intervals (hourly, daily, weekly, monthly, yearly etc.) or as each individual record changes, at no set time interval.

In order for spatial-temporal data to be used in RescueView, the design had to adhere to the organisational, hardware and software limitations imposed on it. The tool had to be inexpensive, lightweight and mobile, and therefore an RDBMS database that supports MS SQL Server or Oracle versioning could not be included. RescueView's design is thus limited to the use of ESRI's shapefile technology utilising DBF files, as well as the supported INFO RDBMS. Spatial-temporal data cannot be stored in the shapefile format as it is an event, and therefore time specific. The system stores the last known spatial location, status, time and attribute information in the shapefile, but all historical data is added to the INFO RDBMS system, where it is kept as a time line of past events. This chronology of events can then be used to replay spatial-temporal specific events.

As each event takes place, such as updating an object's location, changing its status, or adding additional information, the logging system writes these new event changes, as

well as the old data values, to the INFO database before overwriting the shapefile records with the new information.

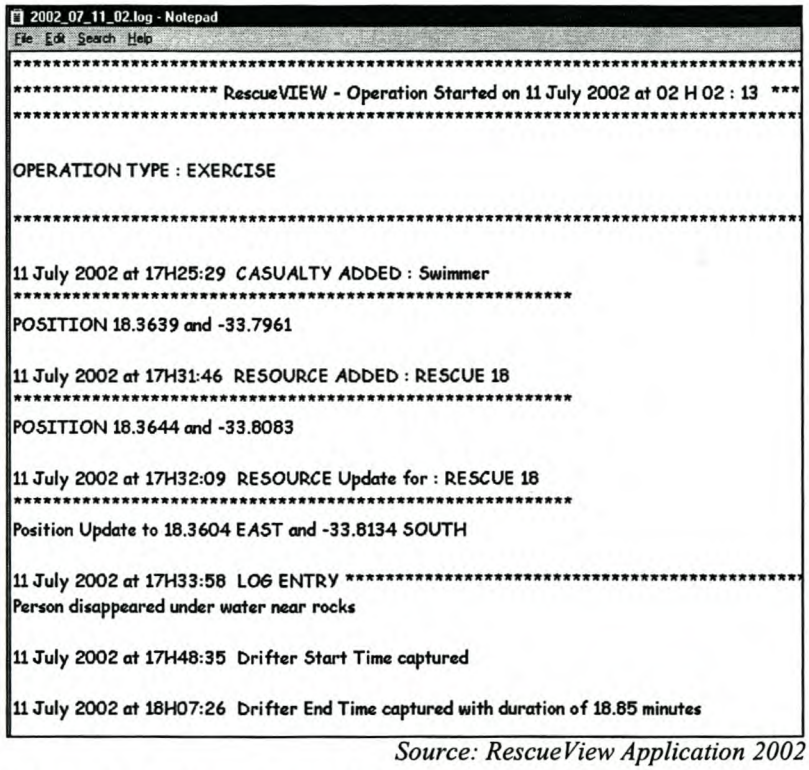
5.10 Logging and report back

The purpose of the log file system is to capture all events that have taken place, and write them to a log database. Every entry has a time and date stamp, allowing the user to view the history of the operation, as well as have a reference of events. Originally the log file was only written to a text file, but this had several limitations.

The log file's current format allows the events to be written to a database. This has several advantages. The first is in the security and credibility of the system. When needed, the system can be used in a court of law to show the chain of events, as the user cannot enter the database during the operation or after it has been shut down. The user is able to browse the existing log files. The second advantage is that information can be categorised by event type. This allows for quick referencing of data to view events like resource updates, user comments etc. Having information in categories makes it easier for the user to skip over unnecessary information. The third advantage is that the data can be sorted by time, allowing for the user to perform boolean operations on the data, to find specific events.

By changing from a text-based logging system to a database, the system can have a playback function, which will be implemented in a future release. Even though the system has been changed to a database driven log file, the user still has the option of exporting the database to a text file, as seen in Figure 5.9.

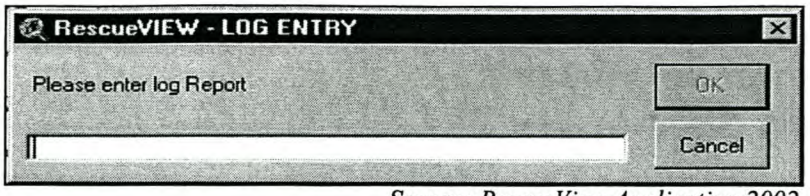
The events captured by the system are as follows. When the user creates a new operation, the start date and time are captured. The user then selects the type of operation, either rescue or training operation, which also gets captured. The operational details then have to be entered into a description box, which also gets time- and date-stamped. These details typically will consist of all the information that has been captured up to that point, e.g. why the rescue team was called out, by whom, caller information or contact details, location information, description of the victim or vessel and any other information deemed important.



Source: RescueView Application 2002

Figure 5.9: Time stamped log file created by application

Each time that a victim or resource is added to the system, or when positional information is updated for either he victim or resource, it will be captured into the log file system. The system also logs other information when the tool for calculating drift speed and direction or any of the other tools that gives positional or course-to-steer information is used. The last feature that was added was to allow the user to enter miscellaneous or descriptive information, as was necessary or deemed important, e.g. updated information from the scene, or a new caller or conditional information as reported by the crew (See Figure 5.10).



Source: RescueView Application 2002

Figure 5.10: Dialog box to add comments/details to log file

Another logging feature that the RescueView application has is to visually trace the progress of the resources on screen. This is achieved by creating a polyline shapefile and storing a polyline shape for each resource that is added to the operation. Each time the position of a resource is updated, a polyline segment is added to this existing polyline giving an accurate view of the chain of events from the start of the addition of that resource to the operation, to the last known position update in the system.

The Avenue source code below, extracted from the RescueView application, illustrates the tracing functionality of resources.

```
theFtab = theTheme.GetFtab
  theShapefield = theFtab.FindField ("Shape")
  theIDfield = theFtab.FindField ("ID")

'-------------------------------------------------------------------------
' Find unique resource record
'-------------------------------------------------------------------------

  theBitmap = theFtab.getSelection
  theQuery = "[ID] = " + theID
  theFTab.Query(theQuery, theBitmap, #VTAB_SELTYPE_NEW)
  theFTab.UpdateSelection

  theFtab.SetEditable(TRUE)

'-------------------------------------------------------------------------
'If no record found, make a new polyline shape record
'-------------------------------------------------------------------------

  if (theBitmap.count = 0) then

    theLinelist = {theXvalue1@theYValue1, theXvalue2@theYValue2}
    thePolyLine = PolyLine.Make ({theLineList})

    rec = theFtab.Addrecord
    theFtab.SetValue(theShapeField, rec, thePolyLine)
    theFtab.SetValue(theIDField, rec, theID)
```

```
  Else

'-------------------------------------------------------------------------
'Else append to existing polyline shape record
'-------------------------------------------------------------------------

    For each rec in theBitmap

      theShape = theFtab.ReturnValue(theShapeField, rec)
      thePolyLine = PolyLine.Make ({{theXvalue1@theYValue1,
theXvalue2@theYValue2}})
      theNewPolyLine = thePolyLine.ReturnMerged (theShape)

      theFtab.SetValue(theShapeField, rec, theNewPolyLine)
    End

  End

  theFtab.SetEditable(FALSE)
  theBitmap.clearAll
```

The tracing of a resource coupled to the logfile system allows for complete post analysis of the operation. This design enables future enhancements to the logging system, such as the addition of a step-by-step playback function. Due to financial constraints and computer technology limitations, it was decided not to implement real-time GPS tracking of resources in time and space. Existing GPS technology already allows for real-time error correction, utilising differential GPS (DGPS) base station technology, but due to the added cost of the DGPS receivers and GPS base station equipment, it will not be implemented at this stage. One such existing addition is to use the ArcView Tracking Analyst extension. The functionality available in the current version of RescueView 2.6 includes all functionality necessary to have a real impact on the way that GIS can come to the rescue.

CHAPTER 6 : RESCUEVIEW CASE STUDY - AIRCRAFT CRASH SIMULATION

The functionality of RescueView is largely based on the need to have the right tools available during a rescue operation. The RescueView application was initially designed as a resource-tracking tool, in order to give station commanders and control room operators an idea of where rescue vessels are located in relation to the last known position of a casualty. During a sea rescue exercise, the software was put to the test, but it quickly became apparent that additional functionality can aid rescuers in the decision making process. This chapter serves to discuss the lessons learned and the functionality that was born out of experience.

6.1 Disaster scenario

"A simulated disaster involving a 'passenger jet crashed' in Table Bay, turned into a dramatic real-life drama on Saturday when three of the supposed 'survivors', including two teenagers, where lost at sea under a blanket of fog" as reported by the Cape Argus (Murray 2002).

The scenario involved the downing of a small passenger jet in Table Bay, Cape Town. Included in the simulation forty-seven passengers were scattered throughout the cold Atlantic Ocean. These 'survivors' were taken by navy boats several miles offshore and dropped into the ocean while being watched by a second boat. The second boat had to watch the survivors until they were spotted by the rescue boats. All went well with the first three groups. As the fourth group was taking to the water, the air was clear and it appeared as if the light mist of the morning was clearing around them.

Unfortunately, the mist moved in again, thicker than before and engulfed the "survivors". The rescue coordination centre soon realised that something had gone

wrong, after none of the rescue boats reported to have found the fourth group. The exercise was aborted and transformed into a fully-fledged search and rescue with five NSRI boats, three navy boats and a police boat all searching for the missing persons. An hour and a half later a navy boat found the missing persons in the water, after hearing the victims blowing on their lifejacket whistles (Murray 2002).

6.2 Role players

The full South African Search and Rescue (SASAR) complement participated in the exercise. This included the NSRI, the navy, the police water wing, the air force, port authorities, METRO Rescue, Cape Town's Maritime Rescue Co-ordination Centre located at Silvermine and several disaster management units of the City of Cape Town (Williams 2002).

Rescue stations from Saldanha (Station 4), Melkbosstrand (Station 18), Table Bay (Station 3), Bakoven (Station 2) and Hout Bay (Station 8) also took part. Each provided search and rescue vessels in conjunction with the navy and police. Several personnel from METRO Rescue, the province's ambulance services, also participated in the exercise. Each survivor was given a few "injuries" which they had to convey to the medical personnel, who treated them accordingly before being taken to "hospital".

6.3 Operational timeline and debriefing

The exercise started at 07h09 on 2 March 2002 and ended at 11h30 the same morning. The RescueView application was used from the start as a tracking system to show the media and other participants the exercise's location and progress. Regular updates, including boat and casualty coordinates and casualty conditions, were fed to the system, with the results being projected onto a large screen. The system also kept a log file to show the time between events, as they happened.

The exercise was coordinated from the Port Captain's office. From here all casualty drop off times were controlled by means of a preconceived plan. Each group of casualties had a call-out time, a casualty drop-off time, resources allocated to those casualties, the position where they had to be dropped off, as well as the injuries. An example of the drop-off time control list can be seen in Table 6.1 below.

Table 6.1: Casualty drop-off times control list

CALL-OUT TIME	CASUALTY DROP-OFF	RESOURCE	POSITION	CASUALTIES
06h45	06h45	R18	33°52.6'S 38°27.8'S	#1 Lacerations - head #2 Open Fib/Tib fracture
07h10	07h10	R2	33°52.5'S 38°27.1'S	#3 Broken Femur #4 Lacerations - head
07h30	07h30	R3A	33°52'S 38°27.4'S	#5 Dislocated Shoulder #48 Fuel inhalation

Source: Barns 2002

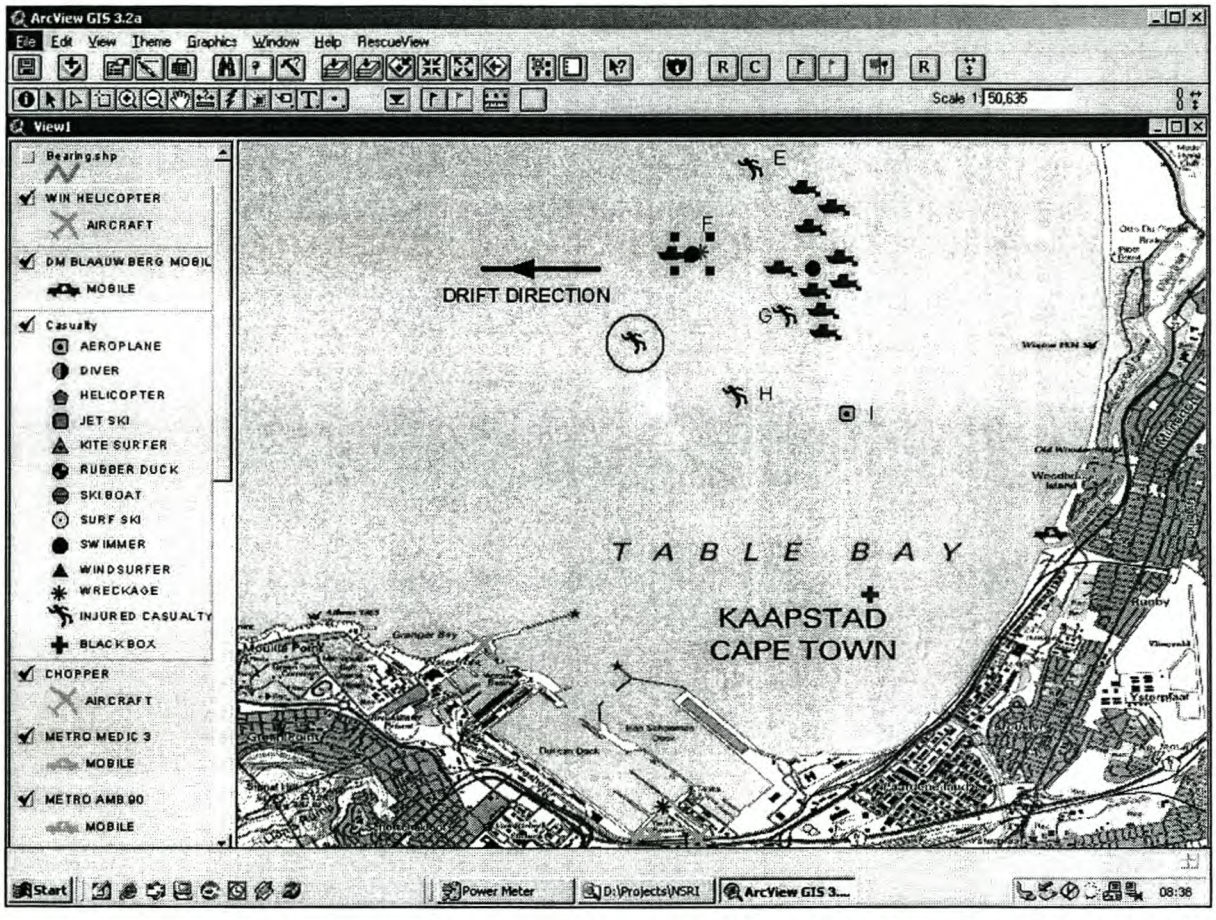
After the rescue effort was completed and all missing persons were recovered, a crew and media briefing was held to demonstrate the RescueView application. The program succeeded in illustrating what transpired during the course of the day. It was also shown where the three missing people were found in relation to the rescue boats and the search area.

6.4 System success and value

The first value of RescueView was to give command centre personnel insight into what was happening out of their sight, on the water. This was a great improvement from the previous system where they had to rely solely on two-way radio communication. The location of each rescue boat was constantly being updated, which ensured that all resources could be correctly allocated and therefore fully utilised.

The system proved its worth when the "survivors" went missing (See Figure 6.1). The system was used to identify the last known location of the "survivors" in relation to the

updated position of the rescue craft. The log file was also screened to obtain the exact time the "survivors" were placed in the water in order to determine the duration they had been exposed to the cold. Both the NSRI Saldanha station and the Blaauwberg Surf Lifesaving commanders were then consulted to determine the endurance and profile of the people who went missing.



Source: RescueView Application 2002

Figure 6.1: Screen capture shot at the time of search for missing "Survivors"

Drift directions were established by placing a diver in the water and recording the direction in which he drifted. Rescue boats were thereafter moved accordingly. This information was also projected on screen, keeping the crews who had returned from their "rescue" informed of the progress in the search for their missing friends.

The system was also successfully used to do a debriefing for the crew and the media. It generated a positive response and a call for the system to be implemented at all rescue

stations. Several functional additions were also proposed. ESRI was also inspired by the use of the system to donate software to be used by all stations in saving lives.

6.5 System failures

The exercise was very valuable in identifying flaws and failures in the system, and it allowed some of the role players to get an idea of the capabilities of the system and where it could be used.

One of the first additional requirements was to see the path that was covered by rescue craft as they swept an area looking for "casualties". This function would have allowed them to see which areas had been covered and by whom.

The second shortcoming of the system was that RescueView only allowed coordinate entry in degrees, minutes and seconds. The fleet make-up consisted of a variety of boats, so that GPS waypoints were returned in either decimal degrees or in degrees minutes and seconds. This led to manual conversion of all non-decimal degree values, often leading to errors. Fortunately these errors were quickly discovered since the visual confirmation on the map showed the craft in the wrong area or very far off course.

The third major failure of the system was identified when drift direction, speed and the possible location of "casualties" had to be calculated. This functionality was needed to determine the possible location of the three missing crew, by calculating a new coordinate based on drift direction and speed. Being able to calculate the possible drift direction and speed could have resulted in finding the missing crew earlier. As this was not available, the exercise changed into a real life rescue drama.

Two other flaws were also apparent. The inability of the system to allow the controller to advise a rescue craft on which course to steer, and the lack of search pattern display techniques also limited its usefulness.

All these system shortcomings have been corrected and the additional functionality was added to the system. The system is being continuously updated as new functionality is added from ideas generated through its continued field use.

CHAPTER 7 : CONCLUSION AND RECOMMENDATIONS

7.1 System aims: Were they met?

The aim of RescueView is to save lives. The fact that it was used on its very first trial run in an actual rescue scenario and that it provided valuable information, however small, proved its value and success. A system of this nature is never really complete, as there may always be ways of improving techniques, technology and training.

The objective - to help rescue crews to locate a missing person or vessel - has been met. A further requirement - to give onshore controllers and commanders a detailed perspective of what was transpiring on the water - has also been met. Making it possible to determine the local drift direction and speed has also been introduced with success. A valuable lesson was learnt by the absence of the capability to compute drift direction and speed during the first rescue scenario, but its subsequent inclusion was of paramount importance to the future success of the system.

The requirement for the ability to generate a complete log file with detailed information of what transpired during the rescue has also been met. Controlling resources, knowing what is taking place and what might occur will greatly lighten the difficult task of the men and woman of the NSRI. This system will contribute to the NSRI's continued success.

