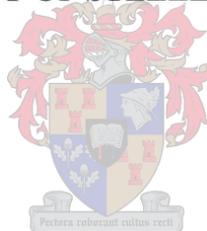


**AN INFRASTRUCTURE MANAGEMENT SUPPORT SYSTEM FOR
WESTERN CAPE NATURE CONSERVATION BOARD**

NICOLAAS MILNE VAN ZYL

**THESIS PRESENTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTERS OF ARTS AT THE
UNIVERSITY OF STELLENBOSCH**



**SUPERVISORS: MR, PJ, ELOFF
PROF, HL, ZIETSMAN**

MARCH 2003

DECLARATION

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

ABSTRACT

This thesis investigates the use of GIS (Geographical Information Systems) to develop an infrastructure management support system for the Western Cape Nature Conservation Board (WCNCB). The primary goal was to design a system to help the managers with their task of managing the infrastructure of a reserve. It involved the development and description of a system in ArcView with the programming language Avenue in conjunction with an Access application developed in Visual Basic for Applications. The end result was a system that can create maps of all the different infrastructure features with ArcView and use an open-ended Access application to input data. The data are stored in an Access database. The thesis describes the user functionality of the system. Basic reporting facilities are provided and the data and system have the potential to provide essential reporting in future development. The conclusion of this thesis is that GIS could fulfil the role of an Infrastructure Management Support System for WCNCB.

OPSOMMING

Die tesis ondersoek die gebruik van GIS (Geografiese Inligting Stelsels) in die ontwikkeling van 'n infrastruktuur bestuurshulpmiddel vir Wes-Kaap Natuurbewaringsraad. Die primêre doel van hierdie studie is om 'n sisteem te ontwikkel wat die bestuurders van die verskillende natuurreserve kan bystaan in die bestuur van hul reserve. Die tesis beskryf die ontwikkeling van 'n infrastruktuur bestuurshulpmiddel met ArcView se programmeringstaal Avenue. Tesame hiermee is 'n Access applikasie wat in Visual Basic for Applications ontwikkel is geïntegreer. Die eind-resultaat is 'n sisteem wat kaarte met ArcView vanaf gestoorde data in 'n Access databasis kan produseer en ook dataïnvordering kan hanteer. Die tesis beskryf die ontwikkeling en funksionaliteit van die sisteem. Daar word voorsiening gemaak vir basiese verslagelêrende funksies en vir toekomstige meer gevorderde analyses in die data samestelling. Die gevolgtrekking wat in die tesis gemaak word is dat GIS die rol van 'n infrastruktuur bestuurshulpmiddel kan vervul vir Wes-Kaap Natuurbewaringsraad.

ACKNOWLEDGEMENTS

Expertise

Department of Geography and Environmental Studies, University of Stellenbosch
Western Cape Nature Conservation Board

Tim Sutton for his help with the development of the Infrastructure management support system application.

Andrew Turner for his help with the development of the Infrastructure management support system application.

Financial support

My parents Bertie and Daleen van Zyl

CONTENTS

DECLARATION.....	i
ABSTRACT	ii
OPSOMMING.....	ii
ACKNOWLEDGEMENTS.....	iii
CHAPTER 1 INFRASTRUCTURE MANAGEMENT.....	1
1.1 INTRODUCTION	1
1.2 THE MISSION, GOALS AND FUNCTIONS OF WESTERN CAPE NATURE CONSERVATION BOARD (WCNCB).....	2
1.3 WESTERN CAPE NATURE CONSERVATION BOARD - GIS DEPARTMENT.....	3
<i>1.3.1 Vision</i>	<i>3</i>
<i>1.3.2 Mission Statement.....</i>	<i>4</i>
<i>1.3.3 Goals</i>	<i>4</i>
1.4 RESEARCH PROBLEM FORMULATION.....	5
1.5 METHODOLOGY.....	5
CHAPTER 2 NEEDS ASSESSMENT FOR WCNCB INFRASTRUCTURE MANAGEMENT.....	7
2.1 INFRASTRUCTURE INVENTORY OF THE WCNCB.....	7
2.2 WCNCB MANAGERS.....	10
<i>2.2.1 Role and Tasks of the WCNCB Park Manager</i>	<i>10</i>
<i>2.2.2 Computer Skill Levels of Managers</i>	<i>11</i>
2.3 DECISION-SUPPORT SYSTEMS (DSS) AND GIS.....	13
2.4 THE IMPORTANCE OF GIS IN SPATIAL DECISION SUPPORT SYSTEMS (SDSS)...	14
2.5 INFORMATION TECHNOLOGY AVAILABILITY.....	15
<i>2.5.1 Computer Hardware.....</i>	<i>15</i>
<i>2.5.2 Networks</i>	<i>16</i>
<i>2.5.3 Database</i>	<i>17</i>
2.5.3.1 Management and Quality Control	18
2.5.3.2 Data Needs	20
<i>2.5.4 Software</i>	<i>21</i>
2.5.4.1 Operating System	21
2.5.4.2 GIS Software.....	21

3.2.5.4.3 Database Management System (DBMS)	22
CHAPTER 3 DESIGN OF THE INFRASTRUCTURE MANAGEMENT SUPPORT SYSTEM	23
3.1 PRELIMINARY PLANNING.....	23
3.2 DESIGN.....	23
3.2.1 <i>Relational Database Design</i>	23
3.2.2 <i>Graphical User Interface (GUI)</i>	27
3.2.2.1 Infrastructure Database Application.....	28
3.2.2.2 GIS Extension.....	35
3.2.3 <i>Information Flow</i>	39
3.3 IMPLEMENTATION	44
3.3.1 <i>Programming</i>	44
3.3.1.1 ArcView Programming.....	44
3.3.1.2 Microsoft Access Programming.....	47
3.3.2 <i>Integration</i>	47
3.3.3 <i>Testing, and Debugging Corrections</i>	51
CHAPTER 4 APPLICATION OF AN INFRASTRUCTURE MANAGEMENT SUPPORT SYSTEM	56
4.1 OPERATION OF THE IMSS	56
4.1.1 <i>Access Database</i>	56
4.1.1.1 Adding Features.....	56
4.1.1.2 Adding Properties	57
4.1.1.3 Data Entering.....	57
4.1.2 <i>ArcView GIS</i>	61
4.1.2.1 Using the <i>Data Select</i> Tool	61
4.1.2.2 Spatial Data Editing Tool.....	62
4.1.2.3 Spatial Data Adding Tool	63
CHAPTER 5 SUMMARY, EVALUATION AND FURTHER DEVELOPMENT	64
CHAPTER 6 REFERENCES	67
CHAPTER 7 APPENDICES	76

FIGURES

Figure 1.1 Research design outline	6
Figure 2.1 Diagram of network system showing LAN and WAN	17
Figure 3.1 IMSS Relational database relationships.....	25
Figure 3.2 a) A table before normalisation was done.	26
Figure 3.2 b) Illustration of the table in a) after normalisation.....	26
Figure 3.3 Example of WCNCB GUI standard	28
Figure 3.4 Database start form.....	29
Figure 3.5 Example of Infrastructure Tool with Feature selected	30
Figure 3.6 Example of Infrastructure tool with Inspection tab selected.	31
Figure 3.7 Picture display form	32
Figure 3.8 Example of the Infrastructure Tool with Pointdata tab selected.....	33
Figure 3.9 Example of a New Feature Form	34
Figure 3.10 Example of New Property adding form.....	34
Figure 3.11 The ArcView menu bar after the IMSS extension was loaded.	35
Figure 3.12 The layout of the Infrastructure item's dropdown list.....	35
Figure 3.13 The layout of the infra tool form.	36
Figure 3.14 An example of the <i>Infrastructure data Select</i> Tool Dialog.	37
Figure 3.15 An example of the <i>Property data select</i> dialog addition.	38
Figure 3.16 An example of the spatial data-adding tool.	38
Figure 3.17 An example of the Editing Tool.	39
Figure 3.18 ArcView data Editing flowchart.....	41
Figure 3.19 ArcView Spatial Data Adding Flowchart.....	42
Figure 3.20 Flowchart of Spatial Data in Microsoft Access	433
Figure 3.21 Screenshot of ODBC Data Source Administrator.....	49
Figure 3.22 Screenshot of form 'Create New Data Source'	50
Figure 3.23 Screenshot of the 'ODBC Microsoft Access set up' form.....	50
Figure 3.24 Screenshot of the 'Select database' form.....	51
Figure 3.25 Cederberg Pointdata sample.....	53
Figure 3.26 Sample of Cederberg line test data.....	54
Figure 4.1 Example of a report on the condition of the entire infrastructure of a nature reserve.	59
Figure 4.2 Example of an inspection report that shows what percentage of the total inspections were done by specific persons.	59
Figure 4.3 Sample of Cederberg reserve inventory list.....	60

CHAPTER 1 INFRASTRUCTURE MANAGEMENT

1.1 INTRODUCTION

The building of man-made structures has always represented progress in human society. In general, man-made structures can be considered as infrastructure, but according to Verhoef (1999) a more detailed definition for infrastructure would be: "... a large-scale technological system, consisting of immovable physical facilities and delivering (an) essential public or private service(s) through the storage, conversion and/or transportation of certain commodities. The infrastructure includes those parts and subsystems necessary for fulfilling the primary storage, transportation and/or conversion function(s) as well as those supporting a proper execution of the primary function(s)" (Verhoef 1999:1). Hudson, Haas and Uddin (1997) see a nation's economic strength as revealed in its infrastructural assets. The National Science Foundation states: "A civilization's rise and fall is linked to its ability to feed and shelter its people and defend itself. These capabilities depend on infrastructure – the underlying, often hidden foundation of a society's wealth and quality of life. A society that neglects its infrastructure loses the ability to transport people and food, provide clean air and water, control disease and conduct commerce" (NSF 1994:4). From this quote it is clear that infrastructure plays a very important role in society.

The interaction of man with the earth's surface (Miller 1996) is greatly influenced by the way the man-made structures, of which infrastructure forms a large percentage, are managed. Therefore the following two definitions of infrastructure management are provided in order to clarify the meaning of the terms:

Management can mean to administer, to control or to co-ordinate the various elements of a unit or system. A dictionary definition is: "the judicious use of means to accomplish an end"(Hudson, Haas & Uddin1997).

Infrastructure management includes the systematic, co-ordinated planning and programming of investments or expenditures and physical facilities design, construction, maintenance, operation and in-service evaluation. It is a wide-ranging process, covering those activities involved in providing and maintaining infrastructure at a level of service acceptable to the public or the owner (Hudson, Haas & Uddin 1997).

To help the manager with the task of managing infrastructure, a decision support system can be implemented. A decision support system (DSS) is a computer program application that analyses industry data and presents them in a way that simplifies business decisions for its users. This is called an "informational application" (Densham 1991).

The Western Cape Nature Conservation Board (WCNCB) requires such a system to assist them in managing their infrastructure. As infrastructure has spatial properties, it is natural to consider using a Geographical Information System (GIS) in the development of the decision support system (Berry & Ripple 1994).

1.2 THE MISSION, GOALS AND FUNCTIONS OF THE WESTERN CAPE NATURE CONSERVATION BOARD (WCNCB)

To develop and/or implement a decision support system for the WCNCB, it is necessary to examine the mission statements of the WCNCB and their GIS Department in order to align the system with their needs. The following section, therefore, deals with the mission, goals and functions of these bodies.

The mission statement of WCNCB states that its mission is the conservation of the natural heritage of the Western Cape for the benefit, well-being and enjoyment of present and future generations. The following goals can be derived from the mission statement:

- Maintain ecological systems and processes;
- Conserve genetic diversity;
- Conserve (and utilise) the natural heritage of the Western Cape Province; and
- Ensure the sustainable utilisation of species and ecosystems.

In order to achieve these goals, the functions of the WCNCB are:

- To prevent the unnatural extinction of any species indigenous to the Western Cape;
- To formulate and apply legislation to ensure the conservation of the Western Cape's natural heritage;

- To establish, manage and conserve reserves representative of each ecological region of the Western Cape;
- To provide scientific services to support conservation programmes;
- To communicate to all people the value of the natural environment and the necessity of conservation;
- To promote the sustainable utilisation of natural resources;
- To evaluate development proposals to ensure that environmental quality is maintained;
- To provide scientific services to support conservation programmes;
- To provide visitor facilities and services in nature reserves;
- To formulate and apply legislation to ensure the conservation of the Western Cape's natural heritage;
- To evaluate development proposals to ensure that environmental quality is maintained; and
- To conserve sites of cultural-historical significance on reserves in accordance with the mission and goals of the Western Cape Nature Conservation Board (De Klerk & Sutton 2000).

1.3 WESTERN CAPE NATURE CONSERVATION BOARD - GIS DEPARTMENT

To provide a background to, and clarify the aims and research methodology of, this thesis an in-depth look at the GIS Department of WCNCB is needed.

1.3.1 Vision

The following section presents a summary of the WCNCB GIS Department's vision.

The vision is to provide access to GIS data and facilities for all WCNCB personnel and provide them with the necessary training to fulfil the functions they need to use spatial data or spatial analysis.

Furthermore the vision of WCNCB is to provide data to personnel to facilitate and promote environmentally sensitive and sustainable development within the Western Cape (Somers 1991).

1.3.2 Mission Statement

Building on the vision statement in 1.3.1 is the mission statement of WCNCB.

The mission is to provide geographically represented data and spatial analysis facilities to WCNCB personnel to facilitate the execution of individual functions (e.g. reserve management, permit issuing) and decision-making processes. The development of a geographical resource base that will aid and guide environmentally sustainable land use planning and development (Spooner 1992) outlines the mission.

1.3.3 Goals

The following is a list of the goals set out for the GIS Department:

- To integrate geographically referenced data pertinent to the execution of the WCNCB's mission from all available sources;
- To provide a tool for the visualisation of these data;
- To improve access to critical management information;
- To ensure continuity of management actions and information used for management;
- To improve the efficiency with which management and conservation activities are carried out; and
- To identify shortcomings in the conservation programme of the region

(De Klerk & Sutton 2000; Maclean 1994).

From these mission statements it can be derived that GIS has an integral role to play in providing solutions to the WCNCB manager. Infrastructure management forms only a part of the functions, but is essential for the other functions to operate properly (Bromley & Coulson 1989).

1.4 RESEARCH PROBLEM FORMULATION

An open day and a workshop were held on 17 November 1998 to introduce WCNCB personnel to GIS and to workshop critical issues such as the required functionality and implementation priorities for the WCNCB's GIS Department. At this workshop the following overarching need was identified:

Infrastructure Management Component:

Such a database should:

Allow for infrastructure inventory and provide a management facility indicating locality, type, age, and service details of equipment and infrastructure (De Klerk & Sutton 2000).

The research problem for this study is, therefore, the lack identified by the WCNCB managers of a system to help them plan, inspect and control their infrastructure. The purpose of this study is to develop a GIS for facility management by catering for the locality, type, age and service details of infrastructure. The decision was consequently made to develop an Infrastructure Management Support System (IMSS) to fulfil the needs and aims established (Guptill 1996; Heit & Shortreid 1991; Heit).

1.5 METHODOLOGY

The methodology of this research entailed collating the background information necessary to develop an IMSS for the WCNCB by doing a needs analysis that considers the current infrastructure inventory, the management procedure and the information technology available to the WCNCB. Decision support systems and GIS will be discussed as well as the involvement of GIS in spatial decision-making. This is described in Chapter 2. After this survey an explanation of the design and implementation procedures adopted is given in Chapter 3. This involves a description of the database and GIS design followed by the implementation description of the two. The IMSS to help the manager with the task of managing the infrastructure will then be described in Chapter 4, with figures and operational descriptions. To conclude, the system will be evaluated and suggestions for further development will

be made in Chapter 5. The information-gathering method adopted in this study is mostly qualitative. The main focus of this research is the development of a system to support the manager in the infrastructure management task by making use of GIS and evaluating the result of such a system. Figure 1.1 shows the basic steps followed in this study.