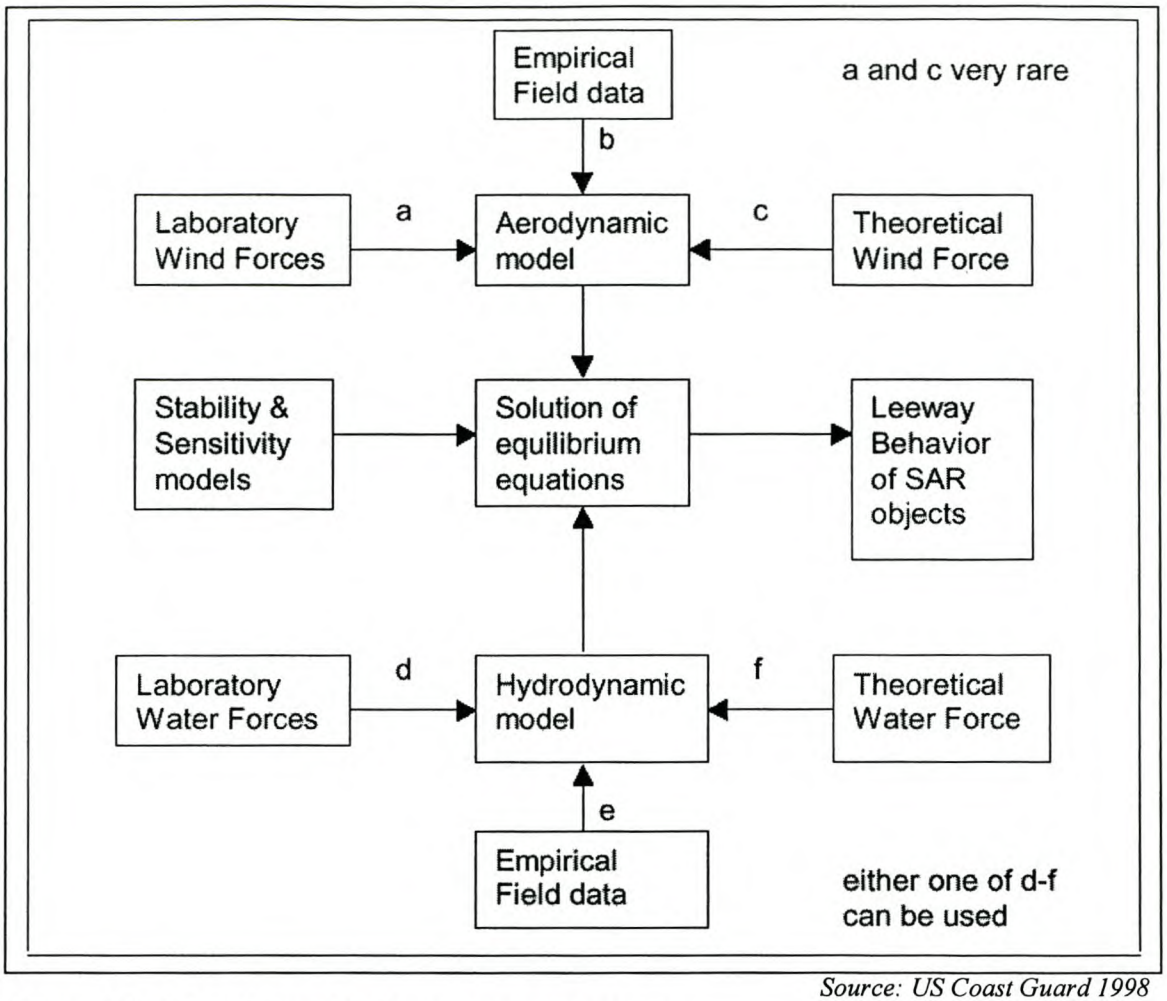
7.2 Future development recommendations

The use of drift speeds, drift angles and estimated positions in search and rescue, whether that of a boat, a person or any submerged or floating object, can be made very complex if all the drift variables have to be considered. The scope of this thesis and the current technology available to most NSRI stations do not allow for the use of

these elaborate techniques. Although some techniques could greatly enhance the final results, it may, for the time being, be impossible to implement additional technology. A study by the US Coast Guard (to be discussed later in this chapter) has shown that too many variables exist and even under laboratory conditions, most of these conditions are difficult to measure. Some of these variables include the current angle of the vessel against the prevailing wind, the constant wind velocity and direction changes while measurements are being taken, the shape of the vessel, the depth of the vessel in the water, and the size and shape of the body in the water. It is clear that these variables are impossible to trace on the type and size vessel currently operated by the NSRI.

This, however, should not detract from the fact that it can be important for future study and possible implementation to include some of the techniques and proven measurements used by other agencies in search and rescue models.

The United States Coast Guard has made an extensive study in this regard, and from their study it is clear that a multitude of factors can play a role, but that most of them may only be possible to determine under scientific laboratory conditions. The angle at which an object lies in the water in relation to the prevailing wind and sea current plays an important role. Since this angle changes continuously, an object's drift speed may increase or decrease constantly while adrift. In Figure 7.1 below, some of the different inputs required to determine Leeway drift prediction can be seen (US Coast Guard 1998). This is however impossible to determine and use outside laboratory conditions.



Source: US Coast Guard 1998

Figure 7.1: Velocity prediction program structure

Based on the research conducted by the US Coast Guard, it is impossible to accurately predict where exactly a victim will be found without using some very complex models and measuring equipment. Too many factors are involved in determining the drift of the object in the water. Variables include the sea current speed and direction, wind speed and direction, and the shape and height of the object above and below the water.

The use of the software can be enhanced in several ways. When talking to a stranded person, rescue crews can get an idea of the search area location by adding the coverage area of cellular transmission towers to the system. The person calling for help on a stranded vessel can relay the name of the current cellular tower that he or she is using. (This is the name that appears on a cellular phone's screen and that changes as

reception is switched from one tower to the next). Overlaying different cellular towers' reception areas would refine the search area even more.

Another enhancement to the system could be the addition of a playback function allowing the crew to be debriefed by showing them the course of action during the operation. Having the system show the events in a step-by-step method could even highlight areas that may have been overlooked or may show where strategy improvements can be made.

The effective use of RescueView, currently only available to land based shore controllers, can be enhanced by making it accessible to boat crews as well. Porting RescueView to mobile handheld devices coupled to a Global Positioning System and automated position updates to the base station will make the integration of the system even more successful.

The future of GIS in a rescue environment will be determined by technological changes. Those changes are happening on two fronts - advances in mobile handheld technology, as well as mobile network technology. Mobile handheld technology is being enhanced with built-in GPS technology, broadcasting the caller's current position. On the network side, major improvements are being made in network triangulation between cellular sites (Zecher 2001).

7.3 Conclusion

The potential of saving a life by using this system is what inspired it. Technology advances on a daily basis and may be obsolete before there has been a chance to use it. In terms of the NSRI, the use of RescueView, however, may change that, allowing this technology to be used as standard equipment.

Creating RescueView will hopefully also inspire others to utilise their knowledge and talents to provide organisations like the NSRI with sophisticated and robust tools to

help them save lives. Every life is precious and if RescueView can save one more, it was successful.

"As for courage and will – we cannot measure how much of each lies within us, we can only trust there will be sufficient to carry us through trials which may lie ahead" (Famous Quotations 2002).

With these words RescueView will remain committed to saving lives!

REFERENCES

1. Ajam, K 2002. NSRI still searching for missing swimmer. *Cape Times*, 19 May.
2. Amdahl, G 2001. *Disaster Response – GIS for Public Safety*. Redlands, California: ESRI Press.
3. Anderson, DR 1990. The Application of Geographic Information Systems in the Spatial Analyses of Crime. In Scholten, HJ & Stillwell, JCH (eds.) *Geographical Information Systems for Urban and Regional Planning*, pp143-145. Dordrecht: Kluwer Academic Publishers.
4. Anonymous 2001(a). Fire Agencies Improve Response with GIS. *ArcUser*, 3.1 : 22-25.
5. Anonymous 1996(b). Newslink – GIS Supports Valuejet Plane Crash Investigation. *GIS World*, 10.3 : 13.
6. Anonymous 1996(c). Newslink – GIS Aids Emergency Response Efforts. *GIS World*, 9.10 : 15.
7. Anonymous 2000(d). Intersecting Technologies Create Location Services. *ArcUser*, 4.2 : 10–12.
8. Anonymous 2001(e). U.S Government Unscrambles GPS Signal. *GeoWorld*, 13.6 : 10.
9. Anonymous 2001(f). Tourist rescued after jetski drama. *Cape Argus*, 29 December.
10. Anonymous 1998(g). Three lost in sea tragedy. *Sunday Times*, 18 October.
11. Aerospace and Telecommunications Engineering Support Squadron - Canadian Forces Base Trenton, Ontario 1997. Search & Rescue Planning: A GIS Solution. Paper (delivered) at the 17th Annual ESRI User Conference, San Diego.
12. Barns, R 2002. Spreadsheet document used by R, Barnes during SASAR exercise on 2nd March 2002 in Table Bay to place "survivors" in water.
13. Bernhardsen, T 1999. *Geographic Information Systems – An Introduction*. New York: John Wiley & Sons, Inc.
14. Cahan, B & Ball, M 2002. GIS at Ground Zero. *GeoWorld*, 15.1 : 26–29.

15. Campbell, G 1992. GIS in the police environment in Cadoux-Hudson, J & Heywood, I (eds.) *Geographic Information 1992/3 - The Yearbook of the Association for Geographic Information*, pp. 115. London: Taylor & Francis.

16. Chou, Y, Rudd, EI, & Pennington, J 1998. Emergency 911: Integrated GPS/GIS to the Rescue. *GIS World*, 11.8 : 48–52.

17. Clarsen H. 1993. *Mapping an GIS Technology for Emergency Services*. The Macedon Digest, Australia.

18. Comfort & Chang 1995. A Distributed, Intelligent, Spatial Information System for Disaster Management: A National Model. Paper (delivered) at the 15th Annual ESRI User Conference, Palm Springs.

19. Cova, TJ 1999. GIS in emergency management in Longley, PA, Goodchild, MF, Maguire, DJ & Rhind, DW (eds.) *Geographical Information Systems* – Volume 2 Management Issues and Applications. 2nd Ed, pp845-856. New York: John Wiley & Sons, Inc

20. ESRI, 2002. What is GIS? [online] Available: http://www.gis.com/whatisgis/ [10.08.2002]

21. Famous Quotes, 2002. Famous Quotation Network [online] Available: http://www.famous-quatations.com [03.08.2002]

22. Fowler, M 1996. *The Speed of Light*. Lecture notes delivered at University of Virginia, Physics Department. Virginia [Online]. Available: http://galileoandeinstein.physics.virginia.edu/lectures/spedlite.html [22.9.2002]

23. GIS Development 2002. GIS Support for Man made Disasters [online] Available: http://www.gisdevelopment.net/magazine/gisdev/2001/sep/tms.shtml [10.08.2002]

24. Govender, R 1998. Empty beer bottles on death boat, court told. *Sunday Times*, 22 November.

25. Greene, RW 2002. *Confronting Catastrophe*. Redlands, California: ESRI Press.

26. Haarwood, S 2002. New York City – Creating a Disaster Management GIS on the Fly. *ArcNews*. Winter 2001/2002.

27. Haack-Benedict, C 1998. GIS to the Rescue – How bottom-line functionality has an impact on Emergency Response. Paper (delivered) at the 18th Annual ESRI User Conference, San Diego.

28. Hatcher, GA & Maher, N 1999. Real-time GIS for Marine Applications in Wright, DJ & Bartlett, D (eds.) *Marine and Coastal Geographic Information Systems*, pp137-147. London: Taylor & Francis.

29. Hudson, C 2002. Information given by Regional Director of N.S.R.I to K.C. Meyer during a personal communication on July 16th 2002.

30. Huxhold, WE & Levinsohn, AG 1995. *Managing Geographic Information System Projects*. New York: Oxford University Press.

31. Jacobs, P 2002. Information contained in email correspondence send to K C Meyer from NSRI head Office on September 3rd 2002.

32. JISAO 2002. Joint Institute for the Study of the Atmosphere and Ocean [Online] Available: http://tao.atmos.washington.edu/freud/demo.html [23.9.2002]

33. Johnson, R 2001. GIS Aids Emergency Response. *ArcUser*, 4.3 : 10-11.

34. Kemp, Y 2001. Four escape 'almost certain' death at sea. *Cape Argus*, 16 March.

35. Kimball, DF 2000. Angle (Azimuth) between two points. [Online] Available http://arcscripts.esri.com [14.08.2002].

36. Klinkenberg, B & Cowen, D 2002. University of South Carolina - What is GIS? [Online]. Available: http://www.geog.ubc.ca/courses/klink/gis.notes/ncgia/u01.html [10.08.2002].

37. Laganparsad, M 2002. Sun's Reflection saves fisherman's lives. *The Mercury*, 18 March.

38. Lithgow, T 1995. GIS hits the street in Green, DR, Rix, D & Corbin, C (eds.) *Geographic Information 1996 - The Yearbook of the Association for Geographic Information*, pp. 72-73. London: Taylor & Francis.

39. Mitchell, A 1999. *The ESRI Guide to GIS Spatial Analysis – Volume 1: Geographic Patterns & Relationships*. Redlands, California: ESRI Press.

40. Michelsen, Jr., MW 1997. GPS-Equipped dogs to the Rescue. *GIS World*, 10.3 : 44–46.

41. NOAA (National Oceanic and Atmospheric Administration) 2000. GIS Proves Valuable During Airline Tragedy at Channel Islands. [Online]. Available: http://spatialnews.geocomm.com/features/noaa/alaska.html [10.06.2002].

42. NSRI (National Sea Rescue Institute) 2002(a). About the NSRI. [Online]. Available: http://www.nsri.org.za/default.htm [10.06.2002].

43. NSRI (National Sea Rescue Institute) 2001(b). Our Volunteers. *South Africa's National Sea Rescue Service Information Brochure*, 2001 : 7 – 8.

44. NSRI (National Sea Rescue Institute) 2001(c). Our Rescue Stations. *South Africa's National Sea Rescue Service Information Brochure*, 2001 : 5-6.

45. NSRI (National Sea Rescue Institute) 2001(d). Sea Rescue Station 19 Home Page. [Online]. Available: http://rbay.nsri.org.za [10.08.2002].

46. NSRI (National Sea Rescue Institute) 2001(e). National Sea Rescue Institute - Knysna. [Online]. Available: http://www.knysna.co.za/nsri12/index.html [10.08.2002].

47. Oram, M(ed) 1995. True compass and Magnetic courses. *Navigation in simple steps, Supplement to Motor boats Monthly*, August 1995: 21-22.

48. Peer, N 2002. NSRI condemns beach after sixth drowning. *Cape Times*, 3 July.

49. Pendleton, G 2002. The Fundamentals of GPS. Directions Magazine [Online]. Available: http://www.directionsmag.com/article.php?article_id=228 [22.9.2002]

50. Peuquet, DJ 1999. Time in GIS and geographical databases in Longley, PA, Goodchild, MF, Maguire, DJ & Rhind, DW (eds.) *Geographical Information Systems* – Volume 1 Principles and Technical Issues. 2nd Ed, pp91-100. New York: John Wiley & Sons, Inc.

51. Pratt, M 2000. Modelling Fire Hazard. *ArcUser*, 3.3 : 32.

52. Pratt, M (Ed) 2001. Editor's Page. *ArcUser*, Vol. 4.3 : 4.

53. Pratt, M (Ed) 2002. Editor's Note – Wireless GIS Solutions Aids WTC Rescue Efforts. *ArcUser*, 4.3 : 4.

54. Rowan, H.S 2000. *Marin County's Level of service – A Wildland Fire Risk Assessment Model*. Technical report. Marin County, California.

55. SailingIssues 2002. Navigation Course – SailingIssues.com [Online]. Available www.sailingissues.com/navcourse.html [14.08.2002]

56. Scott III, RH 1998(a). Computer-Aided Dispatch GIS Can Help Save Lives. *GIS World*, 11.10 : 44 - 50

57. Scott III, RH 1998(b). How does Computer Aided Dispatch Work. *GIS World*, 11.10 : 50

58. Shin, J, Bae, K, Jeong, J, Hahm, C & Ryu, J 1999. 1-1-9 Caller Location Information Systems. Paper (delivered) at the 19th Annual ESRI User Conference, San Diego.

59. Spalding, VL 2000. Automating Spatial Process. *ArcUser*, 3.3 : 34.

60. Steede-Terry, K 2000. *Integrating GIS and the Global Positioning System*. Redlands, California: ESRI Press.

61. The Aerospace Corporation 1997. *The Global Positioning System*. Educational Paper No. U.S.A/PAD/7M/Aug 97/RH0039.

62. US ESCAP 2002. United Nations Economic and Social Commission for Asia and the Pacific - Application of New Technology in Population Data Collection, Processing, Dissemination and Presentation [Online]. Available: http://www.unescap.org/stat/pop-it/pop-wit/mo6d.htm [23.09.2002]

63. US Coast Guard, United States of America 1998. *Modeling of Leeway Drift*. Report No. CG-D-06-99: 5.1 – 5.3

64. U.S. Naval Observatory (USNO) 2002: GPS Capabilities [Online]. Available: http://tycho.usno.navy.mil/gpsinfo.html [22.08.2002]

65. Vorobiev, YL (ed) 1998. *Disasters and Man*. Moscow: AST-LTD Publishers.

66. Walsh, T 2000. Aerial Imagery and GIS Data Combine for Effective Prescribed Burning programs. *Earth Observation Magazine (EOM)*, 9.11 : 37-40.

67. Ward, R, Roberts, C & Furness, R 1999. Electronic Chart Display and Information Systems (ECDIS): State-of-the-Art in Nautical Charting in Wright, DJ & Bartlett, D (eds.) *Marine and Coastal Geographic Information Systems*, pp149-161. London: Taylor & Francis.

68. Wilkinson, T 2002. Collapse of the World Trade Center - University of Sydney [Online]. Available: http://www.civil.usyd.edu.au/wtc.htm [10.08.2002].

69. Williams, M 2002. 'Survivors' go missing in mock rescue. *Cape Argus*. 2 March.

70. Wright, DJ 1999. Down to the Sea in Ships: The Emergence of Marine GIS in Wright, DJ & Bartlett, D (eds.) *Marine and Coastal Geographic Information Systems*, pp1-7. London: Taylor & Francis.

71. Zecher, V 2001. The GIS Puzzle: Mapping for E911 and Phase II Wireless. Paper (delivered) at the 21th Annual ESRI User Conference, San Diego.

APPENDICES

1. A_RESCUEVIEW_ADD_CASUALTY

```
'******
'****** CALLED BY:  A_RESCUEVIEW_ADD_CASUALTY_POSITION_DIALOG SUBMIT BUTTON
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*******************************************************************************

theView = av.GetActiveDoc

theTheme = av.GetActiveDoc.FindTheme("Casualty")

if (theTheme = Nil) then

 theFileName = (_theOperationPath + "\Casualty.dbf")

 if (File.Exists(theFileName.asFileName)) then

    if (MsgBox.YesNo("A Casualty file already exist." + NL + "(YES) Continue with current file?" + NL + "(NO)
Overwrite file",
     "RescueVIEW - CASUALTY FILE EXIST",TRUE)) then

     theFileName = (_theOperationPath + "\Casualty.dbf")
     theVtab = Vtab.Make(theFileName.asFileName, TRUE, FALSE)

     IDfield = theVtab.FindField ("ID")
     Xfield = theVtab.FindField  ("X")
     Yfield = theVtab.FindField ("Y")
     Typefield = theVtab.FindField ("Type")
     Targetfield = theVtab.FindField ("Target")
     Callfield = theVtab.FindField ("Called")
     Statusfield = theVtab.FindField ("Status")
     NoteField = theVtab.FindField ("Note")
    Else
     theVtab = Vtab.MakeNew(theFileName.asFileName, DBASE)

     IDfield = Field.Make ("ID", #FIELD_BYTE  , 5, 0)
     Xfield = Field.Make ("X", #FIELD_DECIMAL , 16, 4)
     Yfield = Field.Make ("Y", #FIELD_DECIMAL , 16, 4)
     Typefield = Field.Make ("Type", #FIELD_CHAR  , 50, 0)
     Targetfield = Field.Make ("Target", #FIELD_CHAR  , 50, 0)
     Callfield = Field.Make ("Called", #FIELD_DATE  , 50, 0)
     Statusfield = Field.Make ("Status", #FIELD_CHAR  , 50, 0)
     Notefield = Field.Make ("Note", #FIELD_CHAR  , 250, 0)

     theVtab.AddFields({IDField, XField, YField, TypeField, targetField, CallField, StatusField, NoteField})

    End
```

```
  Else

    theVtab = Vtab.MakeNew(theFileName.asFileName, DBASE)

    IDfield = Field.Make ("ID", #FIELD_BYTE  , 5, 0)
    Xfield = Field.Make ("X", #FIELD_DECIMAL , 16, 4)
    Yfield = Field.Make ("Y", #FIELD_DECIMAL , 16, 4)
    Typefield = Field.Make ("Type", #FIELD_CHAR  , 50, 0)
    Targetfield = Field.Make ("Target", #FIELD_CHAR  , 50, 0)
    Callfield = Field.Make ("Called", #FIELD_DATE  , 50, 0)
    Statusfield = Field.Make ("Status", #FIELD_CHAR  , 50, 0)
    Notefield = Field.Make ("Note", #FIELD_CHAR  , 250, 0)

    theVtab.AddFields({IDField, XField, YField, TypeField, targetField, CallField, StatusField, NoteField})

  End

Else
 theView.DeleteTheme(theTheme)
 theFileName = (_theOperationPath + "\Casualty.dbf")
 theVtab = Vtab.Make(theFileName.asFileName, TRUE, FALSE)

 IDfield = theVtab.FindField ("ID")
 Xfield = theVtab.FindField  ("X")
 Yfield = theVtab.FindField ("Y")
 Typefield = theVtab.FindField ("Type")
 Targetfield = theVtab.FindField ("Target")
 Callfield = theVtab.FindField ("Called")
 Statusfield = theVtab.FindField ("Status")
 Notefield = theVtab.FindField ("Note")
End

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_ADD_CASUALTY_POSITION_DIALOG")

theCDEGY = (theDialog.GetDialog.FindByName("CtxtDegreeS").GetText).asNumber
theCMINY = (theDialog.GetDialog.FindByName("CtxtDMinuteS").GetText).asNumber

theCDEGX = (theDialog.GetDialog.FindByName("CtxtDegreeW").GetText).asNumber
theCMINX = (theDialog.GetDialog.FindByName("CtxtDMinuteW").GetText).asNumber

if (theCMinX >= 1) then
  theCount = (theCMinX.asString).Count

  if (TheCount = 1) then
    theCMinX = theCMinX / 1
  Elseif (TheCount = 2) then
    theCMinX = theCMinX / 10
  Elseif (TheCount = 3) then
    theCMinX = theCMinX / 100
  Elseif (TheCount = 4) then
    theCMinX = theCMinX / 1000
  Elseif (TheCount = 5) then
    theCMinX = theCMinX / 10000
  Elseif (TheCount = 6) then
    theCMinX = theCMinX / 100000
  Else
     MsgBox.Error("Coordinate cannot be more than 6 digits"+NL+"Should be entered as 0.3456 if more","")
   End
End
```

```
if (theCMinY >= 1) then
   theCount = (theCMinY.asString).Count

   if (TheCount = 1) then
     theCMinY = theCMinY / 1
   Elseif (TheCount = 2) then
     theCMinY = theCMinY / 10
   Elseif (TheCount = 3) then
     theCMinY = theCMinY / 100
   Elseif (TheCount = 4) then
     theCMinY = theCMinY / 1000
   Elseif (TheCount = 5) then
     theCMinY = theCMinY / 10000
   Elseif (TheCount = 6) then
     theCMinY = theCMinY / 100000
   Else
      MsgBox.Error("Coordinate cannot be more than 6 digits"+NL+"Should be entered as 0.3456 if more","")
    End
End


theX1 = theCDEGX +  (theCMINX)
theY1 = theCDEGY -  (theCMINY)


theVtab.SetEditable(TRUE)

theBitmap = theVtab.GetSelection

If (theVtab.IsEditable) then

   rec = theVtab.AddRecord

   theVtab.SetValue(IDfield, rec, Rec)
   theVtab.SetValue(XField, rec, theX1)
   theVtab.SetValue(YField, rec, (theY1))
   theVtab.SetValue(TypeField, rec, theDialog.GetDialog.FindByName("cBoxCallType").GetCurrentValue)
   theVtab.SetValue(TargetField, rec, theDialog.GetDialog.FindByName("cTxtTarget").GetText)
   theVtab.SetValue(CallField, rec, (Date.Now).SetFormat("yyyy/mm/dd hh\Hm:s"))
   theVtab.SetValue(StatusField, rec, theDialog.GetDialog.FindByName("cBoxStatus").GetCurrentValue)
   theVtab.SetValue(NoteField, rec, theDialog.GetDialog.FindByName("ctboxNote").GetText)
End

theVtab.SetEditable(False)

theVtab.Flush

thexy = XYName.Make(theVtab, XField, YField)
theTheme = Theme.make(theXY)
theView.AddTheme(theTheme)

theTheme.SetName("Casualty")

theLegend = theTheme.GetLegend
theLegendFile = (System.GetEnvVar("RESCUEVIEW") + "\Data\Casualty.avl").asFilename
theLegend.Load(theLegendFile, #LEGEND_LOADTYPE_ALL)

theTheme.SetVisible(TRUE)

SELF.GetDialog.Close
```

```
av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theX1, theY1} )

'*****************************************************************************************************
'************************ ADD LOG FILE ENTRY ********************************************************
'*****************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -NEW RESOURCE, -UPDATE RESOURCE, -CASUALTY ADDED, -
CASUALTY UPDATE
'*****************************************************************************************************

  theType = "CASUALTY ADDED"
  theXValue = theX1
  theYValue = theY1
  theMessage = "CASUALTY ADDED : " + theDialog.GetDialog.FindByName("cTxtTarget").GetText + "at
POSITION " + theX1.asString + " and " + theY1.asString

  av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************************
'************************ END LOG FILE ENTRY ********************************************************
'*****************************************************************************************************
```