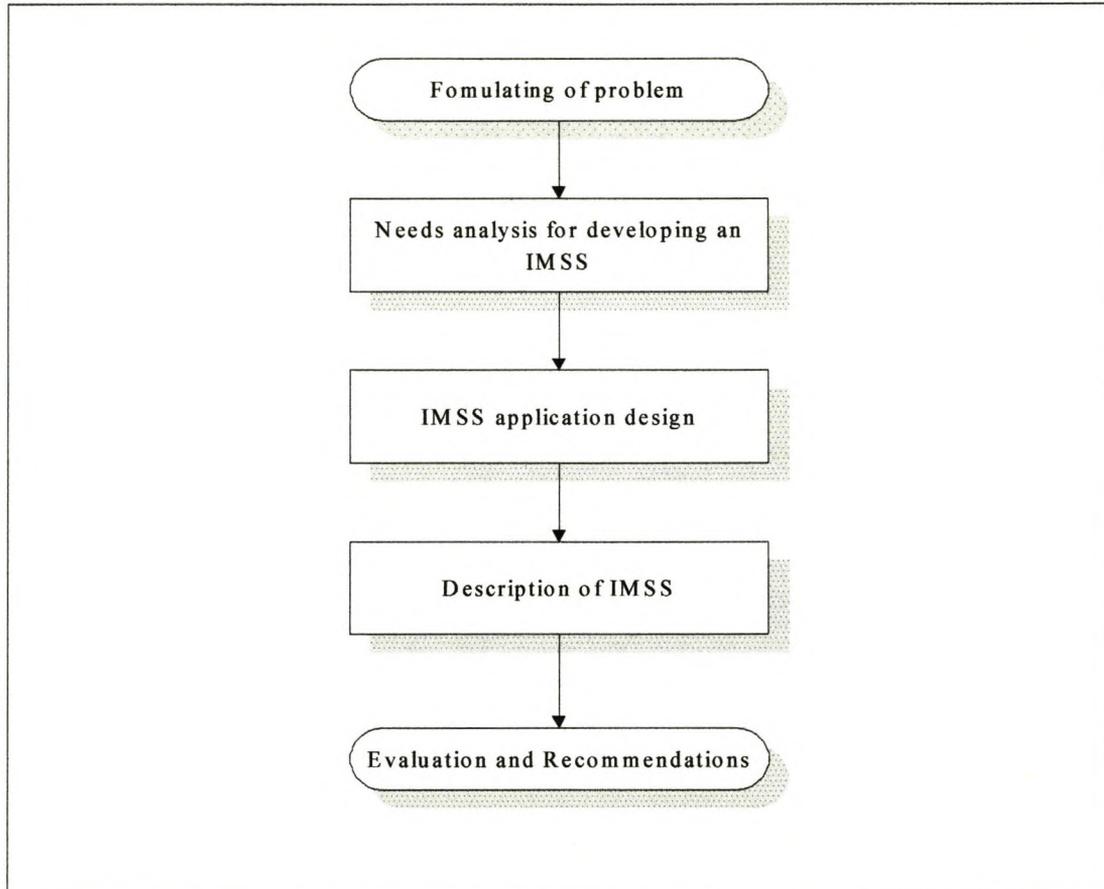


Figure 1.1 Research design outline

CHAPTER 2 NEEDS ASSESSMENT FOR WCNCB INFRASTRUCTURE MANAGEMENT

2.1 INFRASTRUCTURE INVENTORY OF THE WCNCB

For an Infrastructure Management Support System (IMSS) to support the activities of the WCNCB it is necessary to compile an inventory of the infrastructure used in parks under their jurisdiction. The infrastructure feature types in the inventory can basically be divided into two main categories, which can be further sub-divided into a number of subclasses.

The two main spatial feature type categories are points and lines, as derived from the spatial properties of the infrastructure. These two types are deemed sufficient to represent the infrastructure of the WCNCB as the small scale of existing input documents precludes defining aerial features as polygons. Clarke (1997) described a point as a geographical feature recorded as a location on a map with x-y co-ordinates. In this study an example would be the centroid of a building. Although a building has a footprint it is sufficient for this study to define it as a point. A line feature is described as a geographical feature recorded on a map as a sequence of locations tracing out a line with x-y co-ordinates. An example of where it will be used is to represent a road. These two spatial data types can, for the purpose of this study, accommodate all of WCNCB's infrastructural features, but provision will also be made for dealing with polygon features in the future.

The points feature category can be subdivided into the following subclasses:

Buildings:

- Office;
- Store General;
- Store Flammable;
- Ablution Block;
- Hut Hiking;
- Hut 4x4;
- Cottage;
- Shelter;

- Staff House;
- Info Centre;
- Boat House;
- Gate House; and
- Stable.

Activity spots:

- Picnic Site;
- Kiosk;
- Camp Site;
- Bush Camp;
- Inspection Quarter;
- Interpretation Site (*including archaeological sites*);
- Bird-hide;
- Swimming Pool;
- Angling Site;
- View Point;
- Abseil Area;
- Slipway; and
- Pond.

Services:

- Sewage Treatment Plant;
- Dump Site;
- Weather Station;
- Borrow Pit (*Hole from where gravel is taken to repair roads*);
- Parking Area;
- Repeater Site;
- Helipad;
- Loading Ramps;
- Vehicle Pit;
- Water Mill;
- Windmill;
- Bore Hole; and
- Signs.

Other

- Graveyard;
- Reservoir;
- Bridge;
- Gate; and
- Weir.

The following are subclasses for the lines category:

Roads:

- Road Management (all road – large or small – any condition);
 - Road 4x4 (used only by people paying for use of trail);
 - Road Tourism;
 - Road Public (roads with no access control and not maintained by Reserve management);
 - Trail Day;
 - Trail Mountain Bike;
 - Trail Hiking;
 - Trail Canoe;
 - Trail Pony; and
 - Foot Path (*short routes to other infrastructure – e.g. swimming pond*)
- (Paterson & Scullion 1990).

Fences

- Fence Game;
- Fence Jackal-proof;
- Fence Stock-proof; and
- Fence Electric.

Line

- Line Telephone; and
- Line Electrical

Other

- Fire-belt;
- Pipeline; and
- Furrows (drainage ditch).

These lists are not comprehensive and final and can be amended in order to cope with changing future requirements (Grigg 1988).

2.2 WCNCB MANAGERS

The managers have an integral role to play in the management of the nature reserves and their involvement in infrastructural management should therefore be investigated.

The following definitions from dictionaries give an overview of what a manager is:

- a) Someone who controls resources and expenditure (*Webster's Revised Unabridged Dictionary* 1998);
- b) One who handles, controls, or directs, especially:
 - i) One who directs a business or other enterprise;
 - ii) One who controls resources and expenditures, as of a household (*Webster's Revised Unabridged Dictionary* 2001);
- c) Someone who manages, especially someone in overall charge of a commercial enterprise, organisation, project, etc. (*Oxford Concise Dictionary* 1999).

2.2.1 Role and Tasks of the WCNCB Park Manager

In line with the way that the manager is described in the definitions above, the WCNCB park manager is in charge of a particular reserve that represents a certain biome. The manager's tasks vary between the different reserves and he or she is involved in different activities. The following list is therefore only a broad guideline to what a WCNCB park manager does. The main points of focus are:

- a) Environmental Impact Management: This requires of the manager to ensure that the reserve is managed in a sustainable way;
- b) Alien Management:

This involves the monitoring and clearing of alien species from the reserve and putting in place a buffer to prevent alien species from entering the reserve. The task further involves the monitoring of cleared areas and tracking the rehabilitation of indigenous vegetation (Hitt 1985);
- c) The management of the infrastructure: This is one of the major tasks of the park manager because of the role it plays in the management of the other tasks.

The following is a list of the activities involved in infrastructure management:

- Maintaining the standard of roads and building new roads where needed;

- Ensuring use of the correct fencing to allow only some animals through and maintaining the fence so that it keeps the targeted animals in;
- Maintenance of buildings on the reserve;
- Rehabilitation of land that is not in its original form, such as old gravel pits for example (Bromley 1990); and
- Clearing of land for fire-belts as a safety precaution.

Looking at the lists of infrastructure in 2.1, Activity Spots and Service-Related Infrastructure are critical aspects that a manager needs to keep track of. To highlight two aspects from the list:

Firstly: A Repeater Site is an area on the mountain where signal transmitter devices are kept; reserves often rent these sites out to companies (Mallawaarachchi, Walker, Young, Smyth, Lynch & Dudgeon 1996). The park provides a cleared area with an access road to it, manages the amount of equipment at the transmitter and ensures that the rent is paid;

Secondly: the signs in each reserve must be ordered and placed at strategic locations so that they are noticeable. They also need to be replaced after a fire or theft (Baldwin 1984).

d) Ecosystem management:

- The indigenous fauna and flora must be looked after to ensure their survival;
- Fires must be controlled and historical records of the fire scars filed;
- Erosion is always at work on a reserve and the manager must always be on the lookout to control it.

e) Administration:

- Financial management and staff management; and
- Environmental education, issuing of permits and tourist management are also part of the tasks of a nature conservation reserve manager

(Dale & McLaughlin 1989; pers. com. Shaw 2001; Laurens 2000; Sutton 2000; Sutton 2001; Turner 2001 De Klerk 2001 & De Klerk & Sutton 2000).

2.2.2 Computer Skill Levels of Managers

Ascertaining the computer skill levels of the managers is important in designing an infrastructure management system. In this section this is examined.

The skills levels of reserve managers vary but, with the exception of two managers, all the WCNCB managers have done at least one course in ArcView (Urenda 1992).

The following is a summary of the contents of the ArcView course:

- The nature of GIS;
- Types of GIS data and data-gathering techniques;
- Projections;
- Basics of Windows;
- The basics of ArcView:
 - Save
 - Adding data to view and reclassifying of the data (changing of symbolisation).
 - Navigation tools;
- Query data and using tables;
- Creating new data from existing data;
- Labelling data in a view;
- Importing data from spreadsheets;
- Layouts and printing your maps; and
- Analysing results.

For a full description of the course refer to Appendix 5.

From the investigation of the managers it could be determined that infrastructure management does in fact play an important role in the managers' tasks. There is a definite need for a system to assist them in infrastructure management and from the computer skill levels it is clear that it is appropriate to continue developing a computer application for this purpose.

2.3 DECISION SUPPORT SYSTEMS (DSS) AND GIS

DSS is a well-established area of information systems application. There is general agreement that these systems focus on specific decisions and on supporting rather than being a substitute for the user's decision-making processes (Angehrn & Lüthi 1990). Common to all definitions of DSS is a sense that these systems must support a particular type of decision-making process. This characteristic distinguishes DSS from general-purpose management information systems (MIS) (Mallach 1994). Generally interface, database and model components are usually required for full support of decision-making (Densham 1991).

Within the GIS field there is increasing interest in the use of GIS software to provide decision support. Many GIS-based systems are described as being DSS on the basis of the fact that the GIS assisted in the collection or organisation of data used by the decision-maker (Crossland, Wynne & Perkins 1995). While GIS applications may contain the information relevant to a decision, they are usually general-purpose systems, not focused on a particular decision. For those types of decision where the standard features of a GIS provide the information essential to the decision-maker, a GIS may indeed be a DSS. However, for the full range of problem areas where GIS techniques can make an important contribution, particular problem-related models are needed to fully support decisions. For these areas, at least, a standard GIS cannot be said to be a DSS, because such a system lacks the support that the use of customised models can provide. For this wide range of second-order uses of spatial data, additional processing or integration with non-spatial models is required to fully support the decision-maker. This will extend the present use of a GIS as a DSS to a situation where a GIS will be used to build a DSS (Armstrong & Densham 1990).

When a GIS contributes towards the creation of a DSS, it would most likely involve a spatial component. This leads to the development of a Spatial Decision Support System (SDSS). The following section discusses the role of GIS in SDSS.

2.4 THE IMPORTANCE OF GIS IN SDSS

In the period since DSS came to prominence there has been substantial growth in the significance of geographic information systems (GIS). This growth in GIS reflects the decreased cost of the required technology and the increasing availability of appropriate spatial data. Recent improvements in mainstream computer technologies have facilitated this increasingly widespread use of spatial data. These include inexpensive gigabyte-sized hard disks, large high-resolution colour monitors, graphics accelerators and CD-ROM storage. The use of GIS as a DSS generator can draw on new facilities for interaction between software, techniques such as dynamic data exchange (DDE), object linking (OLE) and open database connectivity (ODBC). These techniques will allow data to pass from the GIS to modelling software that can provide facilities not found in the GIS itself. Present software development trends suggest an object-oriented future in which small specialised applications, or applets, will be available for use as part of a larger package. In the PC environment the development of such small applications will be facilitated by the use of development tools such as Microsoft Visual Basic and Borland Delphi. The developments in GIS software since 1990 may allow the use of off-the-shelf software as the basis for an SDSS. An example of this type of software is the ArcView package from ESRI. As its name suggests, this software is primarily designed to allow the user to view and query spatial data. It is intended that the full ARC/INFO package will be required for some GIS operations. ArcView has its own macro language, Avenue, which can interact with SQL database servers, and the ability to use platform specific links with other software. Together with its ability to support spatial queries, these characteristics make ArcView a potential generator of many types of SDSS software. The same can be achieved with Mapbasic as a programming language in Mapinfo (Eom, Lee & Kim 1993).

GIS technology on its own can only make a partial contribution to decision support. Comprehensive decision support will require the effective integration of GIS and non-GIS techniques. This can be achieved by building systems using a GIS as a DSS generator. The building of SDSS applications from GIS has been facilitated by recent technical developments, both within GIS software and in programming tools generally (Jankowski 1995). The challenge for SDSS builders is to achieve an appropriate synthesis of modelling techniques and interface with database approaches,

drawn from the GIS and specialised domains, to provide effective decision support for these areas.

2.5 INFORMATION TECHNOLOGY AVAILABILITY

2.5.1 Computer Hardware

The hardware requirements of WCNCB can be divided into two main classes. The first class is intended for the use of professional GIS users. This class is used by people who would design and implement the GIS. There are about 10 computers in this class in the WCNCB. Their computers should be able to achieve an average response time of 2-4 seconds. According to De Klerk and Sutton (2000), the following are the minimum requirements for such computers:

- Motherboard: Pentium III 500Mhz;
- Memory: 256 Mb;
- Video card: 32 Meg AGP;
- Hard drive: 16 Gig;
- Monitor: 21 Inch;
- CD ROM device; and
- UPS device.

The second class of hardware requirements is that intended for the desktop GIS users. Their computers will be multi-use machines to be used for more than one task. The processing of GIS data would occupy up to 50% of their time spent on the computer (Masser 1992) and in this class there are about 20 computers in the WCNCB. An adequate response time would be an average 6 seconds or less. The minimum requirements recommended by De Klerk and Sutton for such a system are the following:

- Motherboard: Pentium II 400Mhz;
- Memory 128 Mb;
- Video card 4 Meg AGP;
- Hard drive 8 Gig;
- Monitor 17 inch;
- CD ROM device; and
- UPS device.

The hardware used to develop and test the IMSS described in Section 4 of this report consists of the following:

- Motherboard: Pentium 200Mhz;
- Memory 64Mb;
- Video card 2 Meg PCI;
- Hard drive 10 Gig;
- Monitor 14-inch;
- CD-ROM device;
- UPS device.

These minimum hardware requirements give an indication of the type of system that would run at a satisfactory speed and not of what is needed simply to run the software.

2.5.2 Networks

For infrastructure management to be successful for WCNCB the data should be integrated so that it can be controlled centrally. To establish this, a network that connects all the computers must be implemented. The first step would be the use of Local Area Networks (LANs) to connect the different computers on each reserve. See Figure 2.1 for a better understanding of LAN and Wide-Area Network (WAN) networking. The networking enables the IMSS database to be located in a central place from where the users can access it. This method of centralisation not only preserves data integrity, but it also controls redundancy, reduces management costs and allows other applications to run on the same data (Aronoff 1989).

LAN systems are currently operational within the WCNCB, but the WAN that is currently in place is still not suitable for a centralised database because the lines are presently not permanently online and the bandwidth is too low. The LAN would therefore be a better option to implement the IMSS. The next step would be to implement the WAN. This will be done via the local Internet service provider. With this implemented, the data can be stored in an even more centralised place, but until this is done, the LAN is a better option.

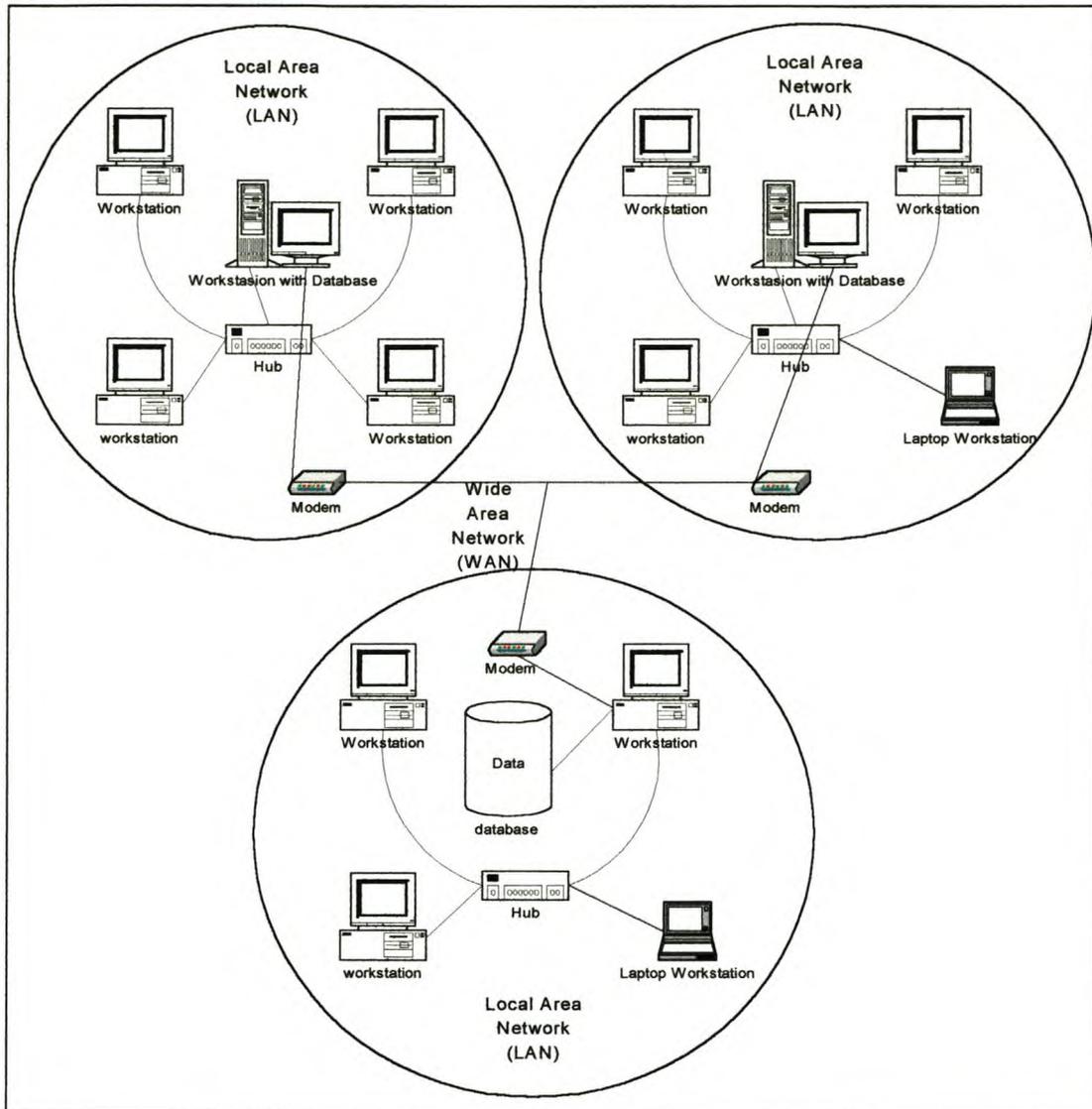


Figure 2.1 Diagram of network system showing LAN and WAN

2.5.3 Database

Aronoff (1989:151) describes a digital database as: "... a collection of information about things and their relationships to each other." Date (1990) compares it to a filing cabinet where the user can add new files, insert new data into existing files, retrieve data, and update, delete and remove existing files from the database.

The question then is why use a digital database system rather than a paper filing system? Date (1990) gives the following reasons:

- Compactness: there is no need for voluminous paper files;
- Speed: The machine can retrieve and change data faster than a human can;
- Less drudgery: Mechanical tasks are better done by machines; and
- Currency: Up-to-date and accurate information is available at any time.

There are four basic types of database structures: hierarchical structures, network systems, object-oriented and relational structures (Demers 1997). The problem with GIS data is that it consists of two data models, the one is the map or spatial data model and the other is the attribute data model. Clarke (1997:150) states that there must be a "...bridge or link to tie the attributes..." to the spatial data. The best way of doing this is to use a field with matching data in each data set so that the data sets can be related through the values. This is called a relational join. In GIS there is a predominance of the use of relational database management systems (Demers 1997). Some of the reasons for this are that the data are kept in reasonably simple tables and any number of tables can be related. A relational database will, therefore, be used in this application (Date 1990).

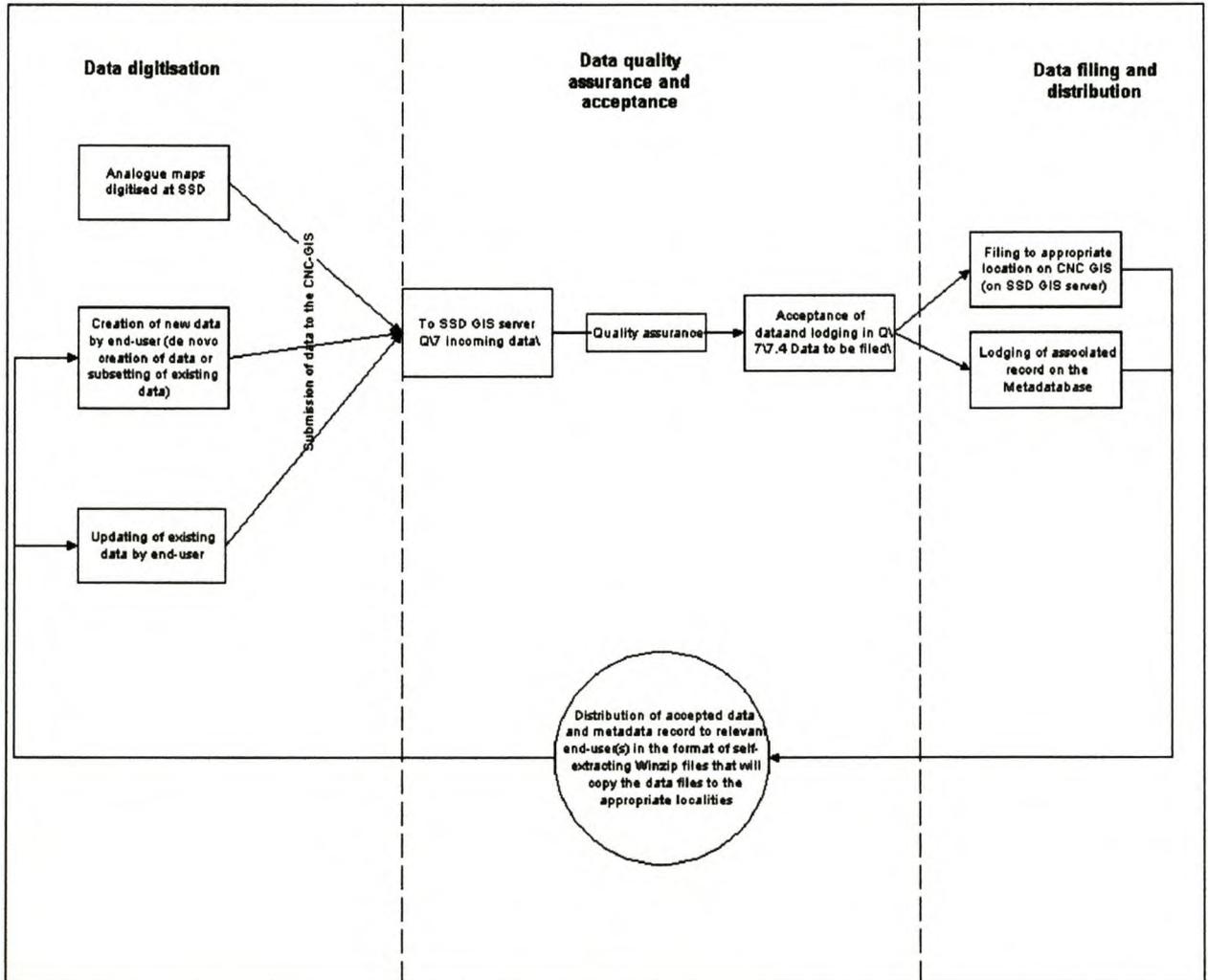
2.5.3.1 Management and Quality Control

The uniqueness of the GIS database stems from the fact that the data elements of the database are closely interrelated and thus need to be structured for easy integration and retrieval (Riley 1991). The GIS database also has to cater for the different needs of applications. In general, a proper database design/structure needs to ensure the following (Healey 1991):

- a) Flexibility in the design to adapt to the needs of different users;
- b) A controlled and standardised approach to data input and updating;
- c) A system of validation checks to maintain the integrity and consistency of the data elements;
- d) A level of security for minimising damage to the data; and
- e) Minimising redundancy in data storage.

At the WCNCB the goal is to centralise all data at Jonkershoek Scientific Services Division (SSD), but at the moment databases are also being compiled on each of the different reserves, mostly in the format of Microsoft (MS) Access databases. The plan, however, is to migrate the MS Access databases to a Sequential Query Language Server (SQLS) at a later stage. The management of these databases will mostly be done by the SSD. All new data will be sent to SSD for quality control purposes (De Klerk & Sutton 2000; Valenzuela 1991).

Quality control is the process of checking the data before releasing them. Figure 2.2 shows how the data are managed in order to check their quality. The quality control process is divided into three parts. In the data digitisation phase the data are converted to a digital form that is then sent to the SSD GIS server (data quality assurance and acceptance phase) for quality assurance checking. If accepted, the data get filed and distributed/redistributed (data filing and distribution phase) in the form of a zip-file. The process can then be repeated if any changes are made to the data.



(De Klerk & Sutton 2000:22)

Figure 2.2 Data quality control flow from the source to the WCNCB GIS and to the end user.

Given the present situation whereby most of the databases at the WCNCB are using MS Access, it would be best to develop a relational database for the IMSS in the windows environment, based on the format of the Access Database Management System (DBMS) (Blau 1993).

2.5.3.2 Data Needs

For the construction of a database data are required, but to determine the scope and nature of the required data the following must be kept in mind: the structure of the institution, the aims of the WCNCB, the amount of effort that can be put into the management and the amount of capital available for this project.

Hudson, Haas & Uddin (1997) state that an infrastructure database should consist of the following:

- Physical description of infrastructure;
- Access to condition data, usage and history;
- Construction and metrical data;
- Maintenance history.

With the above-mentioned points as well as the description of Hudson, Haas and Uddin (1997) kept in mind, the following data needs for the WCNCB were identified (Sutton and Turner 2001):

Condition of Features

The aim would be to give the manager an idea of what the condition of specific features is. It was decided to use three classes, namely Excellent, Good and Bad. The reason for using only three classes is to try and avoid subjective selection between too many subtly varying classes. These measurement levels must be implemented for all the features.

Time Stamping

The time that the information was captured must be stored, so that upgrades or changes can be monitored by means of the database management system.

Name of Assessor

This information must be captured to help the manager monitor who captured the data.

Geometry

For the manager to make decisions regarding the geometry of the infrastructure, it is necessary to store spatial information about the captured features. It would then be

possible for the GIS manager to produce a map of the layout with the appropriate background to give perspective to the spatial relationships.

Features and properties

The list in Section 2.1 is an indication of the *features* to be captured. The *properties* are applicable to each *feature* and should be chosen by each reserve manager to suit his/her task. For example, the manager has a building *feature* and wants to know the construction material, paint colour, roof covering and the area it covers. These data will then be stored as the *properties* of that building. For this system to work the application must be developed in such a way that it is possible to add new *properties* and *features* to the database when required (Uddin 1995).

2.5.4 Software

The software serves as the interface between the users, hardware and data. It allows the user to access data by giving instructions to the hardware. There are three important components of software for database systems: operating systems, database management software and application programs. Generally operating systems and database management software are commercially available, while application programs are often unique to a particular application (Dangermond 1983).

2.5.4.1 Operating system

The WCNCB mostly uses Windows 98, NT and 2000. Some servers utilise Linux as their operating systems. The goal would be to get all of the WCNCB's computers on the NT operating system platform, because of its stability and compatibility with the software they have purchased. At present, however, many computers are running under the Windows 98 operating system.

2.5.4.2 GIS software

The WCNCB prefers to use the Environmental Science Research Institute's (ESRI) products. ArcView 3.1 is the main GIS used by the WCNCB, but Arc/Info and ERMapper are also used. ArcView was chosen because it is user-friendly and has a wide international and national user base (Marble 1990). The program provides a programmable interface, enabling the development of applications with the

programming language Avenue, which suits the needs of the WCNCB. It also has the ability to add new modules such as Spatial Analyst. The ArcView data format is widely used, which makes it easy to exchange data between different users and organisations (Buxton 1991).

2.5.4.3 Database Management System (DBMS)

The DBMS systems available at the WCNCB are Postgres, MS Access 2, 1997 and 2000 and Microsoft SQL server 7. The one most widely used is Access 97. Access 2 has almost been phased out and Access 2000 is not widely used yet. Access has a programmable interface, which makes it possible to create database applications with the object-oriented programming language Visual Basic (VB). The plan is to migrate the Access DBMS to the SQL server at a later stage, when the WCNCB Database Management Systems are ready. The following chapter describes the infrastructure management support system design.

CHAPTER 3 DESIGN OF THE INFRASTRUCTURE MANAGEMENT SUPPORT SYSTEM

3.1 PRELIMINARY PLANNING

In commencing the design of the application, the background information indicated in the previous section (2) was studied. On the basis of this background information, the following were found to be the most appropriate software to use for the development of the IMSS: Windows 98, Microsoft Access 97 relational database and ArcView 3.1. Initially a Local Area Network with a centralised database for each reserve will be created. The first step will be to install all the software on a separate development computer. The WCNCB has standardised storage space locations.

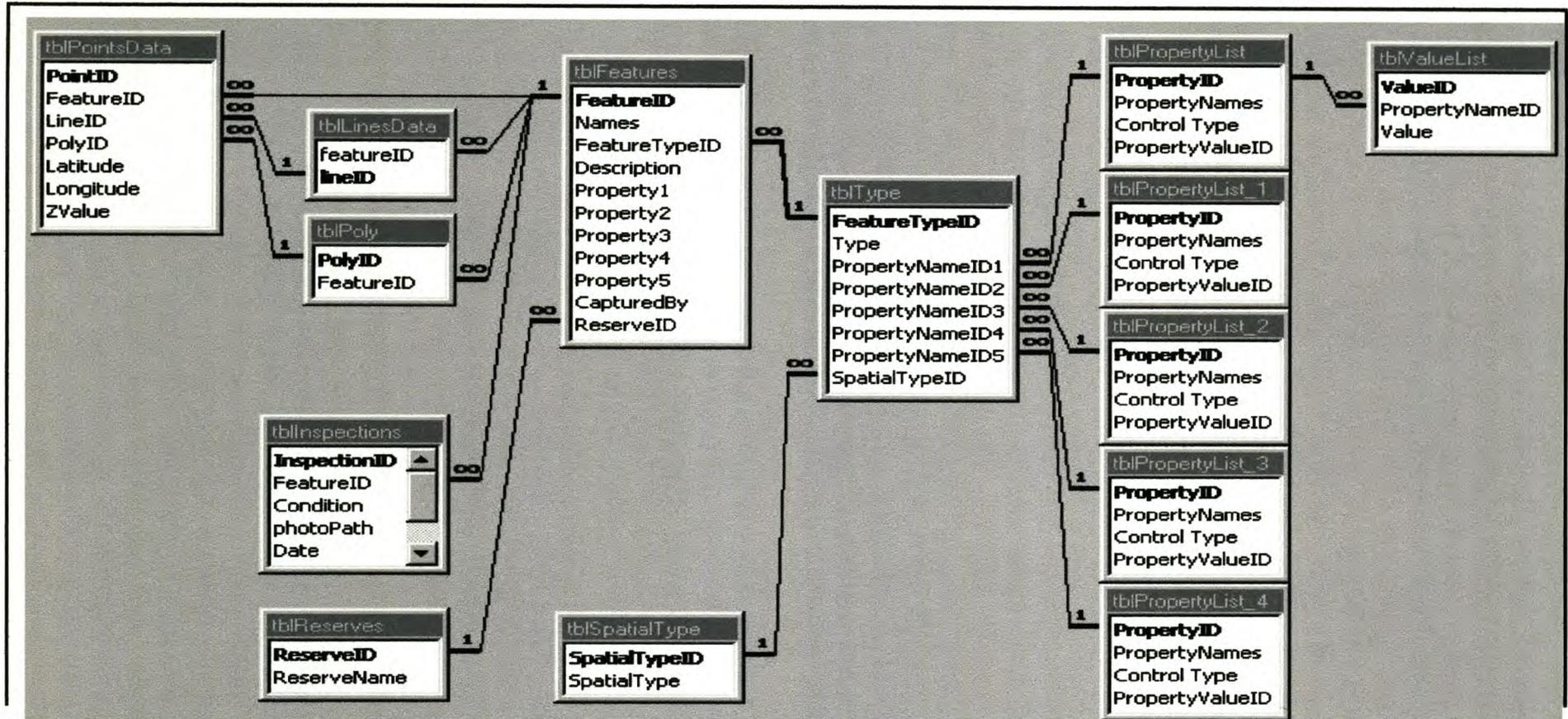
3.2 DESIGN

For the development of the different parts of the Infrastructure Management Support System a sequence of steps had to be decided on. As the DBMS is the connection between the hardware and software that manages the data, the DBMS is the basis on which the rest of the application will build; for this reason the design of the relational database should take top priority. Thereafter the database management application should be developed, followed by the GIS extension (Clarke 1997).

3.2.1 Relational Database Design

Figure 3.1 shows the IMSS database relationships developed for this program. Within this diagram *tblFeatures* is the main table from which all relations to other tables are built. In the first place the *tblPointsData* table contains the spatial data. This table was designed to store spatial/geometrical information on Points, Lines and Polygons. All Lines and Polygons are made up of a sequence of points. All the co-ordinate data are therefore stored in the *tblPointsData* table with the point feature data. Each spatial data type has its own ID that identifies it as follows: 'FeatureID' for points, 'LineID' for Lines and 'PolyID' for Polygons. The problem with this is that it is not possible to make a direct connection from the features listed in the table called *tblFeatures* to the data points listed in the *tblPointsData* table, because there are 3 different IDs in the *tblPointsData* table to connect to. Relational databases are not designed to make more than one connection between two tables. To solve this problem, two new linker tables

were created: *tblLinesData* and *tblPoly*. It is now possible to make a connection between the *tblFeatures* table and the new tables, from where a connection can be made to the *tblPointsData* table (refer to Figure 3.1).