2. A_RESCUEVIEW_ADD_CASUALTY_DIALOG

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE: A_RESCUEVIEW_ADD_CASUALTY_BUTTON
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

SELF.getDialog.Close

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_ADD_CASUALTY_POSITION_DIALOG")

theFileName = (System.GetEnvVar("RESCUEVIEW") + "\Data\CallType.dbf")
theCallVtab = Vtab.Make(theFileName.asFileName, FALSE, FALSE)
theCapture = theCallVTab.findField("Type")

theComboBox = (theDialog.GetDialog).FindByName("cBoxCallType")

theComboBox.DefineFromVTab (theCallVTab, theCapture, False)

theFileName = (System.GetEnvVar("RESCUEVIEW") + "\Data\CStatus.dbf")
theCallVtab = Vtab.Make(theFileName.asFileName, FALSE, FALSE)
theStatus = theCallVTab.findField("Status")

theComboBox = (theDialog.GetDialog).FindByName("cBoxStatus")

theComboBox.DefineFromVTab (theCallVTab, theStatus, False)

theDialog.getDialog.Open
```

3. A_RESCUEVIEW_ADD_COMPASS_ROSE

```
'******
'****** CALLED BY:  A_RESCUEVIEW_COURSE_TO_STEER
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*************************************************************************

theView = SELF.Get(0)

thePnt = SELF.Get(1)

theViewExtent = theView.GetDisplay.ReturnExtent

theHeight = theViewExtent.GetHeight / 2

shrink = 0.005

deg = 15

' some constants (change if nessesary)
txtspc = 0.01  ' text mover to place on the line (for single digits)
txtspc2 = 0.019 ' same as above for double digits
thefont = font.make ("Helvetica","Normal")
textsize = 32
rings = 5
sumdeg = 360 / deg

' determine user rect characteristics for drawing
inc = ((theheight / 2) / rings)

cnt = 0
i = inc
for each num in 0..(rings -1)

 if (cnt = 0) then
   cir = circle.make (thePnt, (inc))
   gra = graphicshape.make (cir)
   gra.setselected (TRUE)
   theview.getgraphics.add (gra)

 else

   cir = circle.make (thePnt, (inc + i))
   gra = graphicshape.make (cir)
   gra.setselected (TRUE)
   theview.getgraphics.add (gra)
   inc = inc + i
 end

 cnt = cnt + 1
end
```

```
lastcir = cir
pntlst = lastcir.asmultipoint.aslist

' ***************************
' adjust the point list from the largest circle to make 0 start at the top
count = 0
deglst = {}

for each pnt in pntlst
  if (count >= 270) then
    deglst.add (pnt)
  end
  count = count + 1
end

count = 0

for each pnt in pntlst
  if (count < 270) then
    deglst.add (pnt)
  end
  count = count + 1
end
newlst = {}
count = 0

for each rec in deglst
  newlst.add (count)
  count = count + 1
end

'*****************************
' reverse list so 90 is on the right side

count = (deglst.count - 1)

for each pnt in deglst
  newlst.set (count,pnt)
  count = count - 1
end

' interval lines
count = 0
cnt = deg
linelst = {}

theDegtxt = 360 - 180

for each pnt in deglst

  if (cnt = deg) then

    aline = line.make (thePnt,pnt)
    gra = graphicshape.make (aLine)
    gra.setselected (TRUE)

    theview.getgraphics.add (gra)

    av.Run("A_RESCUEVIEW_ADD_COMPASS_ROSE_VALUES", {theView, Pnt, theDegtxt} )
    theDegtxt = theDegtxt - 15
```

```
    count = count + 1
    cnt = 0
    linelst.add (aline)
   end
   cnt = cnt + 1

end

theview.getgraphics.groupselected

'theSelectShape = (theView.getGraphics.GetSelected).getShape

theview.getgraphics.unselectall
theview.getgraphics.endbatch
av.clearmsg
av.clearstatus
```

4. A_RESCUEVIEW_ADD_COMPASS_ROSE_VALUES

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

v = SELF.Get(0)   'av.GetActiveDoc
p = SELF.Get(1)   'v.GetDisplay.ReturnUserPoint
txt = SELF.Get(2)  'Text to display

if (txt < 0) then
  txt = txt + 360
End

paramv1 = 0
paramv2 = 0
paramv3 = 0
paramv4 = 0
paramv5 = 0
paramv6 = 0

  '** Text Adjustments represent the percentage label point size
  '** is of the total point

  paramt1 = 0.55
  paramt2 = 0.55
  paramt3 = 0.50
  paramt4 = 0.45
  paramt5 = 0.35
  paramt6 = 0.3

mrk1 = BasicMarker.Make
mrk1.SetStyle(#BASICMARKER_STYLE_PATTERN)
mrk1.SetFont(Font.Make("ESRI Oil, Gas, & Water","Normal"))
mrk1.SetCharacter(244)
mrk1.SetColor(color.GetYellow)
mrk1.SetColorLock(true)
mrk2 = BasicMarker.Make
mrk2.SetStyle(#BASICMARKER_STYLE_PATTERN)
mrk2.SetFont(Font.Make("ESRI Oil, Gas, & Water","Normal"))
mrk2.SetCharacter(243)
mrk2.SetColor(Color.GetBlack)
mrk2.SetColorLock(true)

Hinfo = {p,"HTool5",{paramv1,paramv2,paramv3,paramv4,paramv5,paramv6},
         {paramt1,paramt2,paramt3,paramt4,paramt5,paramt6},
         {mrk1,mrk2}, txt.asString}

x = av.Run("A_RESCUEVIEW_ADD_COMPASS_ROSE_VALUES_COMMIT", Hinfo)
```

5. A_RESCUEVIEW_ADD_COMPASS_ROSE_VALUES_COMMIT

```
'******
'****** CALLED BY:  A_RESCUEVIEW_ADD_COMPASS_ROSE_VALUES
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

p = Self.Get(0)
theDegTxt = Self.Get(5)
v = av.GetActiveDoc

MrkSize = 24

HwyStrSize = theDegTxt.Count

vAdj = Self.Get(2).Get(HwyStrSize - 1)
v.GetGraphics.UnSelectAll

for each xx in Self.Get(4)
  if ((Self.Get(1) = "HTool5") and (HwyStrSize > 2))then
    symAdj = 4
    oldChar = xx.GetCharacter
    newChar = oldChar + symAdj
    xx.SetCharacter(newChar)
  end
  Shld = GraphicShape.Make(p.Clone)
  xx.SetSize(MrkSize)
  v.GetDisplay.HookUpSymbol(xx)
  Shld.SetSymbol(xx)
  Shld.SetDisplay(v.GetDisplay)
  v.GetGraphics.AddBatch(Shld)
  Shld.SetSelected(true)
  Shld.SetName("HwyShield")
end

v.GetGraphics.GroupSelected
Shield = v.GetGraphics.GetSelected.Get(0)
Shield.SetSelected(false)
Shield.SetName("HwyShield")

tmrkSize = MrkSize  * Self.Get(3).Get(HwyStrSize -1)

gexp = GraphicShape.Make(0@0)
b = BasicMarker.Make
b.SetStyle(#BASICMARKER_STYLE_PATTERN)
b.SetFont(Font.Make("#FONT_TIMEB","BOLD"))
b.SetCharacter("O".AsAscii)
b.SetSize(tmrkSize)
b.SetColor(Color.GetRed)
gexp.SetSymbol(b)
gexp.SetDisplay(v.GetDisplay)
cw = gexp.GetBounds.GetWidth
```

```
h = Shield.GetBounds.GetHeight/2
MaxW = Shield.GetBounds.GetWidth * (0.6667)
MaxCW = MaxW/HwyStrSize
ActualOffset = cw - (HwyStrSize*cw)
ActualOffset = (HwyStrSize - 1)*cw/4
t_o = p - ( ActualOffset@(h*vAdj) )

for each c in 0..(HwyStrSize - 1)
  b = BasicMarker.Make
  b.SetStyle(#BASICMARKER_STYLE_PATTERN)
  b.SetFont(Font.Make("#FONT_TIMEB","BOLD"))
  b.SetCharacter(theDegTxt.Middle(c,1).AsASCII)
  b.SetSize(tmrkSize)
  b.SetColor(Color.GetBlue)
  v.GetDisplay.HookUpSymbol(b)
  ng = GraphicShape.Make(t_o)
  ng.SetSymbol(b)
  ng.SetDisplay(v.GetDisplay)
  t_o = t_o + ((cw/2)@0)
  ng.SetName("HwyShield")
  v.GetGraphics.AddBatch(ng)
  ng.SetSelected(true)
end

v.GetGraphics.GroupSelected
NumsWidth = v.GetGraphics.GetSelected.Get(0).GetBounds.GetWidth
ShldWidth = Shield.GetBounds.GetWidth
NumsOrig = v.GetGraphics.GetSelected.Get(0).GetBounds.ReturnOrigin
ShldOrig = Shield.GetBounds.ReturnOrigin
Shield.SetSelected(true)
v.GetGraphics.GroupSelected
v.GetGraphics.GetSelected.Get(0).SetName("HwyShield")
v.GetGraphics.GetSelected.Get(0).SetUniformScaling(true)

v.GetGraphics.EndBatch
```

6. A_RESCUEVIEW_ADD_LOG_MESSAGE

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theMessage = MsgBox.Input("Please enter log Report","RescueVIEW - LOG ENTRY","")

'*****************************************************************************************************
'************************ ADD LOG FILE ENTRY ***********************************************************
'*****************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -NEW RESOURCE, -UPDATE RESOURCE
'*****************************************************************************************************

if (theMessage <> nil) then

  theType = "USER"
  theXValue = Nil
  theYValue = Nil

  av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

End

'*****************************************************************************************************
'************************ END LOG FILE ENTRY ***********************************************************
'*****************************************************************************************************
```

7. A_RESCUEVIEW_ADD_RESOURCE

```
'******
'****** CALLED BY:  A_RESCUEVIEW_ADD_RESOURCE_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_ADD_RESOURCE_POSITION_DIALOG")

if (theDialog.GetDialog.FindByName("cTxtCallSign").GetText = nil) then
  MsgBox.Error("No Callsign Assigned to resource","")
  EXIT

End

theView = av.GetActiveDoc

theTheme = av.GetActiveDoc.FindTheme("RESOURCE")

if (theTheme = Nil) then

  theFileName = (_theOperationPath + "\RESOURCE.dbf")

  if (File.Exists(theFileName.asFileName)) then

     if (MsgBox.YesNo("A RESOURCE file already exist." + NL + "(YES) Continue with current file?" + NL + "(NO)
Overwrite file",
       "RescueVIEW - RESOURCE FILE EXIST",TRUE)) then

       theFileName = (_theOperationPath + "\RESOURCE.dbf")
       theVtab = Vtab.Make(theFileName.asFileName, TRUE, FALSE)

       IDfield = theVtab.FindField ("ID")
       Xfield = theVtab.FindField  ("X")
       Yfield = theVtab.FindField ("Y")
       Typefield = theVtab.FindField ("Type")
       Callsignfield = theVtab.FindField ("Callsign")
       Callfield = theVtab.FindField ("Called")
       Statusfield = theVtab.FindField ("Status")
       NoteField = theVtab.FindField ("Note")
     Else
       theVtab = Vtab.MakeNew(theFileName.asFileName, DBASE)

       IDfield = Field.Make ("ID", #FIELD_BYTE  , 5, 0)
       Xfield = Field.Make ("X", #FIELD_DECIMAL , 16, 4)
       Yfield = Field.Make ("Y", #FIELD_DECIMAL , 16, 4)
       Typefield = Field.Make ("Type", #FIELD_CHAR  , 50, 0)
       Callsignfield = Field.Make ("Callsign", #FIELD_CHAR  , 50, 0)
       Callfield = Field.Make ("Called", #FIELD_DATE  , 50, 0)
       Statusfield = Field.Make ("Status", #FIELD_CHAR  , 50, 0)
       Notefield = Field.Make ("Note", #FIELD_CHAR  , 250, 0)
```

```
      theVtab.AddFields({IDField, XField, YField, TypeField, CallsignField, CallField, StatusField, NoteField})

    End

  Else

    theVtab = Vtab.MakeNew(theFileName.asFileName, DBASE)

    IDfield = Field.Make ("ID", #FIELD_BYTE  , 5, 0)
    Xfield = Field.Make ("X", #FIELD_DECIMAL , 16, 4)
    Yfield = Field.Make ("Y", #FIELD_DECIMAL , 16, 4)
    Typefield = Field.Make ("Type", #FIELD_CHAR  , 50, 0)
    Callsignfield = Field.Make ("Callsign", #FIELD_CHAR  , 50, 0)
    Callfield = Field.Make ("Called", #FIELD_DATE  , 50, 0)
    Statusfield = Field.Make ("Status", #FIELD_CHAR  , 50, 0)
    Notefield = Field.Make ("Note", #FIELD_CHAR  , 250, 0)

    theVtab.AddFields({IDField, XField, YField, TypeField, CallsignField, CallField, StatusField, NoteField})

   End

Else
 theView.DeleteTheme(theTheme)
 theFileName = (_theOperationPath + "\RESOURCE.dbf")
 theVtab = Vtab.Make(theFileName.asFileName, TRUE, FALSE)

 IDfield = theVtab.FindField ("ID")
 Xfield = theVtab.FindField  ("X")
 Yfield = theVtab.FindField ("Y")
 Typefield = theVtab.FindField ("Type")
 Callsignfield = theVtab.FindField ("Callsign")
 Callfield = theVtab.FindField ("Called")
 Statusfield = theVtab.FindField ("Status")
 Notefield = theVtab.FindField ("Note")
End


theCDEGY = (theDialog.GetDialog.FindByName("CtxtDegreeS").GetText).asNumber
theCMINY = (theDialog.GetDialog.FindByName("CtxtDMinuteS").GetText).asNumber

theCDEGX = (theDialog.GetDialog.FindByName("CtxtDegreeW").GetText).asNumber
theCMINX = (theDialog.GetDialog.FindByName("CtxtDMinuteW").GetText).asNumber

if (theCMinX >= 1) then
  theCount = (theCMinX.asString).Count

  if (TheCount = 1) then
    theCMinX = theCMinX / 1
  Elseif (TheCount = 2) then
    theCMinX = theCMinX / 10
  Elseif (TheCount = 3) then
    theCMinX = theCMinX / 100
  Elseif (TheCount = 4) then
    theCMinX = theCMinX / 1000
  Elseif (TheCount = 5) then
    theCMinX = theCMinX / 10000
  Elseif (TheCount = 6) then
    theCMinX = theCMinX / 100000
  Else
     MsgBox.Error("Coordinate cannot be more than 6 digits"+NL+"Should be entered as 0.3456 if more","")
```

```
    End
End

if (theCMinY >= 1) then
   theCount = (theCMinY.asString).Count

   if (TheCount = 1) then
     theCMinY = theCMinY / 1
   Elseif (TheCount = 2) then
     theCMinY = theCMinY / 10
   Elseif (TheCount = 3) then
     theCMinY = theCMinY / 100
   Elseif (TheCount = 4) then
     theCMinY = theCMinY / 1000
   Elseif (TheCount = 5) then
     theCMinY = theCMinY / 10000
   Elseif (TheCount = 6) then
     theCMinY = theCMinY / 100000
   Else
      MsgBox.Error("Coordinate cannot be more than 6 digits"+NL+"Should be entered as 0.3456 if more","")
    End
End

theX1 = theCDEGX +  (theCMINX)
theY1 = theCDEGY -  (theCMINY)

theVtab.SetEditable(TRUE)

theBitmap = theVtab.GetSelection

If (theVtab.IsEditable) then

   rec = theVtab.AddRecord

   theVtab.SetValue(IDfield, rec, Rec)
   theVtab.SetValue(XField, rec, theX1)
   theVtab.SetValue(YField, rec, (theY1))
   theVtab.SetValue(TypeField, rec, theDialog.GetDialog.FindByName("cBoxResourceType").GetCurrentValue)
   theVtab.SetValue(CallsignField, rec, theDialog.GetDialog.FindByName("cTxtCallsign").GetText)
   theVtab.SetValue(CallField, rec, (Date.Now).SetFormat("yyyy/mm/dd hh\Hm:s"))
   theVtab.SetValue(StatusField, rec, theDialog.GetDialog.FindByName("cBoxStatus").GetCurrentValue)
   theVtab.SetValue(NoteField, rec, theDialog.GetDialog.FindByName("ctboxNote").GetText)
End

theVtab.SetEditable(False)

theVtab.Flush

thexy = XYName.Make(theVtab, XField, YField)
theTheme = Theme.make(theXY)
theView.AddTheme(theTheme)

theTheme.SetName("RESOURCE")

theLegend = theTheme.GetLegend
theLegendFile = (System.GetEnvVar("RESCUEVIEW") + "\Data\RESOURCE.avl").asFilename
theLegend.Load(theLegendFile, #LEGEND_LOADTYPE_ALL)

theTheme.SetVisible(TRUE)
```

```
SELF.GetDialog.Close

av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theX1, theY1} )