Where a connecting line is marked with an '∞' symbol it indicates that more than one record from that table can be connected to one record in another table, as indicated by the line. A '1' symbol indicates the reverse: one record from the marked table can be connected to more than one record in another table. Therefore '∞' indicates a many to one relationship while '1' indicates a one to many relationship.

Figure 3.1 IMSS Relational database relationships

The next table in the diagram (Figure 3.1) to be discussed is *tblInspections*. This table is related to the *tblFeatures* table via the field 'FeatureID', which is present in both tables. The information in this table relates to site inspections. It consists of a field for condition rating, photo directory paths so that pictures can be referenced, and a date and time to record time stamps for information about when the data on infrastructure were captured (Lansdale & Ormerod 1995).

The *tblReserves* table has the names of all the reserves under the jurisdiction of the WCNCB stored in it. The reason that this information is stored in a table of its own is to normalise the data in the relational database. Normalisation is the process of splitting up tables to avoid storing the same information more than once. See the examples in Figure 3.2(a) and (b).

tblReserves	
ID	ReserveName
1	Blouberg
2	Cecelia
3	Cederberg
4	Cederberg
5	Cederberg
6	Cederberg
7	Cederberg
8	Cederberg

Figure 3.2 a) A table before normalisation.

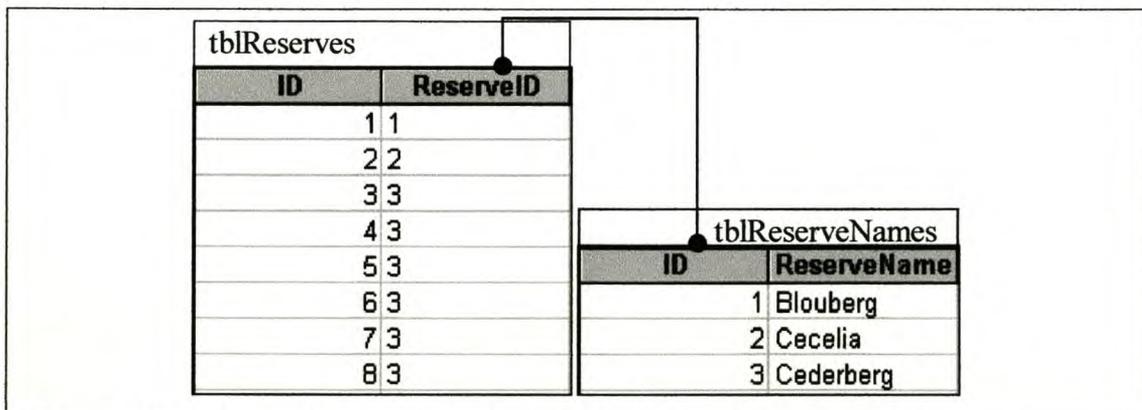


Figure 3.2 b) Illustration of the table in a) after normalisation.

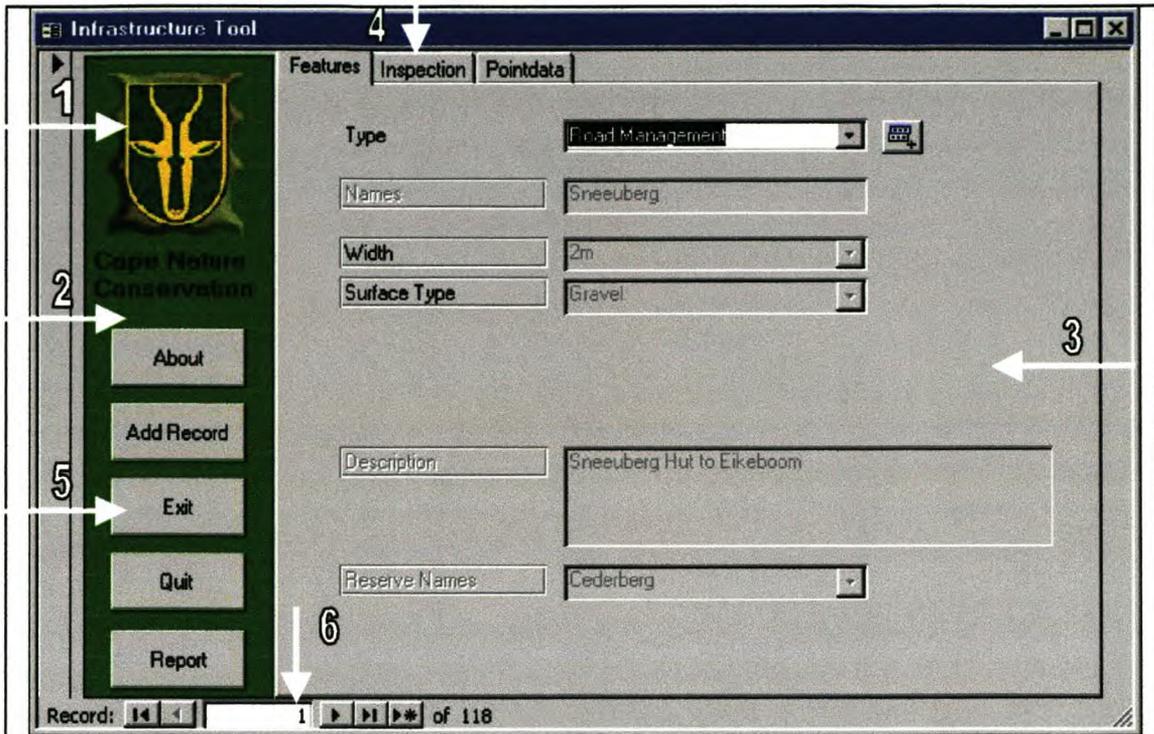
The *tblType* table is related to the *tblFeatures* table with a one-to-many relationship (see Figure 3.1). This means that there can be more than one value in *tblFeatures* for each value in *tblType*. Each feature belongs to a type that can be shared with others. To normalise this an ID value is stored in *tblFeatures* so that the 'Type' value could

only be stored once in *tblType*. It is therefore easy to add a new value in the ‘Type’ field of *tblType*, if the user wants to include a new infrastructure item.

To normalise the database even more, ID values are stored in the *tblType* table’s property ID 1- 5, which represents the property names stored in the *tblPropertyList* table. To help maintain data integrity, the table *tblValueList* serves as a lookup table and all values allowed for each of the property names are stored in the *tblValuelist* table. The user can then pick a value and this value will be stored in the *tblFeatures* table.

3.2.2 Graphical User Interface (GUI)

A GUI is the visible part of a program with which the user interacts. See Figure 3.3 for an example of a GUI. The design of a GUI is particularly significant because of its visibility. For this reason the GUI must be simple, logical and easy to use. The WCNCB already has some applications developed for their purposes (Hentzen 1999). To make it easier for users to adapt to new applications, the design will follow the existing application standards. Figure 3.3 is an example of what the GUI standards are:



1. The bontebok emblem in the left top corner;
2. The green background on the left side;
3. The standard windows grey for the rest;
4. The use of tabs to compact the GUI;
5. The command buttons on the left and not at the bottom like most programs;
6. The navigational bar at the bottom.

Figure 3.3 Example of WCNCB GUI standard

3.2.2.1 Infrastructure Database Application

For a system to be functional, its user options must be confined to those needed for the operation of the system. The user does not need to see the low-level operations.

To start off the program the user needs to know what options the program provides and the assurance that he or she is at the correct level in the menu system. To achieve this, the name of the application should be displayed with a list of the components from which the user can choose (Apers, Blanken & Houtsma 1997). Figure 3.4 shows the start-off point of the infrastructure database application.



Figure 3.4 Database start form

The buttons on the start form are placed in a way that provides easy access to the rest of the application. It is kept simple to avoid confusion. This form is followed by three forms that open in the display when the corresponding button is pressed. These are: the Data Entry tool form, the New Feature tool form and the New Properties tool form. Each of the forms is discussed in turn in the next section.

i) Data Entry Form

The Data Entry Form must cover a large range of data items and must therefore be designed in a compact, easy to use form. Figures 3.5, 3.6 and 3.7 show the layout of this form. The buttons on the left remain the same for all the tabs; this makes it easier for the user to learn the operation of the application (Batty 1992). The application is designed in such a way that the forms flow logically from the one to the next. The tabs are arranged to indicate progression from left to right. The names of all the controls are placed to the left of the input controls and the location of the command buttons are placed to the right of the input box that they relate to. Because of the large number of variables stored in the database, it was decided to use the tab method to divide the data into more manageable parts. If this was not done, all the attribute items would have been listed one below the other. This would not have been practical because the user would then have to scroll up and down a long list to find the input controls. These tabs were divided into three categories, namely Features, Inspections and Point Data (Peeters & Jarosz 1996).

The *Features* tab contains the unique name for each item, its properties (up to five items), a description of the item and the name of the reserve in which it is situated. The tab is designed to grey out all the input controls if the record already contains data. This makes it impossible to change data for a certain record until a change is made to the data type or a new record is added so that the user would not lose information if he or she overwrites old information (National Science Foundation 1995). This makes it more user-robust.

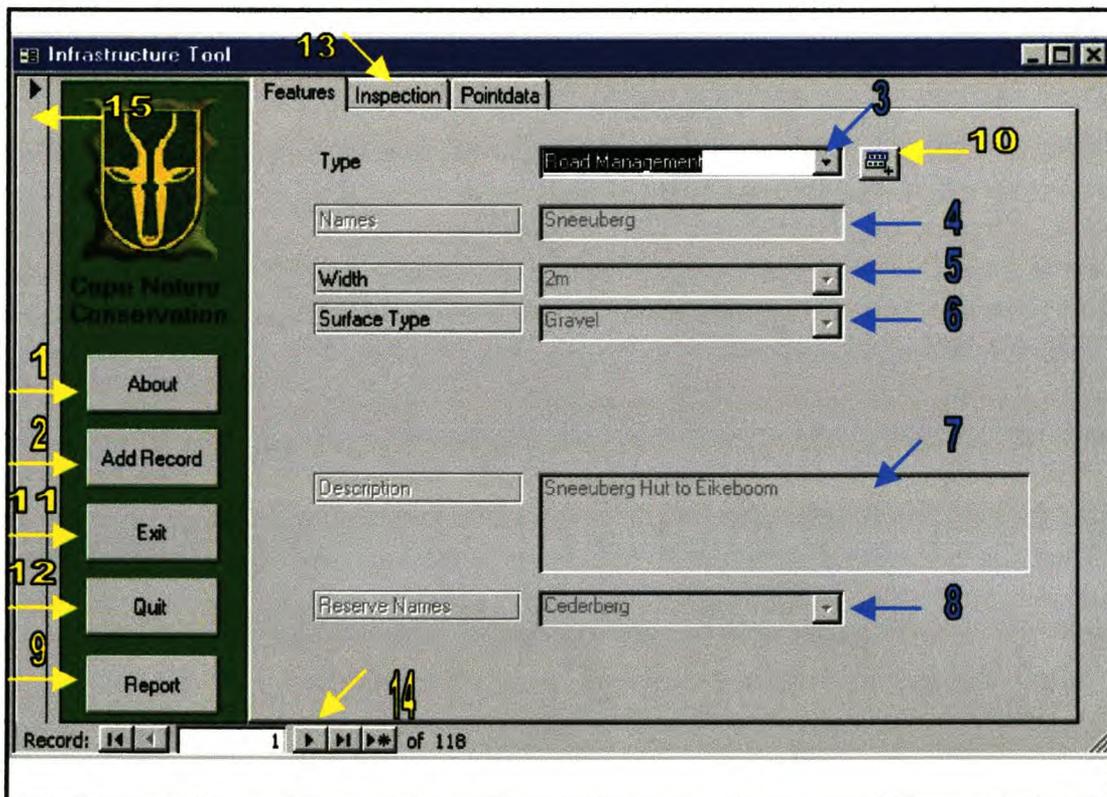


Figure 3.5 Example of Infrastructure Tool with Feature selected

The next tab is the *Inspections* tab (Figure 3.6). This tab contains the data related to a site inspection. The first input control '*condition*' is in the form of a combo box. The name is derived from the combination of a textbox and a list box. In this control the user can select an item from a list. The method is appropriate in this case because the values are available in the database to populate the combo box. The *date* and *current time* values are automatically provided as the current date and time (Freeman 1997).

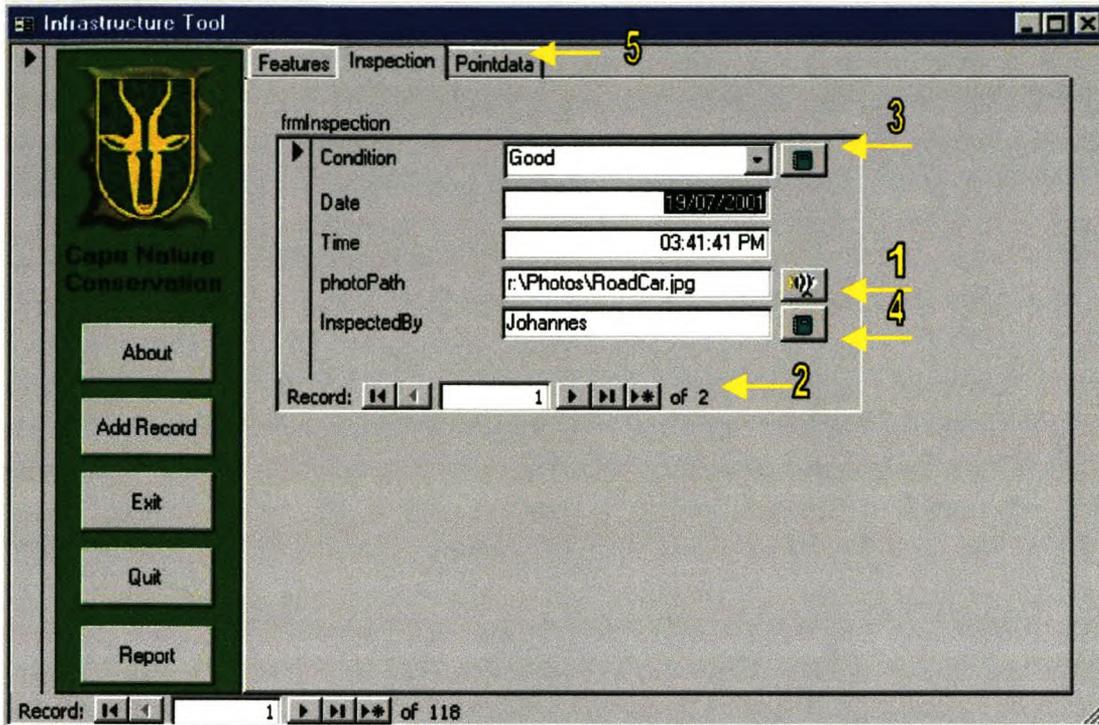


Figure 3.6 Example of Infrastructure tool with Inspection tab selected.

A photo path can be provided and can be viewed by pressing the command button  directly to the right of the text box. Figure 3.7 illustrates how the photo would be displayed. There is an *Exit* button on the image form to return to the *Data Entry form* (Laurel 1990).



Figure 3.7 Picture display form

The *Pointdata tab* contains the spatial points controls to input co-ordinates for each feature. Only the fields needed are shown to the user and the ID fields used in the background by the application are not shown. The WCNCB does not need the three-dimensional Z-value that is provided at this stage, but they would be able to use it in future (see Figure 3.8). This part of the application is not designed to input large amounts of data for lines and polygons; it is suitable to edit data and input point data. The aim is to input the bulk of the data from digitised data in ArcView or from the Cybertracker system currently being implemented by the WCNCB (a system that gathers data from fieldwork with a palm pilot/GPS combination and then stores the data in the database) (Magellan 1997; Getz, Litwin & Gilbert 1999).