'*****************************************************************************************************
'************************ ADD LOG FILE ENTRY ********************************************************
'*****************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*****************************************************************************************************

  theType = "RESOURCE ADDED"
  theXValue = theX1
  theYValue = theY1
  theMessage = theDialog.GetDialog.FindByName("cTxtCallSign").GetText + "at POSITION " + theX1.asString + "
and " + theY1.asString

  av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************************
'************************ END LOG FILE ENTRY ********************************************************
'*****************************************************************************************************
```

8. A_RESCUEVIEW_ADD_RESOURCE_DIALOG

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE: A_RESCUEVIEW_ADD_RESOURCES
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

SELF.getDialog.Close

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_ADD_RESOURCE_POSITION_DIALOG")

theFileName = (System.GetEnvVar("RESCUEVIEW") + "\Data\ResourceType.dbf")
  theCallVtab = Vtab.Make(theFileName.asFileName, FALSE, FALSE)
  theCapture = theCallVTab.findField("Type")
  theComboBox = (theDialog.GetDialog).FindByName("cBoxResourceType")
  theComboBox.DefineFromVTab (theCallVTab, theCapture, False)

theFileName = (System.GetEnvVar("RESCUEVIEW") + "\Data\RStatus.dbf")
  theCallVtab = Vtab.Make(theFileName.asFileName, FALSE, FALSE)
  theStatus = theCallVTab.findField("Status")
  theComboBox = (theDialog.GetDialog).FindByName("cBoxStatus")
  theComboBox.DefineFromVTab (theCallVTab, theStatus, False)

theDialog.getDialog.Open
```

9. A_RESCUEVIEW_RESOURCE_DIALOG_UPDATE

```
'******
'****** CALLED BY:  A_RESCUEVIEW_ADD_RESOURCE_POSITION_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = SELF.GetDialog
theName = SELF.GetName

if (theName = "CtxtMinuteW") then
  theMin = (theDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

Elseif (theName = "CtxtMinuteS") then
  theMin = (theDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

End

if (theName = "CtxtSecondW") then
  theMin = (theDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

Elseif (theName = "CtxtSecondS") then
  theMin = (theDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

End
```

10. A_RESCUEVIEW_BEARING

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: David F. Kimball  Modified by :Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

theFromPnt = Self.Get(0)
theToPNt = = Self.Get(1)
theView = Self.Get(2)

aDisplay = theView.getDisplay

theLine = line.Make(theFromPnt, theToPnt)

  theDist = theLine.ReturnLength
  theDist.SetFormat("d.dd").AsString

  firstpt = theLine.Along (0)
  midpt   = theLine.Along (50)
  lastpt  = theLine.Along (100)

  sx = firstpt.GetX
  sy = firstpt.Gety

  ex = lastpt.GetX
  ey = lastpt.GetY

  dx = ex-sx
  dy = ey-sy

  h = dy.abs
  w = dx.abs

  if ((dx >= 0 ) and (dy >= 0 )) then
   F = "N"
   L = "E"
   Q = 0
   O = 0
  end
  if ((dx >= 0 ) and (dy <= 0 )) then
   F = "S"
   L = "E"
   Q = 180
   O = 180
  end
  if ((dx <= 0 ) and (dy <= 0 )) then
   F = "S"
   L = "W"
   Q = 0
   O = -180
```

```
  end
  if ((dx <= 0 ) and (dy >= 0 )) then
   F = "N"
   L = "W"
   Q = 180
   O = 360
  end

 'Angle
  ta = w/h
  ar = ta.aTan
  theAngle = ar.AsDegrees
  a = (theAngle-Q).abs
  ao = (theAngle-O).abs.truncate

 'Deg, Min, Sec
  theDeg = (theAngle).truncate
  theMin = ((theAngle - theDeg) * 60).truncate
  theSec = ((((theAngle - theDeg) * 60) - theMin) * 60).truncate
  if (theDeg < 10) then
   theDeg = ("0"+theDeg.asString)
  end
  if (theMin < 10) then
   theMin = ("0"+theMin.asString)
  end
  if (theSec < 10) then
   theSec = ("0"+theSec.asString)
  end
  msg = ("Bearing ="++F++theDeg.asString+"°"+theMin.asString+"'"+theSec.asString+"'"++L++"  Distance
="++theDist.asString+""++"  Angle ="++ao.asString+"°")
  av.ShowMsg(msg)

  aString = (F++theDeg.asString+"°"+theMin.asString+"'"+theSec.asString+"'"++L+NL+theDist.asString+"")

  t = GraphicText.Make(aString, midpt)
  t.SetAlignment(#TEXTCOMPOSER_JUST_CENTER)
  t.SetSpacing (1.2)
  t.SetDisplay(theView.GetDisplay)
  tsym = t.GetSymbol
  tsym.SetSize (8)
  theView.GetDisplay.HookupSymbol(t.GetSymbol)

  tp = TextPositioner.Make(t.GetClass)
  tp.SetHAlign(#TEXTPOSITIONER_HALIGN_CENTER)
  tp.SetVAlign(#TEXTPOSITIONER_VALIGN_ON)
  tp.Calculate(theLine,t.GetExtent, 1, nil)

  t.SetOrigin(tp.GetOrigin)
  t.SetAngle(90-a)
  theView.GetGraphics.Add(t)
```

11. A_RESCUEVIEW_CASUALTY_ADD_POINT_TO

```
'******
'****** CALLED BY:  A_RESCUEVIEW_ADD_CASUALTY_POSITION_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_ADD_CASUALTY_POSITION_DIALOG")

theName = SELF.GetName

theView = av.GetActiveDoc

thePoint = theView.GetDisplay.ReturnUserPoint

theXValue = thePoint.GetX
theYValue = thePoint.GetY


theXDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theXValue})
theYDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theYValue})


theDialog.GetDialog.FindByName("CtxtDegreeS").SetText(theYDMS.Extract(0))
theDialog.GetDialog.FindByName("CtxtDegreeW").SetText(theXDMS.Extract(0))
theDialog.GetDialog.FindByName("CtxtMinuteS").SetText(theYDMS.Extract(1))
theDialog.GetDialog.FindByName("CtxtMinuteW").SetText(theXDMS.Extract(1))

theDialog.GetDialog.FindByName("CtxtSecondS").SetText(theYDMS.Extract(2))
theDialog.GetDialog.FindByName("CtxtSecondW").SetText(theXDMS.Extract(2))

'********************** UPDATE DD FROM DMS VALUES ***************

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
```

```
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

theDialog.GetDialog.Open
```

12. A_RESCUEVIEW_CASUALTY_DIALOG_UPDATE

```
'******
'****** CALLED BY:  A_RESCUEVIEW_ADD_CASUALTY_POSITION_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = SELF.GetDialog
theName = SELF.GetName

if (theName = "CtxtMinuteW") then
  theMin = (theDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

Elseif (theName = "CtxtMinuteS") then
  theMin = (theDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

End

if (theName = "CtxtSecondW") then
  theMin = (theDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

Elseif (theName = "CtxtSecondS") then
  theMin = (theDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

End
```

13. A_RESCUEVIEW_CASUALTY_FLASH_POSITION

```
'******
'****** CALLED BY:  A_RESCUEVIEW_CASUALTY_FLASH_POSITION_BUTTON
'******
'****** CALLS MADE: A_RESCUEVIEW_POINT_DISPLAY
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*************************************************************************

theView = av.GetActiveDoc

theCTheme = theView.FindTheme("Casualty")

if (theCTheme = nil) then
  MsgBox.Error("No Casualty position has been logged","")
  EXIT
End

If (theCTheme.GetName <> "Casualty") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

End

theCFtab = theCTheme.GetFTab

'theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theXField = theCFtab.FindField("X")
theYField = theCFtab.FindField("Y")
theTargetField = theCFtab.FindField("Target")

theCBitmap = TheCFtab.GetSelection

theCList = {}

for each rec in theCFtab
  theCIDValue = theCFtab.ReturnValue(theIDField, rec)
  theCTargetValue = theCFtab.ReturnValue(theTargetField, rec)
  theCList.Add(theCIDValue.asString + " --- " + theCTargetValue )
End

theChoice = MsgBox.ChoiceAsString (theCList, "Please select Casualty to flash position", "RescueVIEW - Casualty
Postion locator")

if (theChoice = Nil) then
  Exit
End

SearchStr = theChoice.Extract(0)
theQuery = "[ID] = " + SearchStr
theCFTab.Query(theQuery, theCBitmap, #VTAB_SELTYPE_NEW)
theCFTab.UpdateSelection
```

```
theCount = theCBitmap.Count

For each uprec in theCBitmap
  theXValue = theCFtab.ReturnValue(theXField, uprec)
  theYValue = theCFtab.ReturnValue(theYField, uprec)
End

theDisplay = theView.GetDisplay
theExtent = theDisplay.ReturnExtent

theVisible = theExtent.Intersects (theXValue@theYValue)

if (theVisible) then

  av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theXValue, theYValue} )

Else

  theDisplay.PanTo (theXValue@theYValue)
  av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theXValue, theYValue} )

End

theCBitmap.ClearAll
```

14. A_RESCUEVIEW_CASUALTY_POSITION_MANUAL_UPDATE

```
'******
'****** CALLED BY:  A_RESCUEVIEW_UPDATE_CASUALTY_POSITION_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theView = av.GetActiveDoc

theCTheme = theView.FindTheme("Casualty")

if (theCTheme = nil) then
  MsgBox.Error("No Casualty position has been logged","")
  EXIT
End

If (theCTheme.GetName <> "Casualty") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

End

theCFtab = theCTheme.GetFTab

theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theXField = theCFtab.FindField("X")
theYField = theCFtab.FindField("Y")
theTypeField = theCFtab.FindField("Type")
theTargetField = theCFtab.FindField("Target")
'theCalledField = theCFtab.FindField("Called")
theStatusField = theCFtab.FindField("Status")
theNoteField = theCFtab.FindField("Note")

theCBitmap = TheCFtab.GetSelection

theCount = theCBitmap.Count

theTargetValue = SELF.GetDialog.FindByName("cTxtTarget").GetText

theRDEGX = (SELF.GetDialog.FindByName("CtxtDegreeW").GetText).asNumber
theRMINX = (SELF.GetDialog.FindByName("CtxtDMinuteW").GetText).asNumber

theRDEGY = (SELF.GetDialog.FindByName("CtxtDegreeS").GetText).asNumber
theRMINY = (SELF.GetDialog.FindByName("CtxtDMinuteS").GetText).asNumber

theTypeValue = SELF.GetDialog.FindByName("cBoxType").GetCurrentValue
theStatusValue = SELF.GetDialog.FindByName("cBoxStatus").GetCurrentValue
theNote = SELF.GetDialog.FindByName("cBoxNote").GetText
```

```
theMessage = ""

theCFtab.SetEditable(TRUE)

If (theCFtab.IsEditable) then

 for each uprec in theCBitmap

  if ((SELF.GetDialog.FindByName("CtxtDegreeW").GetText <> "") or
(SELF.GetDialog.FindByName("CtxtDegreeS").GetText <> ""))then
   theXValue = theRDEGX + theRMINX
   theYValue = theRDEGY - theRMINY

   theOldShapeValue = theCFtab.returnValue(theShapeField, uprec)
   theNewShapeValue = theXValue@theYValue

   if (theOldShapeValue <> theNewShapeValue) then

     theCFtab.SetValue(theShapeField, uprec, theNewShapeValue)
     theCFtab.SetValue(theXField, uprec, theXValue)
     theCFtab.SetValue(theYField, uprec, theYValue)

     if (theXValue > 0) then
      theXDegree = theXValue.asString ++ "EAST"
     Else
      theXDegree = theXValue.asString ++ "WEST"
     End

     if (theYValue > 0) then
      theYDegree = theYValue.asString ++ "NORTH"
     Else
      theYDegree = theYValue.asString ++ "SOUTH"
     End


     av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theXValue, theYvalue} )

     '*****************************************************************************************************
     '************************ ADD LOG FILE ENTRY ***********************************************************
     '*****************************************************************************************************
     '*** ENTRY TYPES
     '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED,
-UPDATE CASUALTY
     '*****************************************************************************************************

      theType = "UPDATE CASUALTY"
      'theXValue = theXValue
      'theYValue = theYValue
      theMessage = "Position Update to " + theXDegree + " and " + theYDegree

      av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

     '*****************************************************************************************************
     '************************ END LOG FILE ENTRY ***********************************************************
     '*****************************************************************************************************

   End
  End
```

```
  theOldTypeValue = theCFtab.returnValue(theTypeField, uprec)
  theOldStatusValue = theCFtab.returnValue(theStatusField, uprec)

  if (theTypeValue <> theOldTypeValue) then
    theCFtab.SetValue(theTypeField, uprec, theTypeValue)
    theMessage = "Type Update from " + theOldTypeValue + " to " + theTypeValue

    '*****************************************************************************************
    '********************** ADD LOG FILE ENTRY ***********************************************
    '*****************************************************************************************
    '*** ENTRY TYPES
    '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
    '*****************************************************************************************

      theType = "UPDATE CASUALTY"
      theXValue = nil
      theYValue = nil

      av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

    '*****************************************************************************************
    '********************** END LOG FILE ENTRY ***********************************************
    '*****************************************************************************************
  End

  if (theStatusValue <> theOldStatusValue) then
    theCFtab.SetValue(theStatusField, uprec, theStatusValue)
    theMessage = "Status Update from " + theOldStatusValue + " to " + theStatusValue

    '*****************************************************************************************
    '********************** ADD LOG FILE ENTRY ***********************************************
    '*****************************************************************************************
    '*** ENTRY TYPES
    '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
    '*****************************************************************************************

      theType = "UPDATE CASUALTY"
      theXValue = nil
      theYValue = nil

      av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

    '*****************************************************************************************
    '********************** END LOG FILE ENTRY ***********************************************
    '*****************************************************************************************
  End


  theOldTargetValue = theCFtab.returnValue(theTargetField, uprec)
  theOldNoteValue = theCFtab.returnValue(theNoteField, uprec)

  if (theTargetValue <> theOldTargetValue) then
    theCFtab.SetValue(theTargetField, uprec, theTargetValue)
    theMessage = "Name Update from " + theOldtargetValue + " to "
+SELF.GetDialog.FindByName("cTxtTarget").GetText

    '*****************************************************************************************
    '********************** ADD LOG FILE ENTRY ***********************************************
```

```
    '*****************************************************************************************************
    '*** ENTRY TYPES
    '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
    '*****************************************************************************************************

      theType = "UPDATE CASUALTY"
      theXValue = nil
      theYValue = nil

      av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

    '*****************************************************************************************************
    '********************** END LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************

   End

   if (theNote <> theOldNoteValue) then
    theCFtab.SetValue(theNoteField, uprec, theNote)
    theMessage = "Note Update : " +NL+ theNote

    '*****************************************************************************************************
    '********************** ADD LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************
    '*** ENTRY TYPES
    '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
    '*****************************************************************************************************

      theType = "UPDATE CASUALTY"
      theXValue = nil
      theYValue = nil

      av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

    '*****************************************************************************************************
    '********************** END LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************

   End

  End
End

theCFtab.SetEditable(False)

theCFtab.Flush
theCBitmap.ClearAll
theCTheme.Invalidate(TRUE)

SELF.GetDialog.Close
```

15. A_RESCUEVIEW_CASUALTY_POSITION_UPDATE_DIALOG

```
'******
'****** CALLED BY:  A_RESCUEVIEW_CASUALTY_POSITION_UPDATE_BUTTON
'******
'****** CALLS MADE: A_RESCUEVIEW_CONVERT_DD_TO_DMS
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

SELF.GetDialog.Close

theView = av.GetActiveDoc

theCTheme = theView.FindTheme("Casualty")

if (theCTheme = nil) then
  MsgBox.Error("No Casualty position has been logged","")
  EXIT
End

If (theCTheme.GetName <> "Casualty") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

End

theCFtab = theCTheme.GetFTab

'theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theXField = theCFtab.FindField("X")
theYField = theCFtab.FindField("Y")
theTypeField = theCFtab.FindField("Type")
theTargetField = theCFtab.FindField("Target")
theCalledField = theCFtab.FindField("Called")
theFoundField = theCFtab.FindField("Found")
theStatusField = theCFtab.FindField("Status")
theNoteField = theCFtab.FindField("Note")

theCBitmap = TheCFtab.GetSelection

theCList = {}

for each rec in theCFtab
  theCIDValue = theCFtab.ReturnValue(theIDField, rec)
  theCTargetValue = theCFtab.ReturnValue(theTargetField, rec)
  theCList.Add(theCIDValue.asString + " --- " + theCTargetValue )
End

theChoice = MsgBox.ChoiceAsString (theCList, "Please select Casualty to update position", "RescueVIEW -
Casualty Postion update")

if (theChoice = Nil) then
```

```
  Exit
End

SearchStr = theChoice.Extract(0)
theQuery = "[ID] = " + SearchStr
theCFTab.Query(theQuery, theCBitmap, #VTAB_SELTYPE_NEW)
theCFTab.UpdateSelection

theCount = theCBitmap.Count

For each uprec in theCBitmap
  theXValue = theCFtab.ReturnValue(theXField, uprec)
  theYValue = theCFtab.ReturnValue(theYField, uprec)
  theTypeValue = theCFtab.ReturnValue(theTypeField, uprec)
  theTargetValue = theCFtab.ReturnValue(theTargetField, uprec)
  theCalledValue = theCFtab.ReturnValue(theCalledField, uprec)
  theTargetValue = theCFtab.ReturnValue(theTargetField, uprec)
  theStatusValue = theCFtab.ReturnValue(theStatusField, uprec)
  theNoteValue = theCFtab.ReturnValue(theNoteField, uprec)
End

theXDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theXValue})
theYDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theYValue})

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_UPDATE_CASUALTY_POSITION_DIALOG")

theDialog.GetDialog.FindByName("CtxtDegreeS").SetText(theYDMS.Extract(0))
theDialog.GetDialog.FindByName("CtxtDegreeW").SetText(theXDMS.Extract(0))
theDialog.GetDialog.FindByName("CtxtMinuteS").SetText(theYDMS.Extract(1))
theDialog.GetDialog.FindByName("CtxtMinuteW").SetText(theXDMS.Extract(1))

theDialog.GetDialog.FindByName("CtxtSecondS").SetText(theYDMS.Extract(2))
theDialog.GetDialog.FindByName("CtxtSecondW").SetText(theXDMS.Extract(2))

theDialog.GetDialog.FindByName("cTxtTarget").SetText(thetargetValue)

theDialog.GetDialog.FindByName("cboxNote").SetText(theNoteValue)


'********************** UPDATE DD FROM DMS VALUES ***************

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
```

```
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)


theFileName = (System.GetEnvVar("RESCUEVIEW") + "\Data\CallType.dbf")
theCallVtab = Vtab.Make(theFileName.asFileName, FALSE, FALSE)
theCapture = theCallVTab.findField("Type")

theComboBox = (theDialog.GetDialog).FindByName("cBoxType")
theComboBox.DefineFromVTab (theCallVTab, theCapture, False)
theComboBox.FindByValue (theTypeValue)
theComboBox.SetCurrentValue (theTypeValue)
theComboBox.SelectCurrent

theFileName = (System.GetEnvVar("RESCUEVIEW") + "\Data\CStatus.dbf")
theCallVtab = Vtab.Make(theFileName.asFileName, FALSE, FALSE)
theStatus = theCallVTab.findField("Status")

theComboBox = (theDialog.GetDialog).FindByName("cBoxStatus")
theComboBox.DefineFromVTab (theCallVTab, theStatus, False)
theComboBox.FindByValue (theStatusValue)
theComboBox.SetCurrentValue (theStatusValue)
theComboBox.SelectCurrent


theDialog.getDialog.Open
```

16. A_RESCUEVIEW_CASUALTY_POSITION_UPDATE_SCREEN

```
'******
'****** CALLED BY:  A_RESCUEVIEW_CONTROL_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theView = av.GetActiveDoc

theCTheme = theView.FindTheme("Casualty")

if (theCTheme = nil) then
  MsgBox.Error("No Casualty position has been logged","")
  EXIT
End

If (theCTheme.GetName <> "Casualty") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

End

theCFtab = theCTheme.GetFTab

theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theXField = theCFtab.FindField("X")
theYField = theCFtab.FindField("Y")
theTypeField = theCFtab.FindField("Type")
theTargetField = theCFtab.FindField("Target")
theNoteField = theCFtab.FindField("Note")


theCBitmap = TheCFtab.GetSelection

theCList = {}

for each rec in theCFtab
  theCIDValue = theCFtab.ReturnValue(theIDField, rec)
  theCTargetValue = theCFtab.ReturnValue(theTargetField, rec)
  theCList.Add(theCIDValue.asString + " --- "+ theCTargetValue )
End

theChoice = MsgBox.ChoiceAsString (theCList, "Please select Casualty to update position", "RescueVIEW -
Casualty Postion update")

if (theChoice = Nil) then
  Exit
End

SearchStr = theChoice.Extract(0)
```

```
theQuery = "[ID] = " + SearchStr
theCFTab.Query(theQuery, theCBitmap, #VTAB_SELTYPE_NEW)
theCFTab.UpdateSelection

theCount = theCBitmap.Count

SELF.SetTag("Operate")

 thePoint = theView.GetDisplay.ReturnUserPoint
 theXValue = thePoint.GetX
 theYValue = thePoint.GetY

 theCFtab.SetEditable(TRUE)