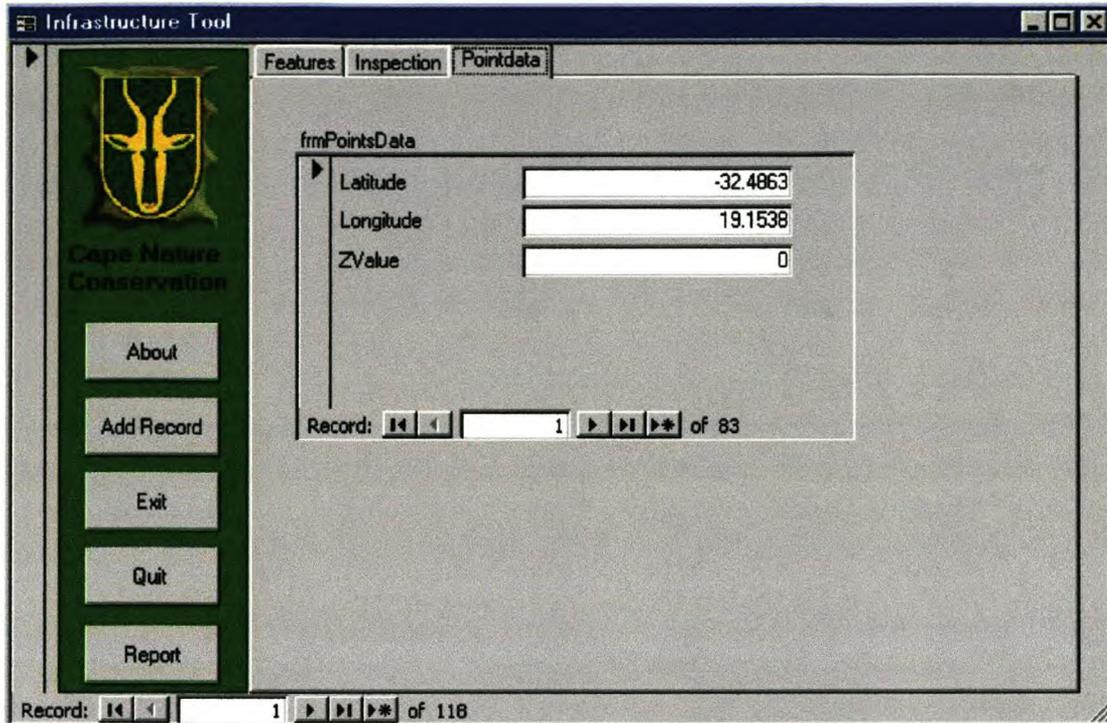


Figure 3.8 Example of the Infrastructure Tool with Pointdata tab selected.

ii) New Feature Tool

The second option on the start form is the *New Feature tool* form (see Figure 3.9). This form can also be accessed from the Data Entry form by pressing the New Features tool button  (Figure 3.5). This form was designed to add flexibility to the application. There are 68 standard data items (see section 2.1 for details of this list) within the system, but this form makes it possible to add more fields if required. All of the form's input devices are combo boxes. This helps to protect the data integrity (Engelken 1993). The first box on the *New Features* tool is called 'Name' and it contains all the features that already exist within the system. The application as a whole is designed to manage a maximum of five properties. In this form the number is kept to five by providing space for five properties only.

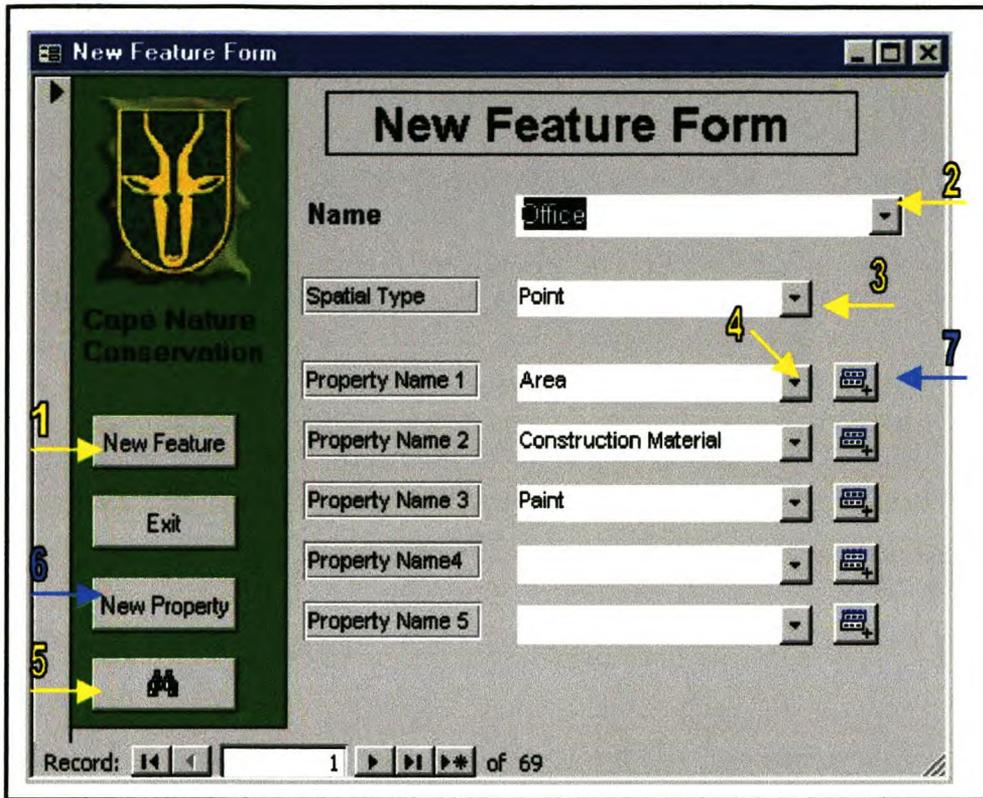


Figure 3.9 Example of a New Feature Form

iii) New Property Tool

Next on the start form is the *New Property tool* form (Figure 3.10).

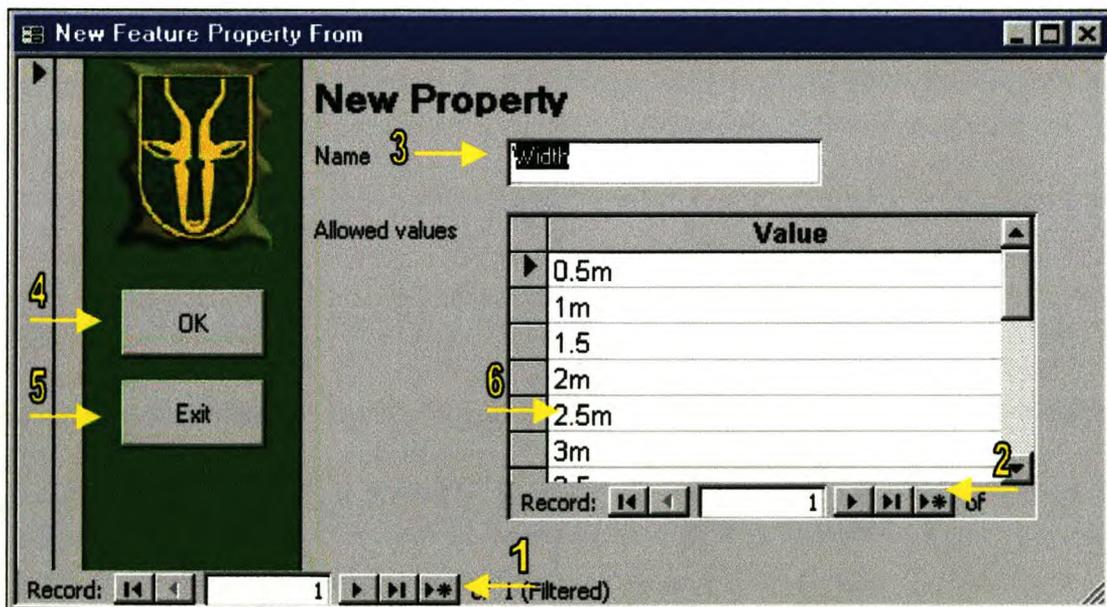
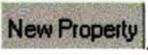


Figure 3.10 Example of New Property adding form

This form can also be accessed from the New Feature Form (figure 3.9) by pressing either the New Property tool button , or the New Property button. 

For the allowed values the designer used a list type textbox (see 6 in Figure 3.10). This approach allows the user to access any one of the values in the list with one click and edit them.

3.2.2.1 GIS extension

The GIS part of the new application is implemented in the form of an extension. The user can therefore remove or add the functionality to ArcView as required. The initial change to ArcView is small, in the form of one menu item added to the interface (De Man 1990; Razavi 1995). This menu is called the '*Infrastructure*' menu and is situated between the Graphics and Window menu (see Figure 3.11).



Figure 3.11 The ArcView menu bar after the IMSS extension was loaded.

The menu consists of four items. These items have been designed to provide access to the whole extent of the application. All the items are provided with shortcut keys, indicated by underlining the first letter of each phrase (ESRI 1996) (see Figure 3.12).

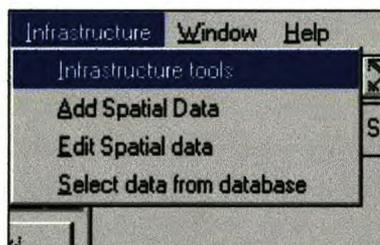


Figure 3.12 The layout of the Infrastructure item's dropdown list.

The first item in the dropdown list is a tool with a graphical interface to provide an alternative way to get to all the parts of the program without using the menu (see Figure 3.13). A yellow bar with the name of the tool appears at the top of all forms used in ArcView (ESRI 1995). A quit button is provided to close the *Infra tool*. This form consists of three command buttons to activate the three different sub-modules of the application.

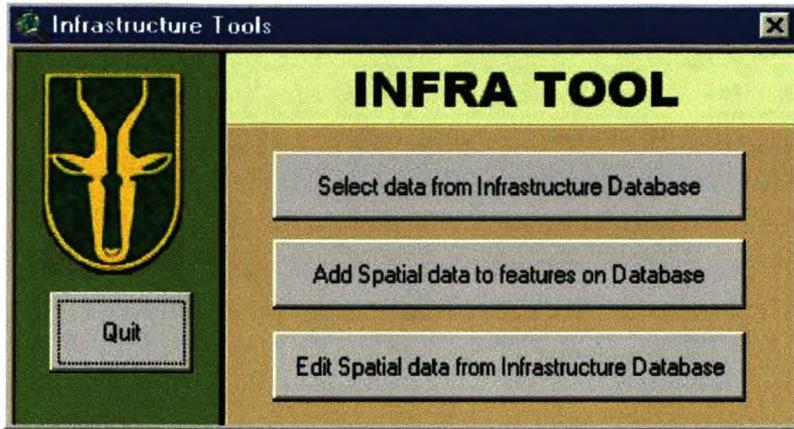


Figure 3.13 The layout of the Infra tool form.

The three forms connected to the command buttons are *Infrastructure Data Select*, *Add Spatial Data to Points* and *Editing tool*. These are each discussed in the following sections.

i) Infrastructure Data Select

The first button activates the form that selects the data from the Access database to be displayed spatially in ArcView (see Figure 3.16). The form consists almost entirely of drop-down list combo boxes populated with data from the database to allow the user to operate simply by selecting from a list. Only the year value needs to be typed (ESRI 1997). This gives the application flexibility in the sense that no changes need to be made to this form to keep it updated, the only changes required would be to the database itself. This form is designed to make a very specific query from the database, and allows for the user to select one record from the database to be displayed in an ArcView view (Berry 1993).

Figure 3.14 An example of the *Infrastructure Data Select* Tool Dialog.

The green diamond command button in Figure 3.14 opens up another form that enables the user to refine the selection even further in terms of the properties of a feature (see Figure 3.15). This form has been designed to display only the properties associated with a feature, while the rest of the properties are hidden from the user. From the example in Figure 3.15 it is clear that the Feature: Hut Hiking has only two properties and the rest are hidden. These drop-down list combo boxes also get their data from the database. An *OK* and *Cancel* buttons were added to the form. The *OK* button has the same function as the one in Figure 3.14, except that it includes the data in the properties form when executing the query. The *Cancel* button gives the user the option to exclude the data in the query, if he or she has already changed some of the values in the drop-down list combo boxes (Marcus 1992).

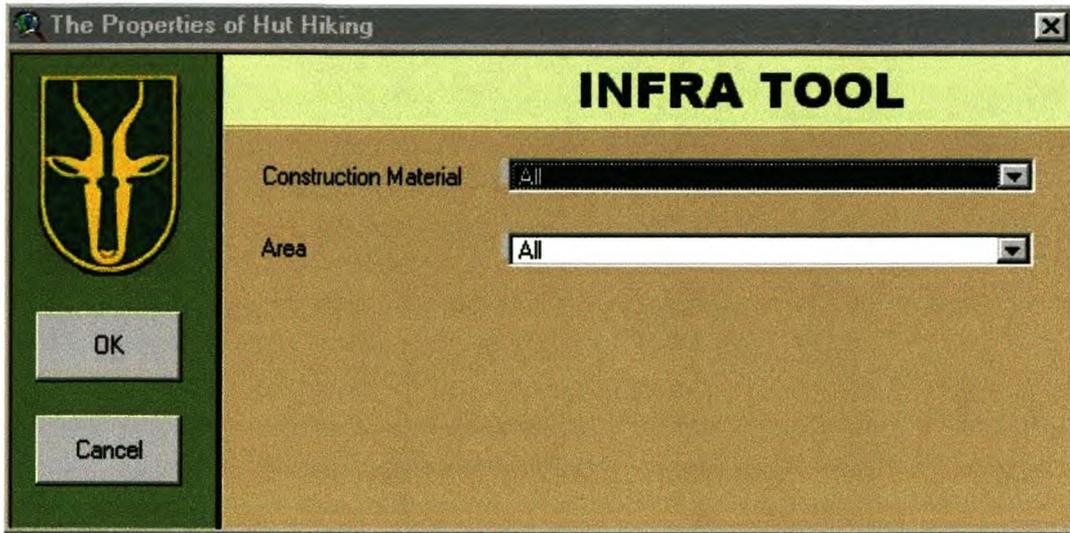


Figure 3.15 An example of the *Property Data Select* dialog addition.

ii) Add Spatial Data to Points

The next button of the ArcView extension is the part where the user can add spatial data to the database. Figure 3.16 illustrates the functionality. This tool has been designed to look up the values in the database without spatial data and display them in the combo box. Displays of the selected x-y co-ordinates are provided in textboxes. A command button with a flag on was chosen to indicate a position, symbolising the planting of a flag (ESRI 1996). All the forms have been given descriptive names along the top bar of the frame.

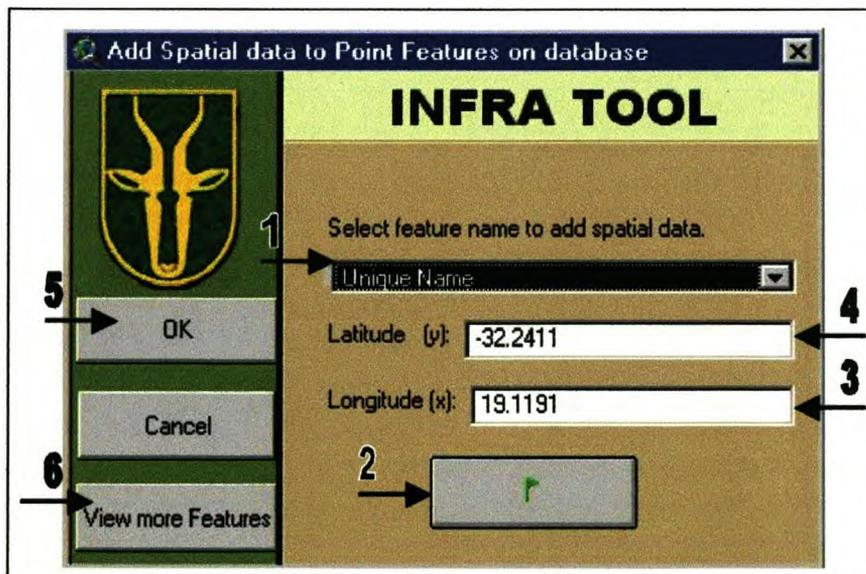


Figure 3.16 An example of the Spatial Data Adding tool.

iii) Editing tool

The next item on the form is the tool to edit data on a View (see Figure 3.17). This form will be used in conjunction with a View and must therefore be in a compact format so that it does not block the View while the user is editing the information (Poolman 1992). The themes in the combo box are automatically populated with the themes from the database. The user can therefore have any themes in the View and this tool will select only themes from the database to add to the list (Burrough 1988). Furthermore, the tool checks the spatial feature types of the selected theme and it hides those tools that are not applicable to the type of spatial feature. The editing tool only allows the user to edit one theme at a time. All the basic navigational, select and editing tools in the form have been selected from the ArcView toolbar, to ensure that they are not unfamiliar to the user (FAO 1996).

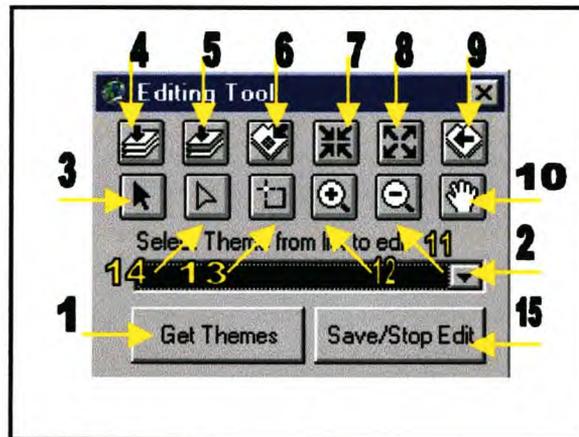


Figure 3.17 An example of the Editing Tool.

3.2.3 Information flow

The flow of information within the IMSS consists of three main parts: the ArcView environment, the Microsoft Access environment and the connection between the two parts. The connection has been made with Open Database Connectivity (ODBC). This is a function provided by Microsoft Windows to conduct data flow between programs. With this connection Sequential Query Language (SQL) statements can be sent from one application to query another application. This issue will be covered in more detail in the implementation section of this thesis.

The following two flowcharts show the flow of information within ArcView. The first diagram (Figure 3.18) shows the flow of data from the Access database from the time

that it enters ArcView to be edited until it is sent back to the database. The second diagram (Figure 3.19) describes the flow of data when spatial data are added to the ArcView database.

The flow of information in Access is rather simple because of the fact that Access is mainly an information input device. The software, in this case the Access forms, connects to the database that interacts with the hardware to store the data. The forms for the input of information are therefore all linked to the fields of the database in a logical manner. Refer to the forms in Figure 3.5, 3.9 and 3.10 and the database relationships in Figure 3.1 to see how the names of the input boxes match the names of the database fields. Also notice that the database names are in Hungarian notation, which means the phrases are written as one word with capital letters indicating the start of a new word.

The property values of each property are stored in the *tblFeatures* table in *Property1-5*. The spatial data flow is complicated in this instance, but Figure 3.20 clarifies this flow. The program first determines the kind of spatial geometry and then selects the corresponding script to run. For the point geometry type the x-y co-ordinates as well as the featureID get inserted directly into the *tblPointsData* table. For the Line and Polygon geometry the *FeatureID* gets inserted into separate tables called *tblLineData* and *tblPolyData* respectively and together with the new IDs created within them the X-Y co-ordinates are inserted into the *tblPointsData* Table. The reason for inserting the IDs in a separate table is that a many-to-many relationship cannot be made with MS Access and this is a solution to that problem.

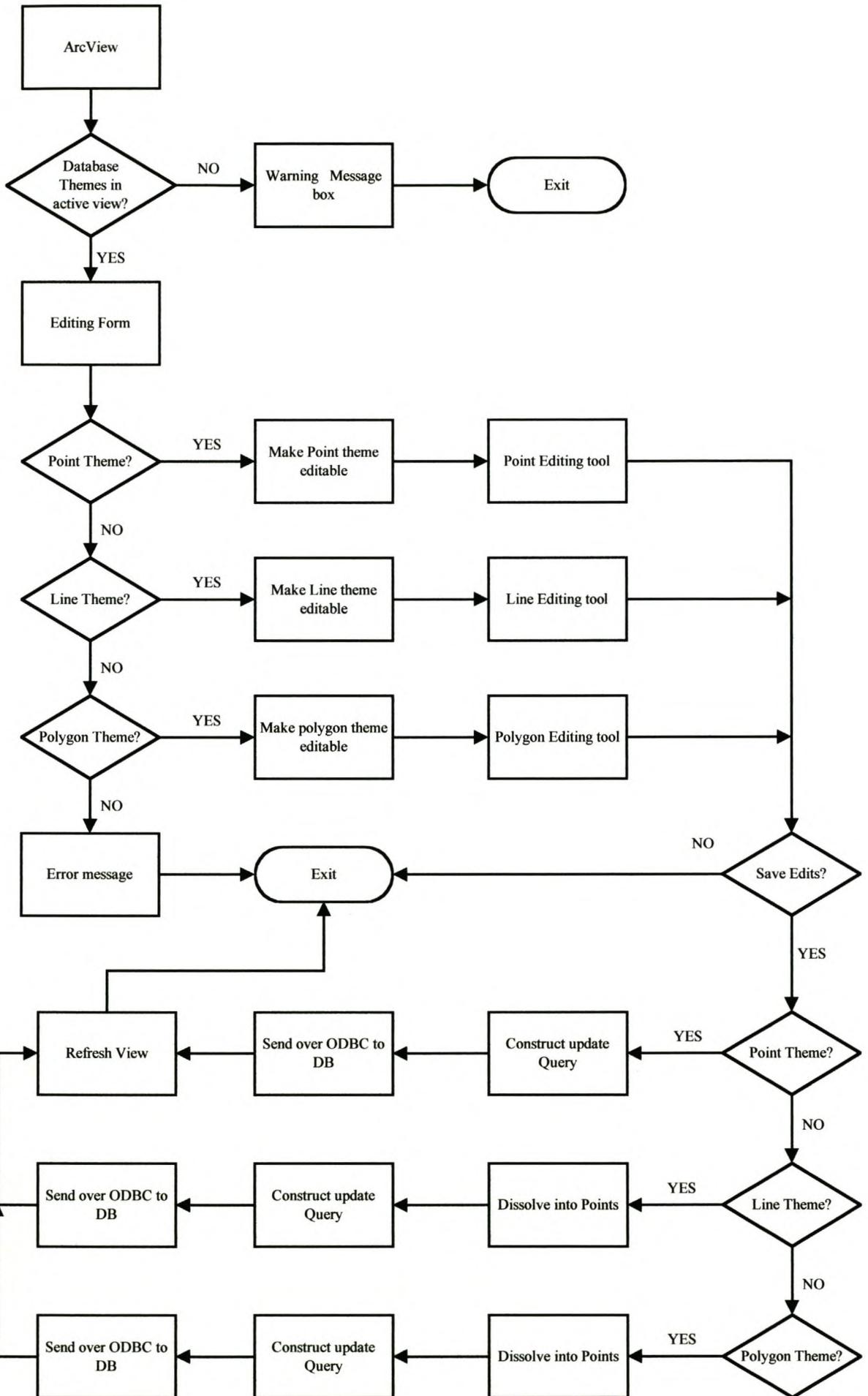


Figure 3.18 ArcView Data Editing Flowchart

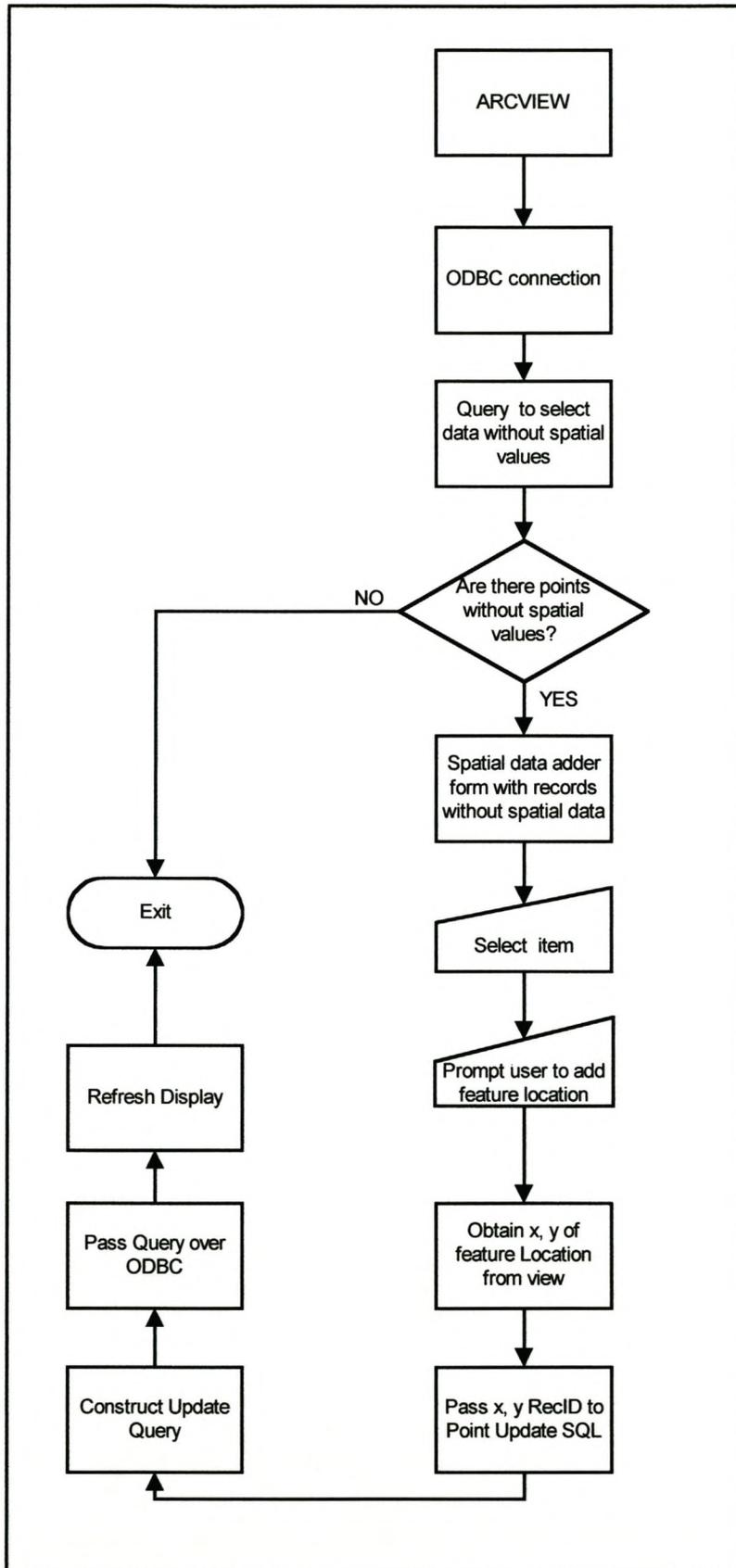


Figure 3.19 ArcView Spatial Data Adding Flowchart

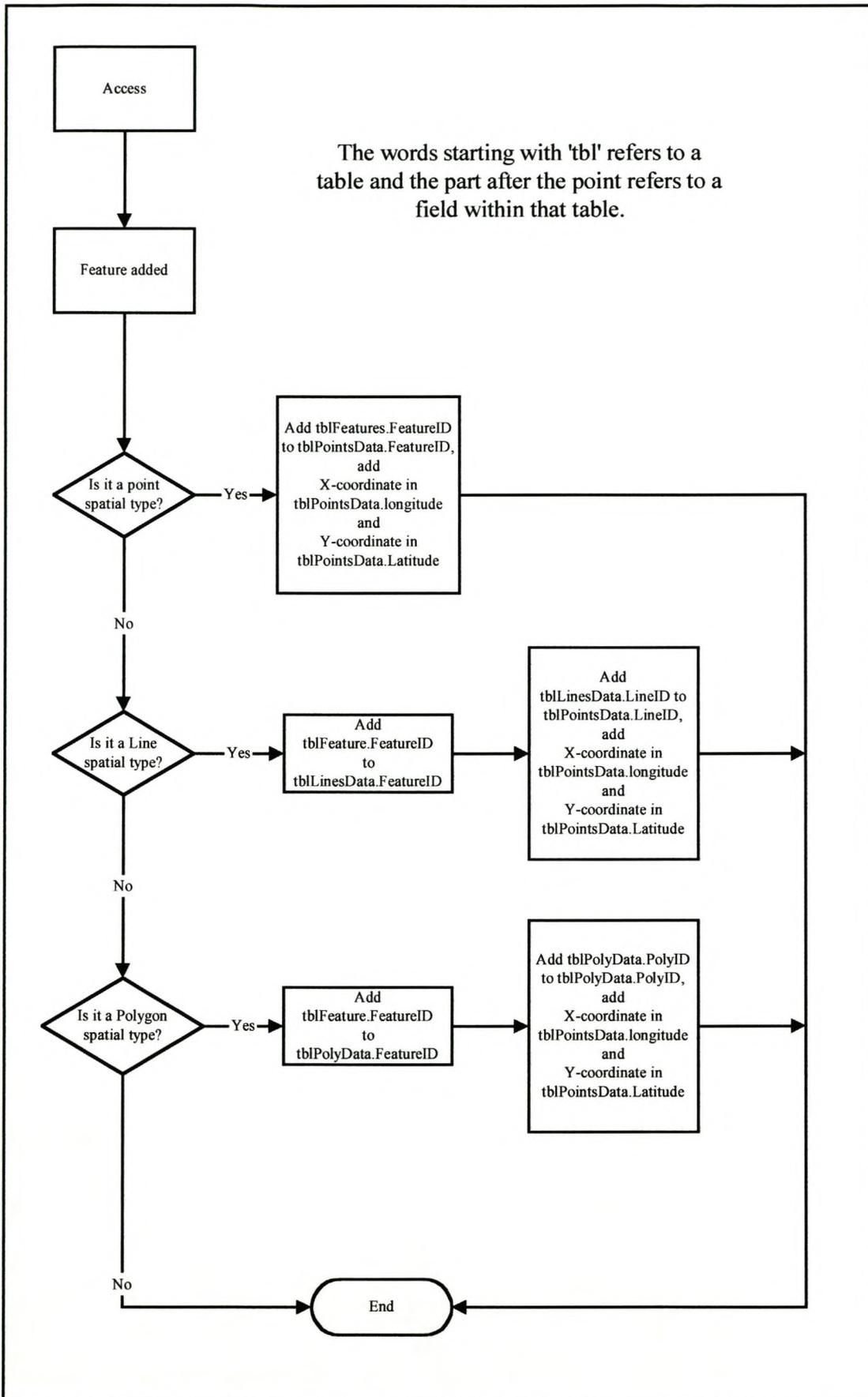


Figure 3.20 Flowchart of Spatial Data in Microsoft Access

3.3 IMPLEMENTATION

After the IMSS has been designed the implementation phase follows. The following section covers the different steps of the implementation namely: programming, integration and testing.

3.3.1 Programming

This section documents the actual programming process. As is often the case in programming, the designer used pseudo code to write the program before writing the real code. The process followed was first to write a comment and then the code that executes the comment description. Due to their size and extent the code and the descriptions used in the IMSS have been attached as Appendix 1 and 2 (not including system scripts) (Amir & Warwick 1997).

The description of the programming process will be done in two parts; the first part will focus on the ArcView section and the second on the Microsoft Access section.

3.3.1.1 ArcView Programming

The object-oriented programming language used in ArcView 3.1 is Avenue. The following discussion on programming was developed using this language.

An item in the menu drop-down list executes a function with the programming script (a script is a set of codes) connected to it. The first item in the list is 'Infrastructure Tool' (refer to Figure 3.12); the code connected to it is *CNCINFCalldlgStart*. This code calls the dialog box *dlgStart* (refer to Figure 3.13). In this dialog box the command buttons each have programme scripts connected to them to enable them to call another dialog box. These pieces of code are the same as the ones used in the menu to call the same dialog box.

The next items in the drop-down list are 'Add Spatial Data' with code *CNCINFAdSpatDat*, 'Edit Spatial Data' with code *CNCINFEditData* and 'Select Data from Database' with code *CNCINFdlgInfDatSel* connected to it. The code calls the dialog box related to it. These dialogs will now be discussed separately.

The 'Add Spatial Data' dialog (refer to figure 3.16) has *CNCINFCallAddsplData* script connected to the 'on open' event. This code sends an SQL statement to the

database to find the features that are not geo-referenced and then puts the values in the combo box provided. The flag button has programme code that changes the cursor, records the x-y co-ordinates, where the user clicks on the View and adds them to the textboxes provided, to enable the user to make manual changes if necessary (Campbell 1992). When the OK button is pressed, an SQL statement is created and sent through ODBC to update the database. When the dialog is closed the *CNCINFStartdlgStart.onClose* script is executed and again opens the Infrastructure tool.

The *'Editing Tool'* dialog (refer to Figure 3.17) draws up a list of themes to edit, when the user clicks the *'GetThemes'* button. The script (*CNCINFStartEditing*) that is connected to it searches the themes in the active view; when it finds the themes with the tag *'ODBC Theme'* it adds them to the list in the combo box provided. This tag was added programmatically for this purpose, when the themes were added from the database to the view. The twelve editing tools on the dialog were copied from ArcView system scripts. By pressing the *'Save/Stop edit'* button the *CNCINFSaveEdits* script executes and the data are sent, with SQL statements, to the database to update the changes. When this dialog is closed it runs the *CNCINFSaveEdits* script and opens the *'Infrastructure Tools'* dialog box. When the *'Infrastructure Data Select'* dialog (refer to Figure 3.14) is opened, the *CNCINFdlgInfraDatOpen* script gets the information from the database it needs to populate the combo boxes. The *'ALL'* keyword enables the user to include all the data from that field. This normally takes a while and for that reason progress is calculated and shown in the status bar. When the *'OK'* button is pressed the *CNCINFOK* script is executed, this error checks the data selected by the user before passing it on to the *CNCINFComboSqlWriter* script. This script takes the user selections and builds SQL statements. The query is then split up into Point, Line and Polygon features and individually sent to the *CNCINFSqlConnection* script to make the ODBC connection in order to select the data from the database. The script checks if there are data available in the query made, provides a user input box to change the name of the themes being added and checks the geometry. It then splits up to do all the different processes for the different geometries. To execute points, it converts the x-y co-ordinates from the database into shape file format and then adds the points to the View. For line data it loops through all the records and constructs each line individually with its own ID in shape format. For polygons the first x-y co-ordinate of

each polygon is stored for re-use as the last record of that polygon, so that the polygons will be closed and not have gaps. After the line and polygon shapes have been created, they are joined to their attribute data. This takes place in the *CNCINFJoinFtabVtab* script.