 If (theCFtab.IsEditable) then

  for each uprec in theCBitmap
   theCFtab.SetValue(theShapeField, uprec, thePoint)
   theCFtab.SetValue(theXField, uprec, theXValue)
   theCFtab.SetValue(theYField, uprec, theYValue)
  End

 End

 theCFtab.SetEditable(False)

 theCFtab.Flush

 if (theXValue > 0) then
  theXDegree = theXValue.asString ++ "EAST"
 Else
  theXDegree = theXValue.asString ++ "WEST"
 End

 if (theYValue > 0) then
  theYDegree = theYValue.asString ++ "NORTH"
 Else
  theYDegree = theYValue.asString ++ "SOUTH"
 End

For each rec in  theCBitmap
 theTargetValue = theCFtab.ReturnValue(theTargetField, rec)
End

av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theXValue, theYValue} )

'*****************************************************************************************************
'************************ ADD LOG FILE ENTRY ********************************************************
'*****************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*****************************************************************************************************

 theType = "UPDATE CASUALTY"
 theMessage = theTargetValue + "Position Update to " + theXDegree + " and " + theYDegree

 av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************************
```

```
'************************ END LOG FILE ENTRY ***************************************************************
'***********************************************************************************************************

theCBitmap.ClearAll
theCTheme.Invalidate (TRUE)
```

17. A_RESCUEVIEW_CLEAR_VIEW_GRAPHICS

```
'******
'****** CALLED BY:  CALLED BY VIEW BUTTON
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*************************************************************************

av.run("Graphic.SelectAll","")

av.run("View.DeleteGraphics","")
```

18. A_RESCUEVIEW_CLOSE_DIALOG

```
'******
'****** CALLED BY:  A_RESCUEVIEW_CONTROL_DIALOG
'******
'****** CALLS MADE: NONE
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*************************************************************************

SELF.GetDialog.Close
```

19. A_RESCUEVIEW_CONVERT_DD_TO_DMS

```
'******
'****** CALLED BY:  A_RESCUEVIEW_CASUALTY_POSITION_UPDATE_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: ESRI Sample Script - Modified by Kobus Meyer
'******          Modified as part of Master Thesis - GIS to the Rescue
'*************************************************************************

 theValue = Self.Get(0)

' get the degrees
d = theValue.asString.asTokens(".").Get(0)

dnum = d.asNumber.Abs
theValue = theValue.Abs

' convert decimal degrees to decimal seconds
ds = theValue * 3600
if (ds = 0) then
  dm = "00"
  dss = "00"
else

  ' get the minutes
  dm = (ds - (dnum * 3600)) / 60
  dm = dm.Truncate

  ' get the seconds
  dss = ds - (dnum * 3600) - (dm * 60)

  dss = dss.Truncate
  if (dm = 0) then
    dm = "00"
  end
  if (dss = 0) then
    dss = "00"
  end
end

dd = (d.asString ++ dm.asString ++ dss.asString)


Return(dd)
```

20. A_RESCUEVIEW_COURSE_TO_STEER

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE: A_RESCUEVIEW_ADD_COMPASS_ROSE
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

SELF.GetDialog.Close

theView = av.GetActiveDoc


thePath = System.GetEnvVar ("RescueView")

theVarianceFile = thePath + "\Data\Variance.odb"

if (theVarianceFile = nil) then
  MsgBox.Error("VARIANCE NOT SET." +NL+  "Go to Control box and set Variance to continue","")
  EXIT
End

theCTheme = theView.FindTheme("RESOURCE")

if (theCTheme = nil) then
  MsgBox.Error("No RESOURCE position has been logged","")
  EXIT
End

theCFtab = theCTheme.GetFTab

theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theCallsignField = theCFtab.FindField("Callsign")

theCBitmap = TheCFtab.GetSelection

theCList = {}

for each rec in theCFtab
  theCIDValue = theCFtab.ReturnValue(theIDField, rec)
  theCCallsignValue = theCFtab.ReturnValue(theCallsignField, rec)
  theCList.Add(theCIDValue.asString + " --- " + theCCallsignValue )
End

theChoice = MsgBox.ChoiceAsString (theCList, "Please select RESOURCE to steer from", "RescueVIEW -
RESOURCE Course to Steer")

if (theChoice = Nil) then
  Exit
End

SearchStr = theChoice.Extract(0)
```

```
theQuery = "[ID] = " + SearchStr
theCFTab.Query(theQuery, theCBitmap, #VTAB_SELTYPE_NEW)
theCFTab.UpdateSelection

theCount = theCBitmap.Count

For each uprec in theCBitmap
  theShape = theCFtab.ReturnValue(theShapeField, uprec)
  theXValue1 = theShape.GetX
  theYValue1 = theShape.GetY
End

theCBitmap.ClearAll

'************************************** GET TARGET CASUALTY ***************************************************************

theCTheme = theView.FindTheme("Casualty")

if (theCTheme = nil) then
  MsgBox.Error("No Casualty position has been logged","")
  EXIT
End

If (theCTheme.GetName <> "Casualty") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

End

theCFtab = theCTheme.GetFTab

theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theTargetField = theCFtab.FindField("Target")

theCBitmap = TheCFtab.GetSelection

theCList = {}

for each rec in theCFtab
  theCIDValue = theCFtab.ReturnValue(theIDField, rec)
  theCTargetValue = theCFtab.ReturnValue(theTargetField, rec)
  theCList.Add(theCIDValue.asString + " --- " + theCTargetValue )
End

theChoice = MsgBox.ChoiceAsString (theCList, "Please select Casualty to steer to", "RescueVIEW - Casualty
Course to Steer")

if (theChoice = Nil) then
  Exit
End

SearchStr = theChoice.Extract(0)
theQuery = "[ID] = " + SearchStr
theCFTab.Query(theQuery, theCBitmap, #VTAB_SELTYPE_NEW)
theCFTab.UpdateSelection

theCount = theCBitmap.Count
```

```
For each uprec in theCBitmap
  theShape = theCFtab.ReturnValue(theShapeField, uprec)
  theXValue2 = theShape.GetX
  theYValue2 = theShape.GetY
End

theCBitmap.ClearAll

point1 = theXValue1@theYValue1
point2 = theXValue2@theYValue2

'*****************************************************************************************************************

theBearing = av.run("A_RESCUEVIEW_DRIFT_BEARING",{point1 ,point2 })

if (theBearing = nil) then
  MsgBox.Error("No valid Bearing could be determined from input Coordinates","")
  EXIT
End

'*********************************************************************************
'************** DETERMINE NEW POSITION BASED ON TIME, SPEED AND DIRECTION ***************
'*********************************************************************************

theCourseLine = line.Make(point1 ,point2)

av.Run("A_RESCUEVIEW_ADD_COMPASS_ROSE", { theView, point1} )

'*** CONVERT DISTANCE FROM NAUTICAL MILES TO DECIMAL DEGREES

theLostDistance = Units.ConvertDecimalDegrees (Point1, Point2, #UNITS_LINEAR_NAUTICALMILES)


'Creates GraphicShapes and Symbols for a View

theGraphics = theView.GetGraphics

' Create a magenta line running diagonally from LL to UR....

gLine = GraphicShape.Make(theCourseLine)

theSymbol = gLine.Getsymbol
thesymbol.SetSize(2)
theSymbol.SetColor(Color.GetMagenta)

theGraphics.Add(gLine)

'**** find rotation coefficients

distance =(((point2.GetX - point1.GetX)^2) + ((point2.GetY - point1.GetY)^2)).sqrt
theCos = (point2.GetX - point1.GetX) / distance
theSin = -(point2.GetY - point1.GetY) / distance

'************** Create arrowhead polygon ******************

M = 0.0030

PointList = List.Make
thePoint = point2
PointList.Add(thePoint)
```



```
X0 =  point2.GetX
Y0 =  point2.GetY
X = (-0.25 * theCos * M) + (0.5 * theSin * M) + X0
Y = (0.25 * theSin * M) + (0.5 * theCos * M) + Y0
thePoint = Point.Make(X,Y)
PointList.Add(thePoint)
X = (0.75 * theCos * M) + X0
Y = (-0.75 * theSin * M) + Y0
thePoint = Point.Make(X,Y)
PointList.Add(thePoint)
X = (-0.25 * theCos * M) + (-0.5 * theSin * M) + X0
Y = (0.25 * theSin * M) + (-0.5 * theCos * M) + Y0
thePoint = Point.Make(X,Y)
PointList.Add(thePoint)
thePoint = point2
PointList.Add(thePoint)
theList = List.Make
theList.Add(PointList)
Arrow = Polygon.Make(theList)


'**** draw arrowhead, set symbol, and group with line

Arrow_G = GraphicShape.Make(Arrow)
Arrow_G.GetSymbol.SetStyle(#RASTERFILL_STYLE_SOLID)
Arrow_G.GetSymbol.SetColor(Color.GetMagenta)
Arrow_G.GetSymbol.SetOutlined(FALSE)
Arrow_G.SetVisible(TRUE)
'Arrow_G.SetSelected(TRUE)
theGraphics.Add(Arrow_G)
'theGraphics.GroupSelected
'Arrow_G.Unselect

' Create a red point at the center of the display...
gPoint = GraphicShape.Make(Point2)

theSymbol = gPoint.GetSymbol
theSymbol.SetColor(Color.GetRed)
theSymbol.SetSize(16)

theGraphics.Add(gPoint)

'*********  CLOSE DIALOG BOX TO DISPLAY LOCATION ON MAP

theView.Invalidate
'theDialog.Close

'***************** SHOW DRIFT INFORMATION **********************

VarODB = ODB.Open(theVarianceFile.AsFileName)

theVar = VarODB.Get(0)
theEast = VarODB.Get(1)


if (theEast = "E") then
  theCTS = theBearing + theVar
Else
  theCTS = theBearing - theVar
End
```

```
'theCTS = theBearing

if (theCTS > 360) then
  theCTS = theCTS - 360
End

if (theBearing > 360) then
  theBearing = theBearing - 360
End

theCTS = theCTS.Round
theBearing = theBearing.Round
'theSpeed = theSpeed.SetFormat("d.dd")
theLostDistance = theLostDistance.SetFormat("d.dd")

theBDialog = av.GetProject.FindDoc("A_RESCUEVIEW_BEARING_INFO_DIALOG").GetDialog

theBDialog.FindByName("tBox_Drift_CTS").SetText(theCTS.asString)
theBDialog.FindByName("tBox_Drift_Bearing").SetText(theBearing.asString)
theBDialog.FindByName("tBox_Drift_Speed").SetText("N/A")
theBDialog.FindByName("tBox_Drift_Distance").SetText(theLostDistance.asString)

theBDialog.Open
```

21. A_RESCUEVIEW_CREATE_LOG_UPDATE

```
'******
'****** CALLED BY:  ALL RESOURCE AND CASUALTY POSITIONAL UPDATE SCRIPTS
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theTypeEntry = SELF.Get(0)  ' Receive type of Log Message
theXEntry = SELF.Get(1)  ' Receive X value if present otherwise "nil"
theYEntry = SELF.Get(2)  ' Receive Y value if present otherwise "nil"
theMemoEntry = SELF.Get(3)  ' Receive message to populate log file

theIDField = _theLogVtab.FindField("ID")
theDateField = _theLogVtab.FindField("Date")
theTypeField = _theLogVtab.FindField("Type")
theXField = _theLogVtab.FindField("X")
theYField = _theLogVtab.FindField("Y")
theMemoField = _theLogVtab.FindField("Memo")

theTime = (Date.Now).SetFormat("dd MMMM yyyy hhhh m s ")
'theTime = ( (Date.Now).SetFormat("dd MMMM yyyy hhhh\Hm\:s ") ).asString
'theTime = Date.Now

rec = _theLogVtab.AddRecord

_theLogVtab.SetValue(theIDField, rec, rec)
_theLogVtab.SetValue(theDateField, rec, theTime)
_theLogVtab.SetValue(theTypeField, rec, theTypeEntry)
_theLogVtab.SetValue(theXField, rec, theXEntry)
_theLogVtab.SetValue(theYField, rec, theYEntry)
_theLogVtab.SetValue(theMemoField, rec, theMemoEntry)

RETURN nil
```

22. A_RESCUEVIEW_CREATE_OPERATION

```
'******
'****** CALLED BY:  A_RESCUEVIEW_CREATE_OPERATION (Menu Option - Create New Operation)
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

av.ClearGlobals
Date.SetDefFormat ("dd MMMM yyyy /at hhhh/Hm/m/:s")

theView = av.GetActiveDoc
theView.SetCoordsDMS (TRUE)

theMessage = "RescueVIEW - Operation Started on "+((Date.Now).SetFormat("dd MMMM yyyy \at hhh\Hm\:s
")).asString

av.SetName( theMessage )

thePath = System.GetEnvVar ("RescueView")

theDate = ((Date.Now).SetFormat("yyyy_MM_dd_hh_m")).asString

StrDir = thePath + "\Operation\" + theDate

strSecAtt = ""
dllName = FileName.FindInSystemSearchPath("kernel32.dll")
if (dllName = NIL) then
  MsgBox.Error("Could not locate the file 'kernel32.dll' in your system directory","Error in creating DIrectory")
  Return False
end
dllKernel = DLL.Make(dllName)
MkDir = DLLProc.Make(dllKernel,"CreateDirectoryA",#DLLPROC_TYPE_INT32,
            {#DLLPROC_TYPE_STR,#DLLPROC_TYPE_VOID})
strDir = strDir.Substitute("/","\")
lstDir = strDir.AsTokens("\")
strPart = lstDir.Get(0)
for each i in 1..(lstDir.Count - 1)
  strPart = strPart+"\"+lstDir.Get(i)
  if (strPart.AsFileName.IsDir.Not) then
    result = MkDir.Call({strPart,strSecAtt})
  else
    result = 1
  end
end

_theOperationPath = StrDir

av.GetActiveGUI.GetMenuBar.FindByScript ("A_RESCUEVIEW_STOP_OPERATION").SetEnabled(True)

SELF.SetEnabled(False)

   av.GetActiveGUI.GetButtonBar.FindByScript ("A_RESCUEVIEW_OPEN_CONTROLS").SetEnabled(TRUE)
```

```
Date.SetDefFormat("dd MMMM yyyy hhh\H\m\:s")

theChoiceValue = Nil
theChoiceList = {"EXERCISE", "RESCUE"}

While (theChoiceValue = Nil)
 theChoiceValue = MsgBox.ListasString(theChoiceList, "Please Select the type of operation","RescueView -
Search & Rescue Dispatch")
End

'*****************************************************************************************************
'********************************** CREATE LOG FILE ***************************************************
'*****************************************************************************************************

theLogDirName = StrDir + "\" + "INFO"

_theLogVtab = VTab.MakeNew (theLogDirName.asFileName, INFO)

if ( _theLogVtab = Nil ) then

 MsgBox.Error("Could not create logfile"+NL+"File Write permission denied","")
 EXIT

End

theIDField = Field.Make("ID", #FIELD_BYTE , 4, 0)
theDateField = Field.Make("Date", #FIELD_ISODATETIME , 25, 0)
theTypeField = Field.Make("Type", #FIELD_CHAR , 30, 0)
theXField = Field.Make("X", #FIELD_DOUBLE , 16, 4)
theYField = Field.Make("Y", #FIELD_DOUBLE , 16, 4)
theMemoField = Field.Make("Memo", #FIELD_CHAR , 250, 0)

_theLogVtab.AddFields({theIDField, theDateField, theTypeField, theXField, theYField, theMemofield})

'*****************************************************************************************************
'************************ ADD LOG FILE ENTRY ***********************************************************
'*****************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -NEW RESOURCE, -UPDATE RESOURCE
'*****************************************************************************************************

theType = "SYSTEM"
theXValue = Nil
theYValue = Nil

av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

theMessage = "Operational Information saved at: " + StrDir

av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************************

theTypeEntry = "SYSTEM"
theXEntry = Nil
theYEntry = Nil
theMessage = "OPERATION TYPE : " + theChoiceValue

av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})
```

```
'*************************************************************************************************

if (MsgBox.YESNO("Would you like to add details of Exercise / Rescue now","RescueView - Search & Rescue
Dispatch",TRUE)) then

   theInput = MsgBox.Input("Please enter details","RescueView - Search & Rescue Dispatch","")

   if ( theInput <> nil ) then
     theTypeEntry = "OPERATION"
     theXEntry = Nil
     theYEntry = Nil
     theMessage = "DETIALS : " + theInput

     av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})
   End

 End

'*************************************************************************************************
'********************** END LOG FILE ENTRY ***************************************************
'*************************************************************************************************

theView.GetDisplay.SetUnits(#UNITS_LINEAR_DEGREES)
theView.GetDisplay.SetDistanceUnits(#UNITS_LINEAR_NAUTICALMILES)

theCasualtyTheme = theView.FindTheme("Casualty")

if (theCasualtyTheme <> nil) then
  theView.DeleteTheme(theCasualtyTheme)
End

av.PurgeObjects
```

23. A_RESCUEVIEW_DELETED_TMP_GRAPHICS

```
'******
'****** CALLED BY:  A_RESCUEVIEW_POINT_DISPLAY
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*************************************************************************************

theView = av.GetActiveDoc

theGraph = SELF.Get(0)

theGraphicList = theView.GetGraphics

theGraphicList.RemoveGraphic (theGraph)
```

24. A_RESCUEVIEW_DRIFT_DIALOG_UPDATE

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = SELF.GetDialog

theName = SELF.GetName

theView = av.GetActiveDoc

if (theName = "LBUTStartTime") then
  theDate = Date.Now

  theDialog.FindByName("txtStartTime").SetLabel((theDate.SetFormat("hhhh\Hm\:s")).asString)
  theDialog.FindByName("txtStartTime").SetObjectTag(theDate)
  theDialog.FindByName("but_Submit").SetEnabled(FALSE)
  '*************************************************************************************************
  '********************** ADD LOG FILE ENTRY ***************************************************
  '*************************************************************************************************
  '*** ENTRY TYPES
  '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
  '*************************************************************************************************

    theType = "SYSTEM"
    theXValue = Nil
    theYValue = Nil
    theMessage = "Drift Start Time captured"

    av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

  '*************************************************************************************************
  '********************** END LOG FILE ENTRY ***************************************************
  '*************************************************************************************************

ElseIf (theName = "LBUTEndTime") then
  theDate = Date.Now
  theDialog.FindByName("txtEndTime").SetLabel((theDate.SetFormat("hhhh\Hm\:s")).asString)

  theStartDate = theDialog.FindByName("txtStartTime").GetObjectTag

  theDiffDate = (theDate - theStartDate).AsMinutes

  theDialog.FindByName("txtDuration").SetLabel(theDiffDate.asString ++ "Minutes")
  theDialog.FindByName("txtDuration").SetObjectTag(theDiffdate)
  theDialog.FindByName("but_Submit").SetEnabled(TRUE)

  '*************************************************************************************************
  '********************** ADD LOG FILE ENTRY ***************************************************
  '*************************************************************************************************
```

```
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*****************************************************************************************

  theType = "USER"
  theXValue = Nil
  theYValue = Nil
  theMessage = "Drifter End Time captured with duration of" ++ theDiffDate.asString ++ "minutes"

  av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************
'********************** END LOG FILE ENTRY ***********************************************
'*****************************************************************************************

End

'************************** UPDATE START DRIFT POSITION ************************************

if (theName = "StxtMinuteE") then
 theMin = (theDialog.FindByName("StxtMinuteE").GetText).asNumber / 60

 if ( (((theDialog.FindByName("StxtSecondE").GetText).asNumber) > 0) and
(((theDialog.FindByName("StxtSecondE").GetText).asNumber) < 60) ) then
  theSec = (theDialog.FindByName("StxtSecondE").GetText).asNumber / 3600
 Else
  theSec = 0
 End
 theDecMin = theMin + theSec

 theDialog.FindByName("StxtDMinuteE").SetText(theDecMin.asString)

Elseif (theName = "StxtMinuteS") then
 theMin = (theDialog.FindByName("StxtMinuteS").GetText).asNumber / 60
 theSec = (theDialog.FindByName("StxtSecondS").GetText).asNumber / 3600
 theDecMin = theMin + theSec

 theDialog.FindByName("StxtDMinuteS").SetText(theDecMin.asString)


End

if (theName = "StxtSecondE") then
 theMin = (theDialog.FindByName("StxtMinuteE").GetText).asNumber / 60
 theSec = (theDialog.FindByName("StxtSecondE").GetText).asNumber / 3600
 theDecMin = theMin + theSec

 theDialog.FindByName("StxtDMinuteE").SetText(theDecMin.asString)

Elseif (theName = "StxtSecondS") then
 theMin = (theDialog.FindByName("StxtMinuteS").GetText).asNumber / 60
 theSec = (theDialog.FindByName("StxtSecondS").GetText).asNumber / 3600
 theDecMin = theMin + theSec

 theDialog.FindByName("StxtDMinuteS").SetText(theDecMin.asString)