To filter the selection of data made in the '*Infrastructure Data Select*' dialog, the '*Properties*' dialog is provided (refer to Figure 3.15). This dialog is opened by the diamond command button next to the type combo box on the '*Infrastructure Data Select*' dialog (see Figure 3.14). The reason for this is that this dialog can only be used if a Feature type has been selected. The script *CNCINFOpendlgProperties*, therefore, checks what feature type has been selected and gives an error message if no type was selected. When the dialog is opened the *CNCINFdlgPropertiesOpen* script runs and constructs a title from the type of Infrastructure. It then builds a SQL statement to get the property values for the specific type, hides the combo boxes not being used and calls the *CNCINFPopulateCombos* script to populate the combos being used. After the user has made a selection of data the 'OK' button works in the same way as the one in the '*Infrastructure Data Select*' dialog. It selects data from the database and adds it to the View, also using the property values from the '*Properties*' dialog (Zhang, Dossey Weissmann & Hudson 1994).

The *CNCINFDisolveThemePoint* and *CNCINFDisolveThemeLine* scripts were developed to import shape files into the database system. They are not connected to the interface, because this is not a frequent operation and the fields in the shapefiles can differ. It is therefore not possible to standardise these scripts. The scripts work by taking the *Name* and *Description* fields and using them as unique values. The shapefile is then dissolved to get point values to send over to the spatial handling part (refer to sub-section 3.2.1) of the database in a manner that ensures the different spatial types function properly afterwards.

The final part of the ArcView programming entails creating an Extension that can be added and removed, as required. The scripts involved in this process are the following: *CNCINFExtensionUpload*, *CNCINFCanUpload*, *CNCINFInstallScript*, *CNCINFSplashStart*, *CNCINFLoad*, *CNCINFUninstallScript* and *InfraExtensionWriter*. *InfraExtensionWriter* needs to run in order to write the extension. The other scripts mentioned are called from within this one.

3.3.1.2 Microsoft Access Programming

The programming language used in Access is Visual Basic for Applications (VBA). All the developments discussed in the following section have therefore been programmed in this language (Barker 1996).

The code structure in VBA is different from Avenue in that modules with smaller subsets of code within them are created. These modules are then connected to the forms created in VBA. The advantage of this is that a more modular structure is created. At the onset the Access part of the application provides a start form with three command buttons (refer to Figure 3.4). The code allocated to each button calls up the appropriate part of the application. The code for each module has been attached as presented in Appendix 2 (Deitel, Deitel & Nieto 1999).

i) Infrastructure Tool

This discussion relates to the *Infrastructure Tool* form, with the *Features tab* selected (refer to Figure 3.5). The module connected to this form is the *Form_frmMain*. In VBA each form must have a dataset allocated to it; in this case it is the table '*tblFeatures*' (refer to Figure 3.1 for table). Notice that the fields in the table all match the input boxes on the form, with exception of the IDs, which are hidden from view. For each button a piece of code is attached to do exactly what the button requires.

The following actions have been taken programmatically to make the form user-friendly:

- The property combo boxes not in use are hidden from view;
- The property values in the combo boxes are selected to include only the values allowed; and
- The data record destinations for the spatial form are changed automatically, so that spatial data are sent to the correct tables. For type Point the table stays '*tblPointsData*', but for Lines and Polygons the table is changed to queries *qryLineSpatialData* and *qryPolySpatialData* respectively (the queries used to get these data sources can be viewed in Appendix 3).

The next tab on the '*Infrastructure Tool*' form, '*Inspection*' is in fact another form (refer to Figure 3.6), placed here to make the application more user-friendly and compact. The module linked to it is '*Form_frmInspection*'. The dataset associated

with it is *tblInspections* (refer to Figure 3.1). The functionality of this form is fairly straightforward: input controls are connected to the fields of the database to store the values entered. The database provides the default values, like the date and time. The command buttons have programme code connected to them to call the appropriate form and reports.

The following tab embeds another form '*frmPointsData*' (refer to Figure 3.8) with module '*Form_frmPointsData*'. This module contains only code to manage the spatial data types: Point, Line and Polygon (Denny 1998). The script adds the correct ID for each type in the correct places and checks if the values already exist (refer to sub-section 3.2.1 for clarity on where IDs go).

The '*New Feature Form*' form (refer to Figure 3.9) has extensive error checking built into it. The command buttons to the right of the combo boxes have code connected to them that check if a value was selected in the combo box to its left. If it does not get a data selection, a warning message is created: "There must be a value in the textbox on the left. Use the 'NEW PROPERTY' button to add a Property; cannot edit the value of nothing". Checking is done to see if a value is selected in the spatial type combo box and an error message is created if no value was selected: "A value in Spatial Type is required to add a record!" The data source table for this form is '*tblType*'. The combo boxes are all populated with the same list of all the data available on property values.

The next form is the input form for the values of the properties (refer to Figure 3.10) '*New Feature Property Form*'. This form has another smaller form imbedded, '*Value list sub-form*', which is linked to enable it to change according to the changes made in the main form. This is done by linking a dependent field in the sub-form to a master field in the main form. The sub-form record source is determined by a query: "SELECT DISTINCTROW *tblValueList.PropertyNameID*, *tblValueList.Value* FROM *tblValueList*".

3.3.2 Integration

ODBC was used to integrate the two systems. Windows has this system of referring to data sources as a built-in function. A user can call the connection preset in ODBC to make a connection to pass data to and from the database.

The following diagrams show how the ODBC connection is set up: the '*ODBC Data Source Administrator*' can be reached from window's control bar control panel by double clicking on ODBC Data Source. In the form shown in Figure 3.21, a user can click the '*Add*' command button to add a new System Data Source.

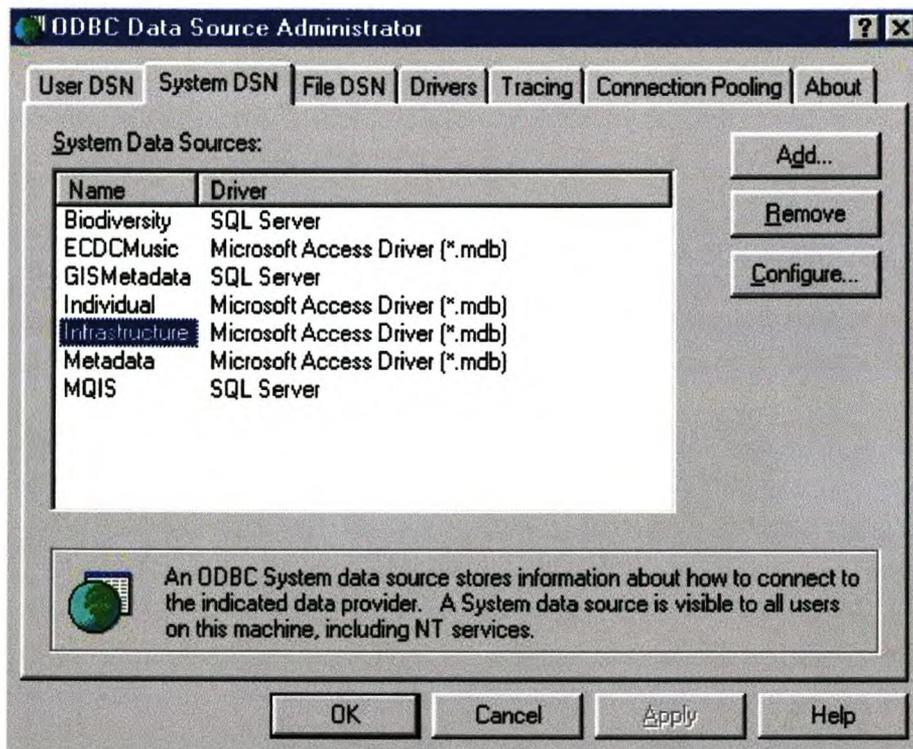


Figure 3.21 Screenshot of ODBC Data Source Administrator

The following list (see Figure 3.22) then appears for the user to choose the driver required to connect to the type of database in use. In this case it is a Microsoft Access database selected in blue in the diagram.



Figure 3.22 Screenshot of form *Create New Data Source*

The setup to identify the location of a data source can be done in the following two message boxes. The name the user wants to use to identify the data source must be typed into the text box with 'Infrastructure' typed within it in Figure 3.23. The 'Select' command button must then be clicked to open the *Database Select Form* (see Figure 3.23).

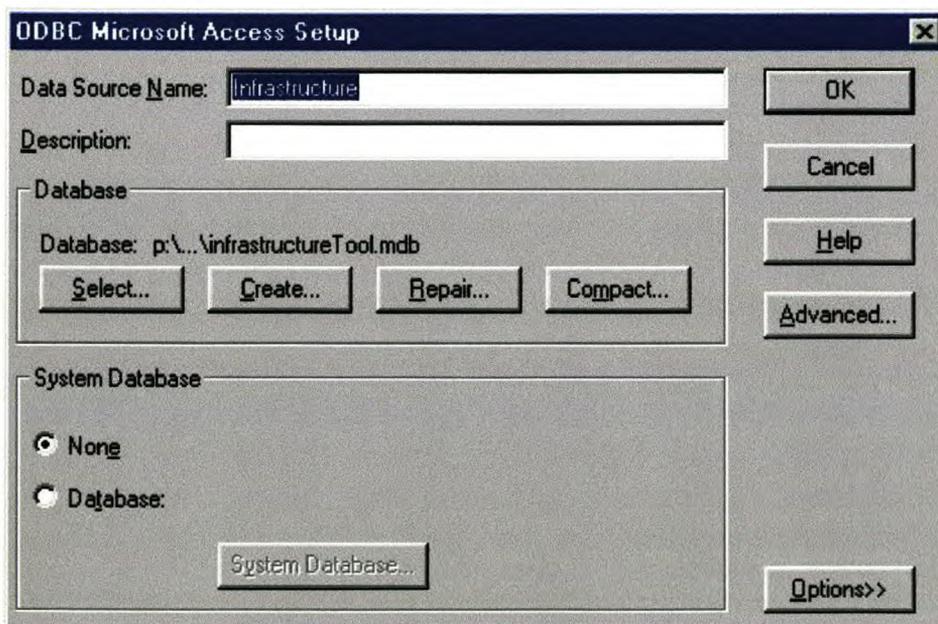


Figure 3.23 Screenshot of the 'ODBC Microsoft Access set up' form

In this form the database that relates to the type of source and the name the user decided on has been selected.

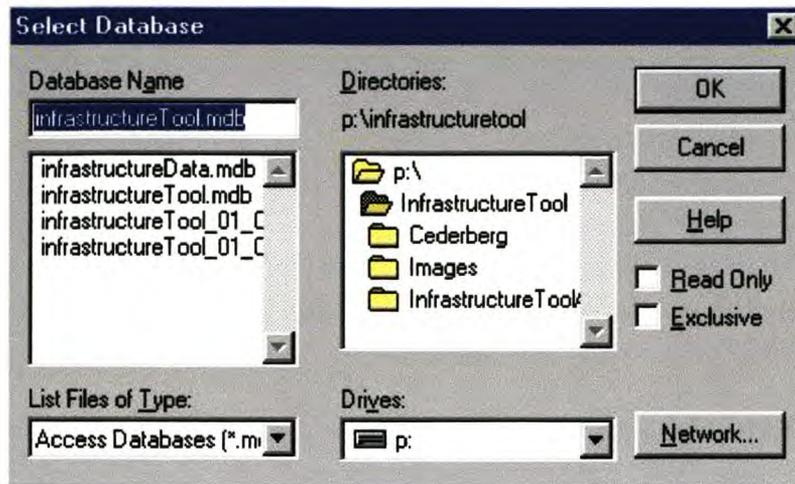


Figure 3.24 Screenshot of the 'Select database' form

When this is done, the user can use the name 'Infrastructure' to refer to the database when making queries. This is then done from ArcView to Query the Access database.

3.3.3 Testing and Debugging Corrections

The testing of a system is very important to make sure that all obvious program errors are removed and that the system functions with real-life data. Before executing the real-life data test, each script was first tested individually until the programmers was satisfied that it functioned correctly and effectively. This testing phase lasted several months.

For the real-life data test, data from Cederberg Nature Reserve, consisting of pointdata contained in *cedb_infrastructure.shp* and lines in *cedb_trails.shp*, were used. Samples of the data are shown in Figures 3.25 and 3.26. Dams were added to test the system's ability to handle polygons.

The testing of the system consisted of two phases. During the first phase the system was tested thoroughly by checking each form and dialog to see if it returns any errors by checking the results manually. The installation and uninstallation of the extension was tested and to get it working without a hitch, the scripts were rebuilt 18 times. The flow of data and the ODBC connection was checked by sending data from ArcView to the Access database and retrieving the data into ArcView. Thereafter the following

corrections were made to solve the problems that were discovered:

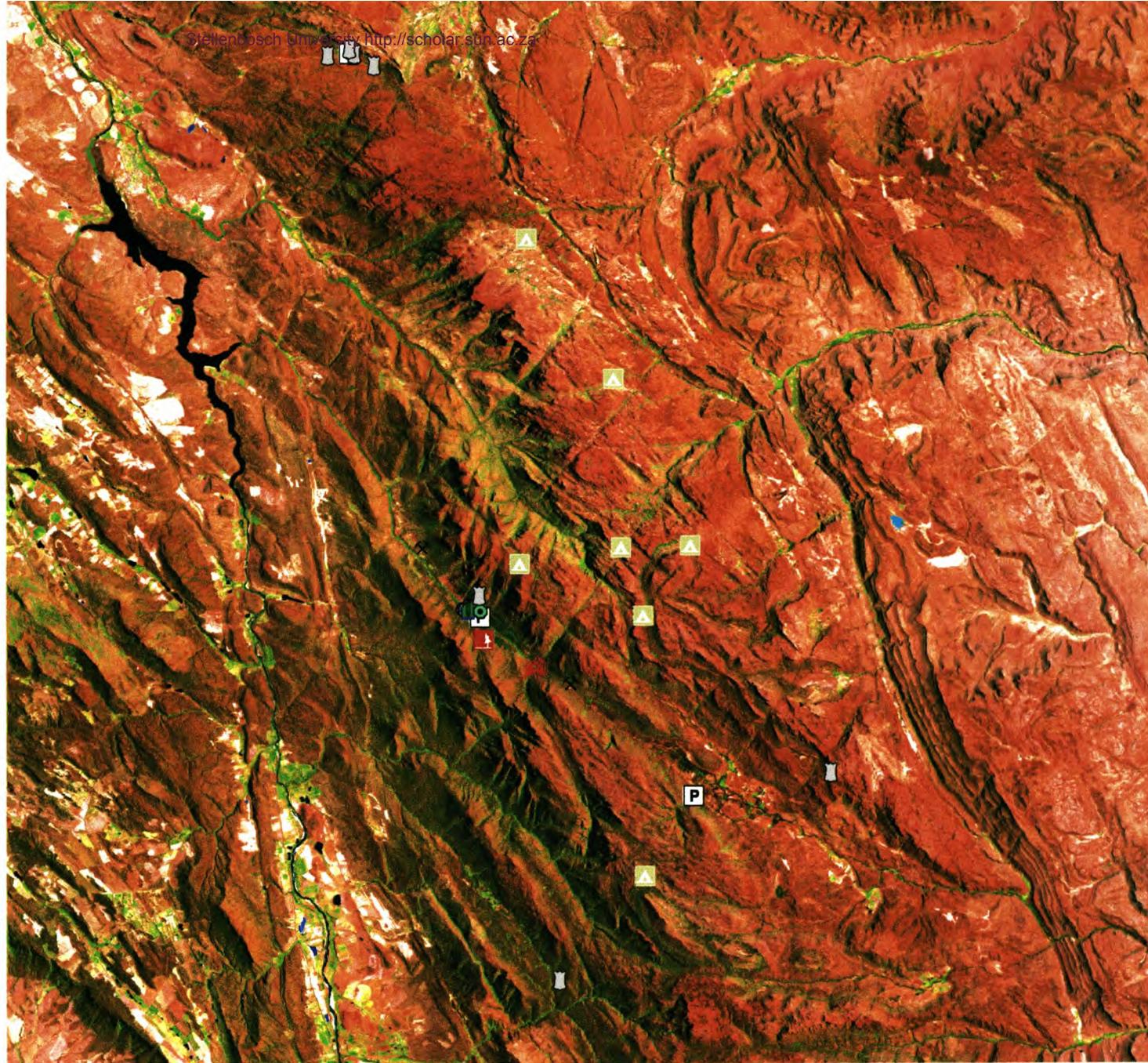
- The IDs that identify the x-y co-ordinates for each feature were not always placed in the correct location, because there was a problem with an update query when line and polygon data were sent. The tables were not correctly joined and resulted in IDs going to incorrect locations.
- When the Line and Polygon Shape files were imported into the database and the data were displayed in ArcView, vertices had spikes, meaning one point was totally out of position. Eventually it was discovered that the update query used for importing data had to be sorted according to auto number fields because the x-y co-ordinates did not remain in sequence.
- Some of the dialogs did not refresh the related values immediately after a new value was added. Simply adding a Refresh command to the programme solved this problem.
- When a query was made in ArcView, all the data related to it were not displayed. This ended up being another query building error.
- A great number of temporary tables were created in ArcView and these had to be deleted automatically after use.
- The editing dialog did not save changes when exiting the dialog. To solve this, programming code was added to save and upload the data to the database.
- Initially, more than one theme could be edited at the same time, but an error occurred when trying to upload the data. This was corrected by limiting the editing to one theme at a time.
- The splash screen did not show when the extension actually loaded. To correct this the file path had to be changed.