End

'************************** UPDATE END DRIFT POSITION **************************************
```

```
if (theName = "EtxtMinuteE") then
  theMin = (theDialog.FindByName("EtxtMinuteE").GetText).asNumber / 60

  if        (        (((theDialog.FindByName("EtxtSecondE").GetText).asNumber)        >        0)        and
(((theDialog.FindByName("EtxtSecondE").GetText).asNumber) < 60) ) then
    theSec = (theDialog.FindByName("EtxtSecondE").GetText).asNumber / 3600
  Else
    theSec = 0
  End
  theDecMin = theMin + theSec

  theDialog.FindByName("EtxtDMinuteE").SetText(theDecMin.asString)

Elseif (theName = "EtxtMinuteS") then
  theMin = (theDialog.FindByName("EtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.FindByName("EtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("EtxtDMinuteS").SetText(theDecMin.asString)


End

if (theName = "EtxtSecondE") then
  theMin = (theDialog.FindByName("EtxtMinuteE").GetText).asNumber / 60
  theSec = (theDialog.FindByName("EtxtSecondE").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("EtxtDMinuteE").SetText(theDecMin.asString)

Elseif (theName = "EtxtSecondS") then
  theMin = (theDialog.FindByName("EtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.FindByName("EtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("EtxtDMinuteS").SetText(theDecMin.asString)

End

'************************** UPDATE CASUALTY DRIFT POSITION ***********************************************

if (theName = "LtxtMinuteE") then
  theMin = (theDialog.FindByName("LtxtMinuteE").GetText).asNumber

  if        (        (((theDialog.FindByName("LtxtSecondE").GetText).asNumber)        >        0)        and
(((theDialog.FindByName("LtxtSecondE").GetText).asNumber) < 60) ) then
    theSec = (theDialog.FindByName("LtxtSecondE").GetText).asNumber / 3600
  Else
    theSec = 0
  End
  theDecMin = theMin + theSec

  theDialog.FindByName("LtxtDMinuteE").SetText(theDecMin.asString)

 Elseif (theName = "LtxtMinuteS") then
  theMin = (theDialog.FindByName("LtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.FindByName("LtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec
```

```
  theDialog.FindByName("LtxtDMinuteS").SetText(theDecMin.asString)


End

if (theName = "LtxtSecondE") then
  theMin = (theDialog.FindByName("LtxtMinuteE").GetText).asNumber / 60
  theSec = (theDialog.FindByName("LtxtSecondE").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("LtxtDMinuteE").SetText(theDecMin.asString)

Elseif (theName = "LtxtSecondS") then
  theMin = (theDialog.FindByName("LtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.FindByName("LtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

  theDialog.FindByName("LtxtDMinuteS").SetText(theDecMin.asString)

End
```

25. A_RESCUEVIEW_DRIFT_DISTANCE

```
'******
'****** CALLED BY:  A_RESCUEVIEW_DRIFT_POSITION_DIALOG
'******
'****** CALLS MADE: A_RESCUEVIEW_DRIFT_BEARING
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

thePath = System.GetEnvVar ("RescueView")

theVarianceFile = thePath + "\Data\Variance.odb"

if (theVarianceFile = nil) then
  MsgBox.Error("VARIANCE NOT SET." +NL+  "Go to Control box and set Variance to continue","")
  EXIT
End

theView = av.GetActiveDoc

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_DRIFT_POSITION_DIALOG").GetDialog

theDialog.Close


theSDEGX = (theDialog.FindByName("StxtDegreeS").GetText).asNumber
theSMINX = (theDialog.FindByName("StxtDMinuteS").GetText).asNumber

theSDEGY = (theDialog.FindByName("StxtDegreeE").GetText).asNumber
theSMINY = (theDialog.FindByName("StxtDMinuteE").GetText).asNumber

theStartXValue = theSDEGX + theSMINX
theStartYValue = theSDEGY - theSMINY

'*************************************************************************************************
'************************ ADD LOG FILE ENTRY ****************************************************
'*************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*************************************************************************************************

  theType = "START DRIFT CAPTURE"
  theXValue = theStartXValue
  theYValue = theStartYValue
  theMessage = "Start drift coordinate captured as" ++ theStartXValue.asString ++ "and" ++
theStartYValue.asString

  av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*************************************************************************************************
'************************ END LOG FILE ENTRY ****************************************************
'*************************************************************************************************
```

```
theEDEGX = (theDialog.FindByName("EtxtDegreeS").GetText).asNumber
theEMINX = (theDialog.FindByName("EtxtDMinuteS").GetText).asNumber

theEDEGY = (theDialog.FindByName("EtxtDegreeE").GetText).asNumber
theEMINY = (theDialog.FindByName("EtxtDMinuteE").GetText).asNumber

theEndXValue = theEDEGX + theEMINX
theEndYValue = theEDEGY - theEMINY

'*****************************************************************************************************
'************************ ADD LOG FILE ENTRY ********************************************************
'*****************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*****************************************************************************************************

 theType = "END DRIFT CAPTURE"
 theXValue = theEndXValue
 theYValue = theEndYValue
 theMessage = "End drift coordinate captured as" ++ theEndXValue.asString ++ "and" ++ theEndYValue.asString

 av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************************
'************************ END LOG FILE ENTRY ********************************************************
'*****************************************************************************************************


theLDEGX = (theDialog.FindByName("LtxtDegreeS").GetText).asNumber
theLMINX = (theDialog.FindByName("LtxtDMinuteS").GetText).asNumber

theLDEGY = (theDialog.FindByName("LtxtDegreeE").GetText).asNumber
theLMINY = (theDialog.FindByName("LtxtDMinuteE").GetText).asNumber

theLastXValue = theLDEGX + theLMINX
theLastYValue = theLDEGY - theLMINY

'*****************************************************************************************************
'************************ ADD LOG FILE ENTRY ********************************************************
'*****************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*****************************************************************************************************

 theType = "CASUALTY LAST KNOWN POSITION"
 theXValue = theLastXValue
 theYValue = theLastYValue
 theMessage = "Swimmer end coordinate captured as" ++ theLastXValue.asString ++ "and" ++
theLastYValue.asString

 av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************************
'************************ END LOG FILE ENTRY ********************************************************
'*****************************************************************************************************


thePoint1 = Point.Make(theStartXValue, theStartYValue)
```

```
thePoint2 = Point.Make(theEndXValue, theEndYValue)
thePoint3 = Point.Make(theLastXValue, theLastYValue)

'theNauticalDistance = Units.ConvertDecimalDegrees (thePoint1, thePoint2, #UNITS_LINEAR_NAUTICALMILES)
'theLostDistance = Units.ConvertDecimalDegrees (thePoint1, thePoint2, #UNITS_LINEAR_NAUTICALMILES)

theLine = Line.Make(thePoint1 , thePoint2)
theDistance = theLine.ReturnLength

theTime =    theDialog.FindByName("txtDuration").GetObjectTag / 60  '**** Convert to hours

if (theTime = nil) then
  MsgBox.Warning("User Cancelled","")
  EXIT
End

theBearing = av.run("A_RESCUEVIEW_DRIFT_BEARING",{thePoint1 , thePoint2})

if (theBearing = nil) then
  MsgBox.Error("No valid Bearing could be determined from input Coordinates","")
  EXIT
End

'*****************************************************************************************
'************************** SPEED = DISTANCE / TIME ***********************************
'*****************************************************************************************

'theSpeed = theNauticalDistance / theTime
theSpeed = theDistance / theTime

'MsgBox.Info("Current Drift Speed is" ++ theSpeed.asString ++ "knots", "")

'*****************************************************************************************
'************************** DISTANCE = SPEED * TIME ***********************************
'*****************************************************************************************


theLostTime = theDialog.FindByName("txtBoxLastDuration").GetText
theLostTime = (theLostTime.asNumber /60 )

theLostDistance = theSpeed * theLostTime

'MsgBox.Info("TOTAL DRIFT DISTANCE IS : " + theLostDistance.asString + "nm","")


'*****************************************************************************************
'*************** DETERMINE NEW POSITION BASED ON TIME, SPEED AND DIRECTION ***************
'*****************************************************************************************

'theLastX = thePoint3.GetX
'theLastY = thePoint3.GetY

'*** CONVERT DISTANCE FROM NAUTICAL MILES TO DECIMAL DEGREES

theProjectDistance = Units.Convert (theLostDistance, #UNITS_LINEAR_NAUTICALMILES,
#UNITS_LINEAR_DEGREES)

newx = ( ( ( theBearing.AsRadians ).Cos ) * theProjectDistance ) + theLastXValue
newy = ( ( ( theBearing.AsRadians ).Sin ) * theProjectDistance ) + theLastYValue
```

```
'*****************************************************************************************
'************************ ADD LOG FILE ENTRY ********************************************
'*****************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*****************************************************************************************

 theType = "CASUALTY PROJECTED POSITION"
 theXValue = newx
 theYValue = newy
 theMessage = "Casualty new projected coordinate at" ++ newx.asString ++ "and" ++ newy.asString

 av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************
'************************ END LOG FILE ENTRY ********************************************
'*****************************************************************************************

theDisplay = theView.GetDisplay

theGraphics = theView.GetGraphics


'*****************************************************************************
'***************** CREATE LINE BETWEEN RESCUE SOURCE & CASUALTY ***************************
'*****************************************************************************

aLine = Line.Make(thePoint1, newx@newy)
gLine = GraphicShape.Make(aLine)

theSymbol = gLine.Getsymbol
thesymbol.SetSize(2)
theSymbol.SetColor(Color.GetRED)

theGraphics.Add(gLine)



'*****************************************************************************
'******* CREATE LINE BETWEEN CASUALTY LAST KNOWN & CASUALTY DRIFT ESTIMATE **************
'*****************************************************************************


aLine = Line.Make(thePoint3, newx@newy)
gLine = GraphicShape.Make(aLine)

theSymbol = gLine.Getsymbol
thesymbol.SetSize(2)
theSymbol.SetColor(Color.GetMagenta)

theGraphics.Add(gLine)



'**** find rotation coefficients

point1 = thePoint3
point2 = newx@newy
```

```
distance =(((point2.GetX - point1.GetX)^2) + ((point2.GetY - point1.GetY)^2)).sqrt
theCos = (point2.GetX - point1.GetX) / distance
theSin = -(point2.GetY - point1.GetY) / distance

'************** Create arrowhead polygon ******************

M = 0.0010

PointList = List.Make
thePoint = point2
PointList.Add(thePoint)
X0 =  point2.GetX
Y0 =  point2.GetY
X = (-0.25 * theCos * M) + (0.5 * theSin * M) + X0
Y = (0.25 * theSin * M) + (0.5 * theCos * M) + Y0
thePoint = Point.Make(X,Y)
PointList.Add(thePoint)
X = (0.75 * theCos * M) + X0
Y = (-0.75 * theSin * M) + Y0
thePoint = Point.Make(X,Y)
PointList.Add(thePoint)
X = (-0.25 * theCos * M) + (-0.5 * theSin * M) + X0
Y = (0.25 * theSin * M) + (-0.5 * theCos * M) + Y0
thePoint = Point.Make(X,Y)
PointList.Add(thePoint)
thePoint = point2
PointList.Add(thePoint)
theList = List.Make
theList.Add(PointList)
Arrow = Polygon.Make(theList)


'**** draw arrowhead, set symbol, and group with line

Arrow_G = GraphicShape.Make(Arrow)
Arrow_G.GetSymbol.SetStyle(#RASTERFILL_STYLE_EMPTY)
Arrow_G.GetSymbol.SetColor(Color.GetMagenta)
Arrow_G.GetSymbol.SetOLWidth(2)
Arrow_G.GetSymbol.SetOLColor(Color.GetMagenta)
Arrow_G.GetSymbol.SetOutlined(TRUE)
Arrow_G.SetVisible(TRUE)
'Arrow_G.SetSelected(TRUE)
theGraphics.Add(Arrow_G)
'theGraphics.GroupSelected
'Arrow_G.Unselect

' Create a red point at the center of the display...
aPoint = Point.Make(newX, newY)
gPoint = GraphicShape.Make(aPoint)

theSymbol = gPoint.GetSymbol
theSymbol.SetColor(Color.GetRed)
theSymbol.SetSize(12)

theGraphics.Add(gPoint)
'gPoint.Unselect

'*********  CLOSE DIALOG BOX TO DISPLAY LOCATION ON MAP
```

```
theView.Invalidate
'theDialog.Close

'****************** SHOW DRIFT INFORMATION ************************

VarODB = ODB.Open(theVarianceFile.AsFileName)

theVar = VarODB.Get(0)
theEast = VarODB.Get(1)

theBearing = av.run("A_RESCUEVIEW_DRIFT_BEARING",{thePoint1 , Point2})

if (theEast = "E") then
  theCTS = theBearing - theVar
Else
  theCTS = theBearing + theVar
End


if (theCTS > 360) then
  theCTS = theCTS - 360
End

if (theBearing > 360) then
  theBearing = theBearing - 360
End

theCTS = theCTS.Round
theBearing = theBearing.Round
theSpeed = theSpeed.SetFormat("d.dd")
theLostDistance = theLostDistance.SetFormat("d.ddd")

'*****************************************************************************************************
'************************ ADD LOG FILE ENTRY ********************************************************
'*****************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*****************************************************************************************************

  theType = "DRIFTPOSITION"
  theXValue = Nil
  theYValue = Nil
  theMessage = "DRIFT CALCULATION - " + "COURSE TO STEER:" ++ theCTS.asString + "; BEARING:" ++
theBearing.asString + "; SPEED: " + theSpeed.asString + "; DRIFT DISTANCE:" ++ theLostDistance.asString

  av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************************
'************************ END LOG FILE ENTRY ********************************************************
'*****************************************************************************************************


'************************  DRAW  COMPASS  ROSE  AT  CURRENT  RESCUER  POSITION
**********************************

av.Run("A_RESCUEVIEW_ADD_COMPASS_ROSE", { theView, thePoint1} )
```

```
theBDialog = av.GetProject.FindDoc("A_RESCUEVIEW_BEARING_INFO_DIALOG").GetDialog

theBDialog.FindByName("tBox_Drift_CTS").SetText(theCTS.asString)
theBDialog.FindByName("tBox_Drift_Bearing").SetText(theBearing.asString)
theBDialog.FindByName("tBox_Drift_Speed").SetText(theSpeed.asString)
theBDialog.FindByName("tBox_Drift_Distance").SetText(theLostDistance.asString)

theBDialog.Open
```


26. A_RESCUEVIEW_DRIFT_POINT_TO_CASUALTY

```
'******
'****** CALLED BY:  A_RESCUEVIEW_DRIFT_POSITION_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

'tbut_GetScreen

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_DRIFT_POSITION_DIALOG")

theName = SELF.GetName

theView = av.GetActiveDoc

if (theName = "lbut_Last_Known") then

 theCTheme = theView.FindTheme("Casualty")

 if (theCTheme = nil) then
  MsgBox.Error("No Casualty position has been logged","")
  EXIT
 End

 If (theCTheme.GetName <> "Casualty") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

 End

 theCFtab = theCTheme.GetFTab

 theShapeField = theCFtab.FindField("Shape")
 theIDField = theCFtab.FindField("ID")
 theTargetField = theCFtab.FindField("Target")

 theCBitmap = TheCFtab.GetSelection

 theCList = {}

 for each rec in theCFtab
  theCIDValue = theCFtab.ReturnValue(theIDField, rec)
  theCTargetValue = theCFtab.ReturnValue(theTargetField, rec)
  theCList.Add(theCIDValue.asString + " --- " + theCTargetValue )
 End

 theChoice = MsgBox.ChoiceAsString (theCList, "Please select Casualty to steer to", "RescueVIEW - Casualty
Course to Steer")

 if (theChoice = Nil) then
  Exit
 End
```

```
  SearchStr = theChoice.Extract(0)
  theQuery = "[ID] = " + SearchStr
  theCFTab.Query(theQuery, theCBitmap, #VTAB_SELTYPE_NEW)
  theCFTab.UpdateSelection

  theCount = theCBitmap.Count

  For each uprec in theCBitmap
    thePoint = theCFtab.ReturnValue(theShapeField, uprec)
  End


  theCBitmap.ClearAll

Else

  thePoint = theView.GetDisplay.ReturnUserPoint
  theDialog.GetDialog.Open
End

theXValue = thePoint.GetX
theYValue = thePoint.GetY


theXDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theXValue})
theYDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theYValue})


theDialog.GetDialog.FindByName("LtxtDegreeE").SetText(theYDMS.Extract(0))
theDialog.GetDialog.FindByName("LtxtDegreeS").SetText(theXDMS.Extract(0))
theDialog.GetDialog.FindByName("LtxtMinuteE").SetText(theYDMS.Extract(1))
theDialog.GetDialog.FindByName("LtxtMinuteS").SetText(theXDMS.Extract(1))

theDialog.GetDialog.FindByName("LtxtSecondE").SetText(theYDMS.Extract(2))
theDialog.GetDialog.FindByName("LtxtSecondS").SetText(theXDMS.Extract(2))


'********************** UPDATE DD FROM DMS VALUES ****************

  theMin = (theDialog.GetDialog.FindByName("LtxtMinuteE").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("LtxtSecondE").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("LtxtDMinuteE").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("LtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("LtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("LtxtDMinuteS").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("LtxtMinuteE").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("LtxtSecondE").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("LtxtDMinuteE").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("LtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("LtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec
```

```
theDialog.GetDialog.FindByName("LtxtDMinuteS").SetText(theDecMin.asString)
```

27. A_RESCUEVIEW_DRIFT_POSITION_DIALOG_OPEN

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE: A_RESCUEVIEW_DRIFT_BEARING
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

SELF.GetDialog.Close

av.GetProject.FindDoc("A_RESCUEVIEW_DRIFT_POSITION_DIALOG").GetDialog.Open
```

28. A_RESCUEVIEW_GET_XY

```
'******
'****** CALLED BY:  A_RESCUEVIEW_CONTROL_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*************************************************************************

theView = av.GetActiveDoc

theUserPnt = TheView.GetDisplay.ReturnUserPoint

TheX = theUserPnt.GetX
TheY = theUserPnt.GetY

MsgBox.Info("Longitude = " + theX.asString + NL + "Latitude = " + theY.asString, "")
```

29. A_RESCUEVIEW_LOAD_WAYPOINTS

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE :  NONE
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

SELF.GetDialog.Close

thepattern = "*.dbf"
theLabel = "DBASE"

theFileName = FileDialog.Show( thePattern, theLabel, "Select Waypoint file")

theVtab = Vtab.Make(theFileName, FALSE, FALSE)

theFieldList = theVtab.GetFields

'msgBox.Info( (theFieldList.Count).asString, "")


theXfield = theVtab.findField("X")

If ( theXfield = Nil ) Then
  theXfield = theVtab.findField("Longitude")
  If ( theXfield = Nil ) Then
    theXField = MsgBox.ListasString(theFieldList, "Select field containing X values", "Waypoint loader")
    If ( theXfield = Nil ) Then
      MsgBox.Error("No valid X field found","")
      EXIT
    End
  End
End

theYfield = theVtab.findField("Y")

If ( theYfield = Nil ) Then
  theYfield = theVtab.findField("Latitude")
  If ( theYfield = Nil ) Then
    theYField = MsgBox.ListasString(theFieldList, "Select field containing Y values", "Waypoint loader")
    If ( theYfield = Nil ) Then
      MsgBox.Error("No valid Y field found","")
      EXIT
    End
  End
End


theView = av.GetActiveDoc

If (MsgBox.YesNo("Do you want to display an id/text with waypoint?","Waypoint loader",TRUE) ) then

    theNameField = MsgBox.ListasString(theFieldList, "Select field containing Id/Name values", "Waypoint loader")
```

```
    If ( theNamefield = Nil ) Then
      MsgBox.Error("No valid Y field found","")
      EXIT
    End
End

thexy = XYName.Make(theVtab, theXField, theYField)
theTheme = Theme.make(theXY)
theView.AddTheme(theTheme)

theTheme.SetName("WayPoints")

theScale = theView.ReturnScale


If ( theNamefield <> Nil ) Then

  theGraphics = theView.GetGraphics

  theBitmap = theVtab.GetSelection

  theBitmap.SetAll

  For each rec in theBitmap

    theTextG=GraphicText.Make(                (theVtab.ReturnValue(theNameField,rec)                ).asString,
theVtab.ReturnValue(theXField,rec)@theVtab.ReturnValue(theYField,rec) )
    theTextG.setobjecttag("mygText")

    thefont=Font.MakeStandard(#FONT_TIMEB)
    thefont.SetStyle ("Bold")
    theSymbol=TextSymbol.Make
    theSymbol.SetFont(thefont)
    theSymbol.SetColor(Color.GetBlack)
    theSymbol.SetSize(14)

    'theView.getDisplay.HookUpSymbol(theSymbol)

    theTextG.SetSymbol(theSymbol)
    theGraphics.Add(theTextG)

  End

  theBitmap.ClearAll

End

theLegend = theTheme.GetLegend
theLegendFile = (System.GetEnvVar("RESCUEVIEW") + "\Data\Waypoint.avl").asFilename
theLegend.Load(theLegendFile, #LEGEND_LOADTYPE_ALL)

theView.GetDisplay.ZoomToScale (theScale)
theTheme.SetVisible(TRUE)
thegraphics.Invalidate
```

30. A_RESCUEVIEW_LOG_FILE_EXPORT

```
'******
'****** CALLED BY: A_RESCUEVIEW_OPEN_CONTROLS-View.Button
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

defname = ( (Date.Now).SetFormat("ddMMyyyyhhhhm") ).asString + ".log"

theFileName = FileDialog.Put( Defname.asFileName, "*.log", "Enter file name and location for Export" )

if (theFileName = Nil) then
  MsgBox.Error("No valid file name entered" +NL+ "Export cancelled","")
  EXIT
End

theLineFile = LineFile.Make( theFilename, #FILE_PERM_WRITE )

if (theLinefile = Nil) then
  MsgBox.Error("File name could not be created.  Check permissions" +NL+ "Export cancelled","")
  EXIT
End

theIdField = _theLogVtab.FindField("ID")
theDateField = _theLogVtab.FindField("Date")
theTypeField = _theLogVtab.FindField("Type")
theXField = _theLogVtab.FindField("X")
theYField = _theLogVtab.FindField("Y")
theMemoField = _theLogVtab.FindField("Memo")

theBitMap = _theLogVtab.GetSelection
theBitmap.SetAll

for each rec in theBitMap

  theLineFile.WriteElt( "Log ID:" ++ _theLogVtab.ReturnValueString(theIdField, rec) +TAB+ "Date:" ++
_theLogVtab.ReturnValueString(theDateField, rec) )
  theLineFile.WriteElt( "Log Type Entry:" ++ _theLogVtab.ReturnValueString(theTypeField, rec) )
  theLineFile.WriteElt( "X-Coordinate:" ++ _theLogVtab.ReturnValueString(theXField, rec) +TAB+ "Y_Coordinate:"
++ _theLogVtab.ReturnValueString(theYField, rec) )
  theLineFile.WriteElt( "Log Entry:" +NL+ _theLogVtab.ReturnValueString(theMemoField, rec) )
  theLineFile.WriteElt( "**************************************************" )

End

  theLineFile.WriteElt( "" )
  theLineFile.WriteElt( "**************************************************" )
  theLineFile.WriteElt( "* * * * END OF LOG FILE - AN ENTRY's AFTER THIS MESSAGES WAS ADDED AFTER
CLOSE OF OPERATION * * *" )
  theLineFile.WriteElt( "**************************************************" )
  theLineFile.WriteElt( "" )
```

```
theLineFile.Close

MsgBox.Info("Log File exported successfully","LOG FILE EXPORT")
```

31. A_RESCUEVIEW_LOG_VIEWER_FIRST_RECORD

```
'******
'****** CALLED BY: A_RESCUEVIEW_OPEN_CONTROLS-View.Button
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_LOG_FILE_VIEWER_DIALOG")

theIdField = _theLogVtab.FindField("ID")
theDateField = _theLogVtab.FindField("Date")
theTypeField = _theLogVtab.FindField("Type")
theXField = _theLogVtab.FindField("X")
theYField = _theLogVtab.FindField("Y")
theMemoField = _theLogVtab.FindField("Memo")


rec = 0

theDialog.GetDialog.FindByName("txtRecord").Setlabel( _theLogVtab.ReturnValueString(theIdField, rec) )
theDialog.GetDialog.FindByName("txtDate").Setlabel( _theLogVtab.ReturnValueString(theDateField, rec) )
theDialog.GetDialog.FindByName("txtLogType").Setlabel( _theLogVtab.ReturnValueString(theTypeField, rec) )
theDialog.GetDialog.FindByName("txtXValue").Setlabel( _theLogVtab.ReturnValueString(theXField, rec) )
theDialog.GetDialog.FindByName("txtYValue").Setlabel( _theLogVtab.ReturnValueString(theYField, rec) )
theDialog.GetDialog.FindByName("tbox_Memo").SetText( _theLogVtab.ReturnValueString(theMemoField, rec) )

theDialog.getDialog.Open
```

32. A_RESCUEVIEW_LOG_VIEWER_GOTO_RECORD

```
'******
'****** CALLED BY: A_RESCUEVIEW_OPEN_CONTROLS-View.Button
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_LOG_FILE_VIEWER_DIALOG")

theIdField = _theLogVtab.FindField("ID")
theDateField = _theLogVtab.FindField("Date")
theTypeField = _theLogVtab.FindField("Type")
theXField = _theLogVtab.FindField("X")
theYField = _theLogVtab.FindField("Y")
theMemoField = _theLogVtab.FindField("Memo")


Currentrec = theDialog.GetDialog.FindByName("tbox_Goto").GetText

rec = Currentrec.asNumber

if (_theLogVtab.ReturnValueString(theIdField, rec) = nil) then
  MsgBox.Warning("No more records to view","LOG FILE VIEWER - END OF RECORDS")
  EXIT
End


theDialog.GetDialog.FindByName("txtRecord").Setlabel( _theLogVtab.ReturnValueString(theIdField, rec) )
theDialog.GetDialog.FindByName("txtDate").Setlabel( _theLogVtab.ReturnValueString(theDateField, rec) )
theDialog.GetDialog.FindByName("txtLogType").Setlabel( _theLogVtab.ReturnValueString(theTypeField, rec) )
theDialog.GetDialog.FindByName("txtXValue").Setlabel( _theLogVtab.ReturnValueString(theXField, rec) )
theDialog.GetDialog.FindByName("txtYValue").Setlabel( _theLogVtab.ReturnValueString(theYField, rec) )
theDialog.GetDialog.FindByName("tbox_Memo").SetText( _theLogVtab.ReturnValueString(theMemoField, rec) )

theDialog.getDialog.Open
```

33. A_RESCUEVIEW_LOG_VIEWER_LAST_RECORD

```
'******
'****** CALLED BY: A_RESCUEVIEW_OPEN_CONTROLS-View.Button
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_LOG_FILE_VIEWER_DIALOG")

theIdField = _theLogVtab.FindField("ID")
theDateField = _theLogVtab.FindField("Date")
theTypeField = _theLogVtab.FindField("Type")
theXField = _theLogVtab.FindField("X")
theYField = _theLogVtab.FindField("Y")
theMemoField = _theLogVtab.FindField("Memo")

theBitMap = _theLogVtab.GetSelection
theBitmap.SetAll

rec = theBitMap.count - 1

theDialog.GetDialog.FindByName("txtRecord").Setlabel( _theLogVtab.ReturnValueString(theIdField, rec) )
theDialog.GetDialog.FindByName("txtDate").Setlabel( _theLogVtab.ReturnValueString(theDateField, rec) )
theDialog.GetDialog.FindByName("txtLogType").Setlabel( _theLogVtab.ReturnValueString(theTypeField, rec) )
theDialog.GetDialog.FindByName("txtXValue").Setlabel( _theLogVtab.ReturnValueString(theXField, rec) )
theDialog.GetDialog.FindByName("txtYValue").Setlabel( _theLogVtab.ReturnValueString(theYField, rec) )
theDialog.GetDialog.FindByName("tbox_Memo").SetText( _theLogVtab.ReturnValueString(theMemoField, rec) )

theDialog.getDialog.Open
```

34. A_RESCUEVIEW_LOG_VIEWER_NEXT_RECORD

```
'******
'****** CALLED BY: A_RESCUEVIEW_OPEN_CONTROLS-View.Button
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_LOG_FILE_VIEWER_DIALOG")

theIdField = _theLogVtab.FindField("ID")
theDateField = _theLogVtab.FindField("Date")
theTypeField = _theLogVtab.FindField("Type")
theXField = _theLogVtab.FindField("X")
theYField = _theLogVtab.FindField("Y")
theMemoField = _theLogVtab.FindField("Memo")


Currentrec = theDialog.GetDialog.FindByName("txtRecord").GetLabel

rec = Currentrec.asNumber + 1

theBitMap = _theLogVtab.GetSelection
theBitmap.SetAll

if (rec = theBitMap.count) then
  MsgBox.Warning("No more records to view","LOG FILE VIEWER - END OF RECORDS")
  EXIT
End


theDialog.GetDialog.FindByName("txtRecord").Setlabel( _theLogVtab.ReturnValueString(theIdField, rec) )
theDialog.GetDialog.FindByName("txtDate").Setlabel( _theLogVtab.ReturnValueString(theDateField, rec) )
theDialog.GetDialog.FindByName("txtLogType").Setlabel( _theLogVtab.ReturnValueString(theTypeField, rec) )
theDialog.GetDialog.FindByName("txtXValue").Setlabel( _theLogVtab.ReturnValueString(theXField, rec) )
theDialog.GetDialog.FindByName("txtYValue").Setlabel( _theLogVtab.ReturnValueString(theYField, rec) )
theDialog.GetDialog.FindByName("tbox_Memo").SetText( _theLogVtab.ReturnValueString(theMemoField, rec) )

theDialog.getDialog.Open
```

35. A_RESCUEVIEW_LOG_VIEWER_PREVIOUS_RECORD

```
'******
'****** CALLED BY: A_RESCUEVIEW_OPEN_CONTROLS-View.Button
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_LOG_FILE_VIEWER_DIALOG")

theIdField = _theLogVtab.FindField("ID")
theDateField = _theLogVtab.FindField("Date")
theTypeField = _theLogVtab.FindField("Type")
theXField = _theLogVtab.FindField("X")
theYField = _theLogVtab.FindField("Y")
theMemoField = _theLogVtab.FindField("Memo")


Currentrec = theDialog.GetDialog.FindByName("txtRecord").GetLabel

rec = Currentrec.asNumber - 1

if (rec < 0) then
  MsgBox.Warning("No more records to view","LOG FILE VIEWER - END OF RECORDS")
  EXIT
End


theDialog.GetDialog.FindByName("txtRecord").Setlabel( _theLogVtab.ReturnValueString(theIdField, rec) )
theDialog.GetDialog.FindByName("txtDate").Setlabel( _theLogVtab.ReturnValueString(theDateField, rec) )
theDialog.GetDialog.FindByName("txtLogType").Setlabel( _theLogVtab.ReturnValueString(theTypeField, rec) )
theDialog.GetDialog.FindByName("txtXValue").Setlabel( _theLogVtab.ReturnValueString(theXField, rec) )
theDialog.GetDialog.FindByName("txtYValue").Setlabel( _theLogVtab.ReturnValueString(theYField, rec) )
theDialog.GetDialog.FindByName("tbox_Memo").SetText( _theLogVtab.ReturnValueString(theMemoField, rec) )

theDialog.getDialog.Open
```

36. A_RESCUEVIEW_OPEN_CONTROLS

```
'******
'****** CALLED BY: A_RESCUEVIEW_OPEN_CONTROLS-View.Button
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_CONTROL_DIALOG")

theDialog.getDialog.Open
```

37. A_RESCUEVIEW_OPEN_LOG_VIEWER_DIALOG

```
'******
'****** CALLED BY: A_RESCUEVIEW_OPEN_CONTROLS-View.Button
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

SELF.GetDialog.Close

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_LOG_FILE_VIEWER_DIALOG")

theIdField = _theLogVtab.FindField("ID")
theDateField = _theLogVtab.FindField("Date")
theTypeField = _theLogVtab.FindField("Type")
theXField = _theLogVtab.FindField("X")
theYField = _theLogVtab.FindField("Y")
theMemoField = _theLogVtab.FindField("Memo")


theBitMap = _theLogVtab.GetSelection
theBitmap.SetAll

therecCount = theBitMap.count


rec = theDialog.GetDialog.FindByName("txtRecord").GetLabel

if (rec.asNumber <= theRecCount) then
  rec = rec.asNumber
Else
  rec = 0
End



theDialog.GetDialog.FindByName("txtRecord").Setlabel( _theLogVtab.ReturnValueString(theIdField, rec) )
theDialog.GetDialog.FindByName("txtDate").Setlabel( _theLogVtab.ReturnValueString(theDateField, rec) )
theDialog.GetDialog.FindByName("txtLogType").Setlabel( _theLogVtab.ReturnValueString(theTypeField, rec) )
theDialog.GetDialog.FindByName("txtXValue").Setlabel( _theLogVtab.ReturnValueString(theXField, rec) )
theDialog.GetDialog.FindByName("txtYValue").Setlabel( _theLogVtab.ReturnValueString(theYField, rec) )
theDialog.GetDialog.FindByName("tbox_Memo").SetText( _theLogVtab.ReturnValueString(theMemoField, rec) )

theDialog.getDialog.Open
```

38. A_RESCUEVIEW_POINT_DISPLAY

```
'******
'****** CALLED BY:  ALL_POINT_DISPLAY_SCRIPTS
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*************************************************************************************

theXVal = SELF.Get(0)
theYVal = SELF.Get(1)
'theTextVal = SELF.Get(2)

thePoint = point.Make(theXVal,theYVal)

theView = av.GetActiveDoc
theGraphics = theView.GetGraphics

'mygText=GraphicText.Make(theTextVal,thePoint)
'mygText.setobjecttag("mygText")
'theGraphics.Add(mygText)


theExtent = theView.GetDisplay.ReturnExtent
theLowerLeft = theExtent.ReturnOrigin
w = (theExtent.GetWidth) * 0.05

' Create a hatched yellow circle with red outline centered in the display...
aCircle = Circle.Make(thePoint, w)
gCircle = GraphicShape.Make(aCircle)

aSymbol = RasterFill.Make
aSymbol.SetStyle(#RASTERFILL_STYLE_EMPTY)
aSymbol.SetOLColor(Color.GetRed)
aSymbol.SetOlWidth (2)
gCircle.SetSymbol(aSymbol)

theGraphics.Add(gCircle)

av.DelayedRun("A_RESCUEVIEW_DELETED_TMP_GRAPHICS",{gcircle},6)
'theView.Invalidate
```

39. A_RESCUEVIEW_RESOURCE_ADD_POINT_TO

```
'******
'****** CALLED BY:  A_RESCUEVIEW_ADD_CASUALTY_POSITION_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_ADD_RESOURCE_POSITION_DIALOG")

theName = SELF.GetName

theView = av.GetActiveDoc

thePoint = theView.GetDisplay.ReturnUserPoint

theXValue = thePoint.GetX
theYValue = thePoint.GetY


theXDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theXValue})
theYDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theYValue})


theDialog.GetDialog.FindByName("CtxtDegreeS").SetText(theYDMS.Extract(0))
theDialog.GetDialog.FindByName("CtxtDegreeW").SetText(theXDMS.Extract(0))
theDialog.GetDialog.FindByName("CtxtMinuteS").SetText(theYDMS.Extract(1))
theDialog.GetDialog.FindByName("CtxtMinuteW").SetText(theXDMS.Extract(1))

theDialog.GetDialog.FindByName("CtxtSecondS").SetText(theYDMS.Extract(2))
theDialog.GetDialog.FindByName("CtxtSecondW").SetText(theXDMS.Extract(2))


'********************** UPDATE DD FROM DMS VALUES ***************

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)
```

```
  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

theDialog.GetDialog.Open
```

40. A_RESCUEVIEW_RESOURCE_FLASH_POSITION

```
'******
'****** CALLED BY:  A_RESCUEVIEW_RESOURCE_POSITION_UPDATE_BUTTON
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theView = av.GetActiveDoc

theCTheme = theView.FindTheme("RESOURCE")

if (theCTheme = nil) then
  MsgBox.Error("No RESOURCE position has been logged","")
  EXIT
End

If (theCTheme.GetName <> "RESOURCE") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

End

theCFtab = theCTheme.GetFTab

'theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theXField = theCFtab.FindField("X")
theYField = theCFtab.FindField("Y")
'theTypeField = theCFtab.FindField("Type")
theCallsignField = theCFtab.FindField("Callsign")
'theCalledField = theCFtab.FindField("Called")
'theFoundField = theCFtab.FindField("Found")
'theStatusField = theCFtab.FindField("Status")
'theNoteField = theCFtab.FindField("Note")

theCBitmap = TheCFtab.GetSelection

theCList = {}

for each rec in theCFtab
  theCIDValue = theCFtab.ReturnValue(theIDField, rec)
  theCCallsignValue = theCFtab.ReturnValue(theCallsignField, rec)
  theCList.Add(theCIDValue.asString + " --- " + theCCallsignValue )
End

theChoice = MsgBox.ChoiceAsString (theCList, "Please select RESOURCE to flash position", "RescueVIEW -
RESOURCE Postion locator")

if (theChoice = Nil) then
  Exit
End
```

```
SearchStr = theChoice.Extract(0)
theQuery = "[ID] = " + SearchStr
theCFTab.Query(theQuery, theCBitmap, #VTAB_SELTYPE_NEW)
theCFTab.UpdateSelection

theCount = theCBitmap.Count

For each uprec in theCBitmap
  theXValue = theCFtab.ReturnValue(theXField, uprec)
  theYValue = theCFtab.ReturnValue(theYField, uprec)
End

theDisplay = theView.GetDisplay
theExtent = theDisplay.ReturnExtent

theVisible = theExtent.Intersects (theXValue@theYValue)


if (theVisible) then

  av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theXValue, theYValue} )

Else

  theDisplay.PanTo (theXValue@theYValue)
  av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theXValue, theYValue} )

End


theCBitmap.ClearAll
```

41. A_RESCUEVIEW_RESOURCE_POSITION_MANUAL_UPDATE

```
'******
'****** CALLED BY:  A_RESCUEVIEW_RESOURCE_POSITION_DIALOG
'******
'****** CALLS MADE: A_RESCUEVIEW_RESOURCE_TRACE
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theView = av.GetActiveDoc

theCTheme = theView.FindTheme("RESOURCE")

if (theCTheme = nil) then
  MsgBox.Error("No RESOURCE position has been logged","")
  EXIT
End

If (theCTheme.GetName <> "RESOURCE") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

End

theCFtab = theCTheme.GetFTab

theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theXField = theCFtab.FindField("X")
theYField = theCFtab.FindField("Y")
theTypeField = theCFtab.FindField("Type")
theCallsignField = theCFtab.FindField("Callsign")
'theCalledField = theCFtab.FindField("Called")
theStatusField = theCFtab.FindField("Status")
theNoteField = theCFtab.FindField("Note")

theCBitmap = TheCFtab.GetSelection

theCount = theCBitmap.Count

theCallsignValue = SELF.GetDialog.FindByName("cTxtCallSign").GetText

theRDEGX = (SELF.GetDialog.FindByName("CtxtDegreeW").GetText).asNumber
theRMINX = (SELF.GetDialog.FindByName("CtxtDMinuteW").GetText).asNumber

theRDEGY = (SELF.GetDialog.FindByName("CtxtDegreeS").GetText).asNumber
theRMINY = (SELF.GetDialog.FindByName("CtxtDMinuteS").GetText).asNumber

theTypeValue = SELF.GetDialog.FindByName("cBoxResourceType").GetCurrentValue
theStatusValue = SELF.GetDialog.FindByName("cBoxStatus").GetCurrentValue
theNote = SELF.GetDialog.FindByName("cBoxNote").GetText
```

```
'theMessage = ""

theCFtab.SetEditable(TRUE)

If (theCFtab.IsEditable) then

 for each uprec in theCBitmap

  if ((SELF.GetDialog.FindByName("CtxtDegreeW").GetText <> "") or
(SELF.GetDialog.FindByName("CtxtDegreeS").GetText <> ""))then
    theXValue = theRDEGX + theRMINX
    theYValue = theRDEGY - theRMINY

    theIDValue = theCFtab.returnValue(theIDField, uprec)
    theOldShapeValue = theCFtab.returnValue(theShapeField, uprec)
    theXValueOld = theOldShapeValue.GetX
    theYValueOld = theOldShapeValue.GetY

    av.Run("A_RESCUEVIEW_RESOURCE_TRACE", {theIDValue.asString, theXValueOld, theYValueOld,
theXValue, theYValue} )

    theNewShapeValue = theXValue@theYValue

    if (theOldShapeValue <> theNewShapeValue) then

      theCFtab.SetValue(theShapeField, uprec, theXValue@theYValue)
      theCFtab.SetValue(theXField, uprec, theXValue)
      theCFtab.SetValue(theYField, uprec, theYValue)

      if (theXValue > 0) then
       theXDegree = theXValue.asString ++ "EAST"
      Else
       theXDegree = theXValue.asString ++ "WEST"
      End

      if (theYValue > 0) then
       theYDegree = theYValue.asString ++ "NORTH"
      Else
       theYDegree = theYValue.asString ++ "SOUTH"
      End

      av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theXValue, theYValue} )

      '*****************************************************************************************************
      '************************ ADD LOG FILE ENTRY ********************************************************
      '*****************************************************************************************************
      '*** ENTRY TYPES
      '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED,
-UPDATE CASUALTY
      '*****************************************************************************************************

       theType = "UPDATE RESOURCE"
       'theXValue = Nil
       'theYValue = Nil
       theMessage = "Position Update to " + theXDegree + " and " + theYDegree

       av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

      '*****************************************************************************************************
      '************************ END LOG FILE ENTRY ********************************************************
```

```
      '*****************************************************************************************************

    End
  End


  theOldTypeValue = theCFtab.returnValue(theTypeField, uprec)
  theOldStatusValue = theCFtab.returnValue(theStatusField, uprec)

  if (theTypeValue <> theOldTypeValue) then
    theCFtab.SetValue(theTypeField, uprec, theTypeValue)
    theMessage = "Type Update from " + theOldTypeValue + " to " + theTypeValue

    '*****************************************************************************************************
    '********************** ADD LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************
    '*** ENTRY TYPES
    '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
    '*****************************************************************************************************

      theType = "UPDATE RESOURCE"
      theXValue = nil
      theYValue = nil

      av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

    '*****************************************************************************************************
    '********************** END LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************
  End

  if (theStatusValue <> theOldStatusValue) then
    theCFtab.SetValue(theStatusField, uprec, theStatusValue)
    theMessage = "Status Update from " + theOldStatusValue + " to " + theStatusValue

    '*****************************************************************************************************
    '********************** ADD LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************
    '*** ENTRY TYPES
    '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
    '*****************************************************************************************************

      theType = "UPDATE RESOURCE"
      theXValue = nil
      theYValue = nil

      av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

    '*****************************************************************************************************
    '********************** END LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************
  End


  theOldCallsignValue = theCFtab.returnValue(theCallsignField, uprec)
  theOldNoteValue = theCFtab.returnValue(theNoteField, uprec)

  if (theCallsignValue <> theOldCallsignValue) then
```

```
    theCFtab.SetValue(theCallsignField, uprec, theCallsignValue)
    theMessage = "Callsign Update from " + theOldCallsignValue + " to "
+SELF.GetDialog.FindByName("cTxtCallSign").GetText

    '*****************************************************************************************************
    '********************** ADD LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************
    '*** ENTRY TYPES
    '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
    '*****************************************************************************************************

      theType = "UPDATE CASUALTY"
      theXValue = nil
      theYValue = nil

      av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

    '*****************************************************************************************************
    '********************** END LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************
   End

   if (theNote <> theOldNoteValue) then
    theCFtab.SetValue(theNoteField, uprec, theNote)
    theMessage = "Note Update : " +NL+ theNote

    '*****************************************************************************************************
    '********************** ADD LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************
    '*** ENTRY TYPES
    '*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
    '*****************************************************************************************************

      theType = "UPDATE RESOURCE"
      theXValue = nil
      theYValue = nil

      av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

    '*****************************************************************************************************
    '********************** END LOG FILE ENTRY ***********************************************************
    '*****************************************************************************************************
   End

  End
End

theCFtab.SetEditable(False)

theCFtab.Flush
theCBitmap.ClearAll
theCTheme.Invalidate(TRUE)

SELF.GetDialog.Close
```

42. A_RESCUEVIEW_RESOURCE_POSITION_UPDATE_DIALOG

```
'******
'****** CALLED BY:  A_RESCUEVIEW_RESOURCE_POSITION_UPDATE_BUTTON
'******
'****** CALLS MADE: A_RESCUEVIEW_CONVERT_DD_TO_DMS
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

SELF.GetDialog.Close

theView = av.GetActiveDoc

theCTheme = theView.FindTheme("RESOURCE")

if (theCTheme = nil) then
  MsgBox.Error("No RESOURCE position has been logged","")
  EXIT
End

If (theCTheme.GetName <> "RESOURCE") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

End

theCFtab = theCTheme.GetFTab

'theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theXField = theCFtab.FindField("X")
theYField = theCFtab.FindField("Y")
theTypeField = theCFtab.FindField("Type")
theCallsignField = theCFtab.FindField("Callsign")
theCalledField = theCFtab.FindField("Called")
theFoundField = theCFtab.FindField("Found")
theStatusField = theCFtab.FindField("Status")
theNoteField = theCFtab.FindField("Note")

theCBitmap = TheCFtab.GetSelection

theCList = {}

for each rec in theCFtab
  theCIDValue = theCFtab.ReturnValue(theIDField, rec)
  theCCallsignValue = theCFtab.ReturnValue(theCallsignField, rec)
  theCList.Add(theCIDValue.asString + " --- " + theCCallsignValue )
End

theChoice = MsgBox.ChoiceAsString (theCList, "Please select RESOURCE to update position", "RescueVIEW -
RESOURCE Postion update")

if (theChoice = Nil) then
  Exit
End
```

```
SearchStr = theChoice.Extract(0)
theQuery = "[ID] = " + SearchStr
theCFTab.Query(theQuery, theCBitmap, #VTAB_SELTYPE_NEW)
theCFTab.UpdateSelection

theCount = theCBitmap.Count

For each uprec in theCBitmap
  theXValue = theCFtab.ReturnValue(theXField, uprec)
  theYValue = theCFtab.ReturnValue(theYField, uprec)
  theTypeValue = theCFtab.ReturnValue(theTypeField, uprec)
  theCallsignValue = theCFtab.ReturnValue(theCallsignField, uprec)
  theCalledValue = theCFtab.ReturnValue(theCalledField, uprec)
 ' theFoundValue = theCFtab.ReturnValue(theFoundField, uprec)
  theCallsignValue = theCFtab.ReturnValue(theCallsignField, uprec)
  theStatusValue = theCFtab.ReturnValue(theStatusField, uprec)
  theNoteValue = theCFtab.ReturnValue(theNoteField, uprec)
End

theXDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theXValue})
theYDMS = av.run("A_RESCUEVIEW_CONVERT_DD_TO_DMS", {theYValue})

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_UPDATE_RESOURCE_POSITION_DIALOG")

theDialog.GetDialog.FindByName("CtxtDegreeS").SetText(theYDMS.Extract(0))
theDialog.GetDialog.FindByName("CtxtDegreeW").SetText(theXDMS.Extract(0))
theDialog.GetDialog.FindByName("CtxtMinuteS").SetText(theYDMS.Extract(1))
theDialog.GetDialog.FindByName("CtxtMinuteW").SetText(theXDMS.Extract(1))

theDialog.GetDialog.FindByName("CtxtSecondS").SetText(theYDMS.Extract(2))
theDialog.GetDialog.FindByName("CtxtSecondW").SetText(theXDMS.Extract(2))

theDialog.GetDialog.FindByName("cTxtCallsign").SetText(theCallsignValue)

theDialog.GetDialog.FindByName("cboxNote").SetText(theNoteValue)


'********************** UPDATE DD FROM DMS VALUES ***************

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteW").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondW").GetText).asNumber / 3600
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteW").SetText(theDecMin.asString)

  theMin = (theDialog.GetDialog.FindByName("CtxtMinuteS").GetText).asNumber / 60
  theSec = (theDialog.GetDialog.FindByName("CtxtSecondS").GetText).asNumber / 3600
```

```
  theDecMin = theMin + theSec

theDialog.GetDialog.FindByName("CtxtDMinuteS").SetText(theDecMin.asString)


theFileName = (System.GetEnvVar("RESCUEVIEW") + "\Data\ResourceType.dbf")
theCallVtab = Vtab.Make(theFileName.asFileName, FALSE, FALSE)
theType = theCallVTab.findField("Type")

theComboBox = (theDialog.GetDialog).FindByName("cBoxResourceType")
theComboBox.DefineFromVTab (theCallVTab, theType, False)
theComboBox.FindByValue (theTypeValue)
theComboBox.SetCurrentValue (theTypeValue)
theComboBox.SelectCurrent

theFileName = (System.GetEnvVar("RESCUEVIEW") + "\Data\RStatus.dbf")
theCallVtab = Vtab.Make(theFileName.asFileName, FALSE, FALSE)
theStatus = theCallVTab.findField("Status")

theComboBox = (theDialog.GetDialog).FindByName("cBoxStatus")
theComboBox.DefineFromVTab (theCallVTab, theStatus, False)
theComboBox.FindByValue (theStatusValue)
theComboBox.SetCurrentValue (theStatusValue)
theComboBox.SelectCurrent


theDialog.getDialog.Open
```

43. A_RESCUEVIEW_RESOURCE_POSITION_UPDATE_SCREEN

```
'******
'****** CALLED BY:  A_RESCUEVIEW_CONTROL_DIALOG
'******
'****** CALLS MADE: A_RESCUEVIEW_RESOURCE_TRACE
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theView = av.GetActiveDoc

theCTheme = theView.FindTheme("RESOURCE")

if (theCTheme = nil) then
  MsgBox.Error("No RESOURCE position has been logged","")
  EXIT
End

If (theCTheme.GetName <> "RESOURCE") then

  MsgBox.Error("PLEASE USE RESOURCE UPDATE TOOL","")
  EXIT

End

theCFtab = theCTheme.GetFTab

theShapeField = theCFtab.FindField("Shape")
theIDField = theCFtab.FindField("ID")
theXField = theCFtab.FindField("X")
theYField = theCFtab.FindField("Y")
theTypeField = theCFtab.FindField("Type")
theCallsignField = theCFtab.FindField("Callsign")
theNoteField = theCFtab.FindField("Note")


theCBitmap = TheCFtab.GetSelection

theCList = {}

for each rec in theCFtab
  theCIDValue = theCFtab.ReturnValue(theIDField, rec)
  theCCallsignValue = theCFtab.ReturnValue(theCallsignField, rec)
  theCList.Add(theCIDValue.asString + " --- "+ theCCallsignValue )
End


theChoice = MsgBox.ChoiceAsString (theCList, "Please select RESOURCE to update position", "RescueVIEW -
RESOURCE Postion update")

if (theChoice = Nil) then
  Exit
End
```

```
SearchStr = theChoice.Extract(0)
theQuery = "[ID] = " + SearchStr
theCFTab.Query(theQuery, theCBitmap, #VTAB_SELTYPE_NEW)
theCFTab.UpdateSelection

theCount = theCBitmap.Count

SELF.SetTag("Operate")


 thePoint = theView.GetDisplay.ReturnUserPoint
 theXValue = thePoint.GetX
 theYValue = thePoint.GetY

 theCFtab.SetEditable(TRUE)

 If (theCFtab.IsEditable) then

  for each uprec in theCBitmap

   theOldShapeValue = theCFtab.ReturnValue(theShapeField, uprec)

   theXValueOld = theOldShapeValue.GetX
   theYValueOld = theOldShapeValue.GetY

   av.Run("A_RESCUEVIEW_RESOURCE_TRACE", {SearchStr, theXValueOld, theYValueOld, theXValue,
theYValue} )


   theCFtab.SetValue(theShapeField, uprec, thePoint)
   theCFtab.SetValue(theXField, uprec, theXValue)
   theCFtab.SetValue(theYField, uprec, theYValue)
  End

 End

 theCFtab.SetEditable(False)

 theCFtab.Flush

 if (theXValue > 0) then
  theXDegree = theXValue.asString ++ "EAST"
 Else
  theXDegree = theXValue.asString ++ "WEST"
 End

 if (theYValue > 0) then
  theYDegree = theYValue.asString ++ "NORTH"
 Else
  theYDegree = theYValue.asString ++ "SOUTH"
 End

av.Run( "A_RESCUEVIEW_POINT_DISPLAY",{theXValue, theYValue} )

'*************************************************************************************************
'********************** ADD LOG FILE ENTRY ***************************************************
'*************************************************************************************************
'*** ENTRY TYPES
```

```
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*****************************************************************************************************

  theType = "USER"
'  theXValue = Nil
'  theYValue = Nil
  theMessage = theCCallsignValue ++ "Position Update to " + theXDegree + " and " + theYDegree

  av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

'*****************************************************************************************************
'************************ END LOG FILE ENTRY ***********************************************************
'*****************************************************************************************************

theCBitmap.ClearAll
theCTheme.Invalidate (TRUE)
```

44. A_RESCUEVIEW_RESOURCE_TRACE

```
'******
'******        CALLED        BY:              A_RESCUEVIEW_RESOURCE_POSITION_MANUAL_UPDATE;
A_RESCUEVIEW_RESOURCE_POSITION_UPDATE_SCREEN
'******
'****** CALLS MADE :  NONE
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******        Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************


theID = SELF.Get(0)
theXValue1 = SELF.Get(1)
theYValue1 = SELF.Get(2)
theXValue2 = SELF.Get(3)
theYValue2 = SELF.Get(4)

theView = av.GetActiveDoc

theTheme = av.GetActiveDoc.FindTheme("Resource Trace")

if (theTheme = Nil) then

 theFileName = Filename.Make( _theOperationPath +"\trace.shp" )

 if (theFileName <> nil) then
  theFtab = Ftab.MakeNew( theFileName, Polyline )
  theIDfield = Field.Make ("ID", #FIELD_BYTE  , 50, 0)

  theFtab.AddFields( {theIDField} )
  theTheme = FTheme.Make(theFtab)
  theTheme.SetName( "Resource Trace" )
  theView.AddTheme(theTheme)

  theShapefield = theFtab.FindField ("Shape")

  theLinelist = {theXvalue1@theYValue1, theXvalue2@theYValue2}
  thePolyLine = PolyLine.Make ({theLineList})

  rec = theFtab.Addrecord
  theFtab.SetValue(theShapeField, rec, thePolyLine)
  theFtab.SetValue(theIDField, rec, theID)
  theFtab.SetEditable(FALSE)

 Else
  MsgBox.Error("Can't create trace file" +NL+ "Tracking of resource not possible" +NL+ "Check Read/Write
Permissions","")
  EXIT
 end

Else

 theFtab = theTheme.GetFtab
 theShapefield = theFtab.FindField ("Shape")
```

```
  theIDfield = theFtab.FindField ("ID")

  theBitmap = theFtab.getSelection
  theQuery = "[ID] = " + theID
  theFTab.Query(theQuery, theBitmap, #VTAB_SELTYPE_NEW)
  theFTab.UpdateSelection

  theFtab.SetEditable(TRUE)

  if (theBitmap.count = 0) then

    theLinelist = {theXvalue1@theYValue1, theXvalue2@theYValue2}
    thePolyLine = PolyLine.Make ({theLineList})

    rec = theFtab.Addrecord
    theFtab.SetValue(theShapeField, rec, thePolyLine)
    theFtab.SetValue(theIDField, rec, theID)

  Else

    For each rec in theBitmap

      theShape = theFtab.ReturnValue(theShapeField, rec)
      thePolyLine = PolyLine.Make ({{theXvalue1@theYValue1, theXvalue2@theYValue2}})
      theNewPolyLine = thePolyLine.ReturnMerged (theShape)

      theFtab.SetValue(theShapeField, rec, theNewPolyLine)
    End

  End

  theFtab.SetEditable(FALSE)
  theBitmap.clearAll

End


theLegend = theTheme.GetLegend
theLegendFile = (System.GetEnvVar("RESCUEVIEW") + "\Data\Trace.avl").asFilename
theLegend.Load(theLegendFile, #LEGEND_LOADTYPE_ALL)

if (theTheme.IsVisible = FALSE) then
  theTheme.SetVisible(TRUE)
End
```

45. A_RESCUEVIEW_STOP_OPERATION

```
'******
'****** CALLED BY:
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

If (MsgBox.YesNo("Please confirm that you want to stop this operation"+NL+NL+
            "A Stand Down' cannot be re-opened","RescueVIEW - OPERATION STAND DOWN",TRUE)) then


  '*************************************************************************************************
  '************************ ADD LOG FILE ENTRY ****************************************************
  '*************************************************************************************************
  '*** ENTRY TYPES
  '*** - SYSTEM, -OPERATION, -USER, -NEW RESOURCE, -UPDATE RESOURCE
  '*************************************************************************************************

  theType = "SYSTEM"
  theXValue = Nil
  theYValue = Nil
  theMessage = "***************** OPERATION STAND DOWN - LOGFILE CLOSED FOR ENTRIES
************************"

  av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})

  '*************************************************************************************************
  '************************ END LOG FILE ENTRY ****************************************************
  '*************************************************************************************************

Else

  EXIT

End


if (MsgBox.YesNo("Do you want to save a project file","RescueVIEW Project Save", TRUE)) then

  theProject = av.GetProject
  if (av.Run("Project.CheckForEdits",nil).Not) then
   return nil
  end

  theDate = ((Date.Now).SetFormat("yyyyMMddhhm")).asString

  theDefname = (_theOperationPath + "\" + theDate+".apr").asFileName

  if (theDefname = nil) then
   if ((System.GetEnvVar("RescueView") <> nil) and File.IsWritable("RescueView".AsFileName)) then
    theDefname = FileName.Make(_theOperationpath).MakeTmp(theDate, "apr")
   else
```

```
      MsgBox.Error("Could not save project."+NL+"Check permissions","")
    end
  end

'   theFName = FileDialog.Put(defName, "*.apr", "Save Project As")
  if (theDefname <> nil) then
    theProject.SetFileName(theDefname)
    if (theProject.Save) then
      av.ShowMsg( "Project saved to '"+theProject.GetFileName.GetBaseName+"'")
      if (System.GetOS = #SYSTEM_OS_MAC) then
        realFName = theProject.GetFileName
        if (nil <> realFName) then
          Script.Make("MacClass.SetDocInfo(SELF, Project)").DoIt(realFName)
        end
      end
    end
  end

End

av.GetActiveGUI.GetMenuBar.FindByScript ("A_RESCUEVIEW_CREATE_OPERATION").SetEnabled(True)

SELF.SetEnabled(FALSE)

av.GetActiveGUI.GetButtonBar.FindByScript ("A_RESCUEVIEW_OPEN_CONTROLS").SetEnabled(False)

Msgbox.Info("Operation Stopped","RescueVIEW - OPERATION STAND DOWN")
'_theOperationPath = Nil
av.PurgeObjects
```

46. A_RESCUEVIEW_VARIANCE_DIALOG_OPEN

```
'******
'****** CALLED BY:  A_RESCUEVIEW_CONTROL_DIALOG_OPTIONS
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theControlDialog = SELF.GetDialog

theControlDialog.Close


thePath = System.GetEnvVar ("RescueView")

theVarianceFile = thePath + "\Data\Variance.odb"

VarODB = ODB.Open(theVarianceFile.AsFileName)

theVar = VarODB.Get(0)
theEast = VarODB.Get(1)

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_SET_VARIANCE_DIALOG").GetDialog

theDialog.FindByName("txtDegreeV").SetText(theVar.asString)
theDialog.FindByName("txtDegreeEW").SetText(theEast)

theDialog.Open
```

47. A_RESCUEVIEW_VARIANCE_UPDATE

```
'******
'****** CALLED BY:  A_RESCUEVIEW_SET_VARIANCE_DIALOG
'******
'****** CALLS MADE:
'******
'******
'****** WRITTEN BY: Kobus Meyer
'******          Developed as part of Master Thesis - GIS to the Rescue
'*********************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_SET_VARIANCE").GetDialog

theVar = theDialog.FindByName("txtDegreeV").GetText

if (theVar.asNumber > 360) then
  MsgBox.Error("Variance value can only be between 0 and 360","")
  EXIT
End

theEast = theDialog.FindByName("txtDegreeEW").GetText

if ( (theEast.UCase = "E") or (theEast.UCase = "W") ) then
  Else
    MsgBox.Error("Variance can only be E or W"+NL+"If variance is 0 then add E or W to it","")
    Exit
End

thePath = System.GetEnvVar ("RescueView")

theVarianceFile = thePath + "\Data\Variance.odb"

VarODB = ODB.Make(theVarianceFile.AsFileName)

VarODB.Add(theVar.asNumber)
VarODB.Add(theEast.UCase)

VarODB.Commit

theDialog.Close

'*************************************************************************************************
'************************ ADD LOG FILE ENTRY ***************************************************
'*************************************************************************************************
'*** ENTRY TYPES
'*** - SYSTEM, -OPERATION, -USER, -RESOURCE ADDED, -UPDATE RESOURCE, -CASUALTY ADDED, -
UPDATE CASUALTY
'*************************************************************************************************

  theType = "SYSTEM"
  theXValue = Nil
  theYValue = Nil
  theMessage = "Variance updated to:" ++ theVar + theEast.UCase

  av.run("A_RESCUEVIEW_CREATE_LOG_UPDATE",{theType, theXvalue, theYValue, theMessage})
```

```
'*****************************************************************************************************
'********************** END LOG FILE ENTRY ***********************************************************
'*****************************************************************************************************

theDialog = av.GetProject.FindDoc("A_RESCUEVIEW_CONTROL_DIALOG").GetDialog

theDialog.Open
```

---------- **END SCRIPTS** ----------