When the initial testing was completed, the second phase was to find users, explain the system to them and get feedback from them regarding their assessment of the system. The feedback was discussed at a meeting with the users, where it was decided to stop expanding the functionality and make the following changes to the system:

- The users thought some of the commands took too long to execute and wondered if the system was unstable. To resolve this problem, a progress bar was added to the bottom whenever a process takes time.
- Some dialogs had retained old data when the user restarted them and, although this did not affect the operation, this confused users. The dialogs are therefore refreshed before they are opened.

KEY

-  Ablution block
-  Block House
-  Borrow Pit
-  Campsite
-  Cottage
-  Dorsvloer
-  Fire Look out
-  Helipad
-  Hut Hiking
-  Info Centre
-  Inspection Quarters
-  Leopard cage
-  Office
-  Parking Area
-  Reservoir
-  Sewage Treatment Plant
-  Stable
-  Staff house
-  Store Flammable
-  Store General
-  Swemgat
-  Unmarked Graves
-  Weather Station



7 0 7 14 Kilometers

Figure 3.25 Cederberg
Pointdata sample

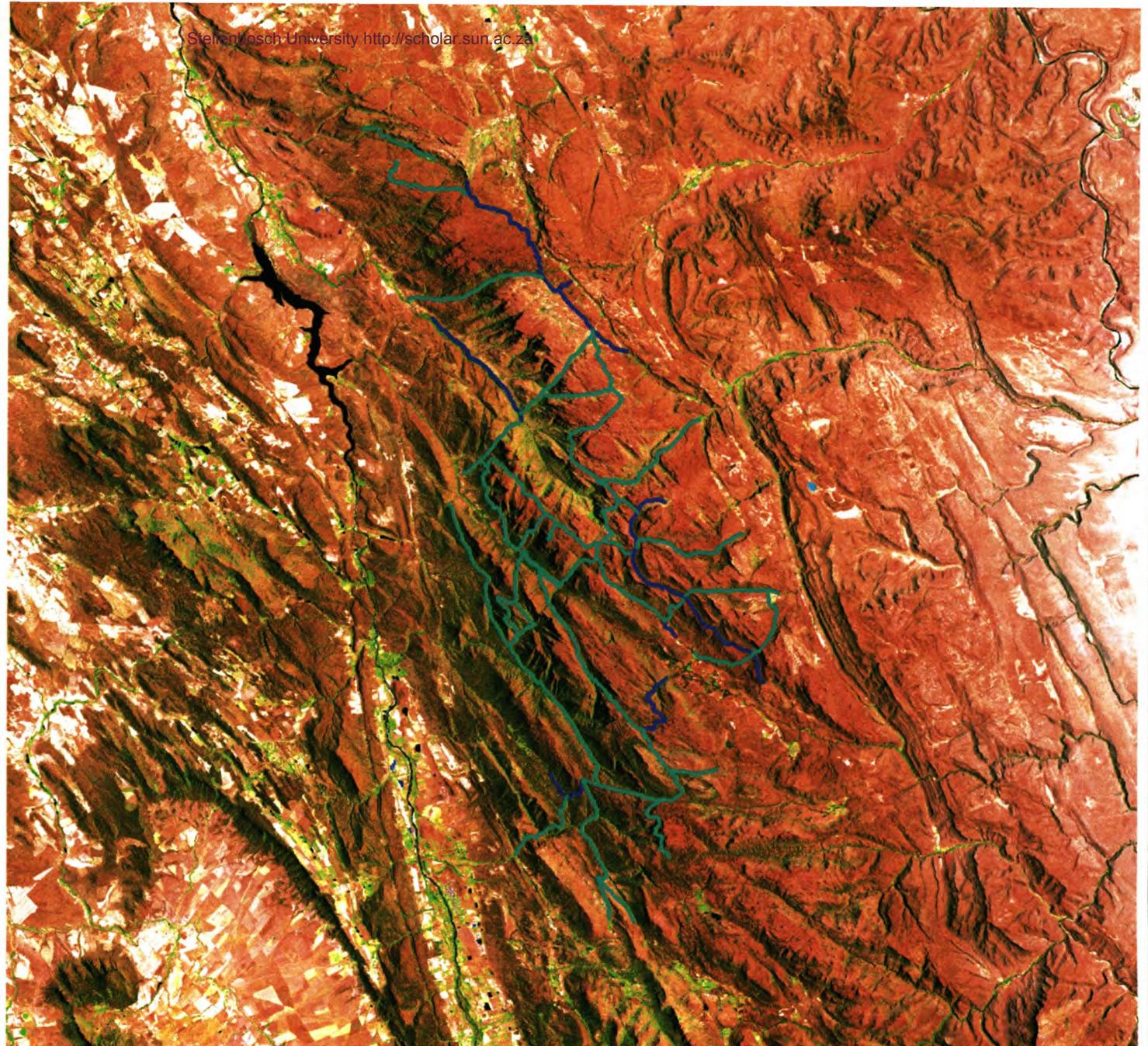
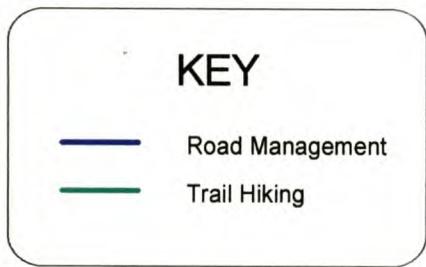


Figure 3.26
Sample of Cederberg
Line test data

Now that the second phase of testing has been completed, the IMSS is ready to be distributed to the users. It is inevitable that more problems will arise from the day-to-day use of the IMSS. Fixing the problem in a local copy at the Scientific Services Division and then redistributing the application will address these issues.

The next chapter of this report provides assistance to the user in applying the system. Appendix 4 gives a description of the installation of the system on a Personal Computer.

CHAPTER 4 APPLICATION OF AN INFRASTRUCTURE MANAGEMENT SUPPORT SYSTEM

In this section the operation of the system will be discussed from a user's perspective. The focus is on the GUI of the application and describes its functionality in detail.

4.1 OPERATION OF THE IMSS

The following section consists of a description of the Access Database that is mostly used for data manipulation.

4.1.1 Access Database

This part of the system has two main functions. The first is data entry and editing, while the second is creating reports to assist the user with decisions. The adding and editing function can be further sub-divided into two parts (Lyon 1989). The first part entails the adding of features and their properties, which must be done before the second part, which is entering attribute data. To start this system off, the user can select one of the three buttons displayed when the 'InfrastructureTool' file is opened (see an example of this form in Figure 3.4).

The following sub-section of the report provides a description of how new features and properties can be added to the available list.

4.1.1.1 Adding features

With reference to Figure 3.9 follow the numbering. By clicking on the button at *1* a new feature type can be added, all input boxes are cleared and a new record is created. Therefore, a new feature name can be added at *2*. This must be a unique name that does not appear in the drop-down list. At *3* the user must choose from the list what spatial type the new feature should be. At *4* the user can choose a property from the list for all five of the properties if required. The properties chosen must relate to the feature added in *1*. If the property the user needs does not appear in the list, the button at *6* can be clicked to go to the 'New Property' form (see Figure 3.10) in order to add a new feature. If the user wants to change the values allowed for that property, the button at *7* (adjacent to the property to be edited) can be clicked. By clicking on *5* the user can do a text search in any field on the form to speed up the search for a specific record. The cursor must, however, be in the field the user wants to search before the

button is pressed. After the new feature data has been added successfully, the form can be exited by pressing the exit button or the window close button (Horton 1991).

4.1.1.2 Adding Properties

The New Features Properties Form (Figure 3.10) is used to add new values allowed for each property. This form helps to maintain data integrity. When the button at **1** is clicked, all the input spaces are cleared and a new record is added for a new property. At **3** the new property name must be added. When the user wants to change the values allowed in an existing property, the list can either scroll down and new ones added or a specific one edited by clicking on it, for example, by clicking at **6** and changing the value. By clicking the button at number **2** the cursor jumps to the end of the list, ready to add a new record. When all changes have been made, the OK button at **4** can be clicked. The list entered in the frame labelled 'Value' will be displayed in the property values list box on the *Infrastructure tool* form (refer to Figure 3.5 Nos **5** and **6**), so that the user can select an item from the list (Horton 1994). Once all the required features and their properties have been added, the data-entering process can proceed.

4.1.1.3 Data entering

The Data Entry form (Figure 3.5) can be used to add, edit or delete features. When the user needs to edit an entity, he or she can navigate to that feature with the navigation bar at **14**. To delete an entire record, click on the bar at **15** to select the whole entry and then use the delete key on the keyboard to delete it. To add a new record, click on the button at **2**; items **3** to **8** will then be cleared and the user can start entering the data from the top. When a type is selected from the list in **3**, the properties associated with it will be displayed just below the 'Names' text box, numbered as **5** and **6** in this example. A name that distinguishes the record from others of the same type must be entered at **4**. The description at **7** will help the user to identify the item. Number **8** provides a list of all the Nature Reserves in the Western Cape and the reserve to which the data belong must be selected (Lynch 1994). The button indicated as **1** gives a description of the application. The form is closed by clicking on Exit at **11**, whereas Quit at **12** closes the application. A report of all the data in the database can be obtained by clicking on button **9** so that the reserve manager will be able to obtain an inventory of the infrastructure in the nature reserve under his/her authority (refer to Figure 4.3 for a sample of the report). The form is divided into three tabs. To get from the one tab to the next, the user can click on the tab at **13**.

Figure 3.5 shows an example of the *Infrastructure Tool* with its *Features tab* selected and Figure 3.6 shows the same tool with its *Inspection tab* selected. On this tab the user should key in the inspection data. The *inspection data* are related to all the data in the other tabs. The date and time boxes relate to the date and time the inspection was performed. A photo path is available if the user has photographs of the features. The user has to copy the photograph to the exact location the path indicates in order for the application to display the photo. The button at *1* can be pressed to display the photo (see Figure 3.7 for an example of the form).

If a second inspection of the same feature was performed at a different time, the user can use the navigation bar at *2* (refer to Figure 3.6) to add a new inspection. With these data the user can track changes to the infrastructure over time. By clicking the button at *3* the user can activate the report in Figure 4.1 that divides the infrastructure into different condition categories to monitor the condition of the infrastructure and with the button at *4* it is possible to monitor the inspections made to see how much work was done by whom (refer to Figure 4.2).

To move to the next tab for spatial data entry, the user must click the tab at *5*. The *Pointdata* tab is the last tab in the *Infrastructure Tool* and is used to enter the spatial data related to the system. It can be used to edit, add and delete data. It is not recommended that the bulk of the spatial data be entered from this tab. Spatial data should rather be entered from the Cybertracker GPS system of the WCNCB, or from shapefiles and imported into the system (scripts were developed to enter line and point data). If the user wants to enter data using this tool, the x and the y co-ordinates should be keyed into the Longitude and the Latitude text boxes respectively. The best way to enter line data would be to add two dummy points with this tool, go to ArcView, edit the line and send it back to the database. The user would give the line a name and use this name to find the line in ArcView. The same procedure can be used for a polygon, but three dummy points must be entered before it is accessed in ArcView (Le Clercq & Tomson 1992).

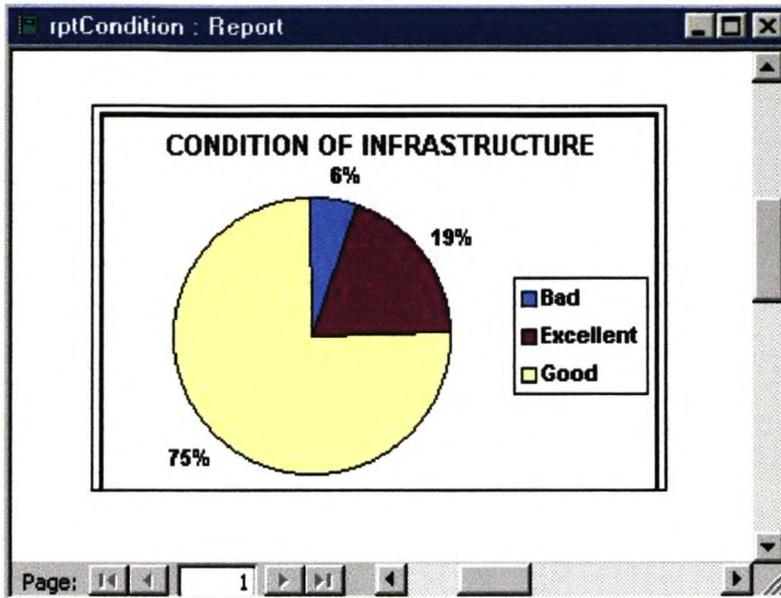


Figure 4.1 Example of a report on the condition of the entire infrastructure of a nature reserve.

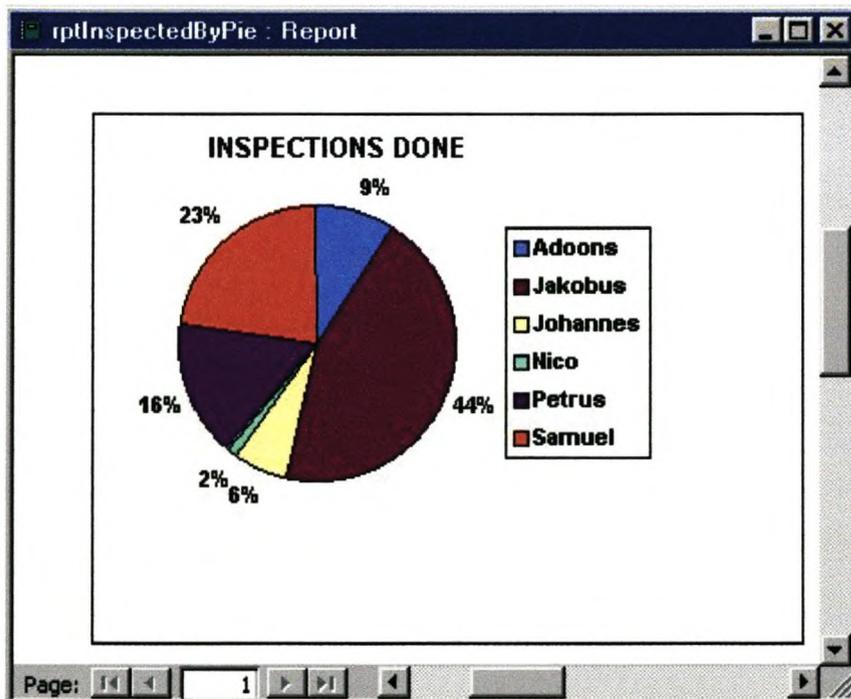


Figure 4.2 Example of an inspection report that shows what percentage of the total inspections were done by specific persons.

INFRASTRUCTURE PROPERTY VALUES

<i>Type</i>		<i>Ablution Block</i>									
<i>Names</i>	<i>Description</i>	<i>Property</i>	<i>Value1</i>	<i>Property</i>	<i>Value2</i>	<i>Property</i>	<i>Value3</i>	<i>Property</i>	<i>Value4</i>	<i>Property</i>	<i>Value5</i>
Algeria	Big abluion block	Construction Material	Stone	Area	4m2						
Algeria	Small abluion block	Construction Material	Stone	Area	4m2						
Kliphuis	At Kliphuis	Construction Material	Stone	Area	4m2						
<i>Type</i>		<i>Block Houses</i>									
<i>Names</i>	<i>Description</i>	<i>Property</i>	<i>Value1</i>	<i>Property</i>	<i>Value2</i>	<i>Property</i>	<i>Value3</i>	<i>Property</i>	<i>Value4</i>	<i>Property</i>	<i>Value5</i>
Boskloof	Build during Anglo Boer W	Construction Material	Stone	Paint	Green	Area	16m2				
Krakadouw	Build during Anglo Boer W	Construction Material	Stone	Paint		Area	9m2				
<i>Type</i>		<i>Borrow Pit</i>									
<i>Names</i>	<i>Description</i>	<i>Property</i>	<i>Value1</i>	<i>Property</i>	<i>Value2</i>	<i>Property</i>	<i>Value3</i>	<i>Property</i>	<i>Value4</i>	<i>Property</i>	<i>Value5</i>
Roolkoppie	Not in use	Area	4m2	Depth	2m						
Uitkyk	Not in use	Area	4m2	Depth	2m						
<i>Type</i>		<i>Camp Site</i>									
<i>Names</i>	<i>Description</i>	<i>Property</i>	<i>Value1</i>	<i>Property</i>	<i>Value2</i>	<i>Property</i>	<i>Value3</i>	<i>Property</i>	<i>Value4</i>	<i>Property</i>	<i>Value5</i>
Algeria	48 sites	Surface Type	Grass								
Kliphuis	10 sites	Surface Type	Grass								

Figure 4.3 Sample of Cederberg reserve inventory list.

4.1.2 ArcView GIS

When the extension *CNC Infrastructure Extension* is loaded, the menu item *Infrastructure* is added with a list of the functionality encompassed within this extension. Selecting the first item in the list will provide the user with a Graphical User Interface (refer to Figure 3.16) to navigate through the program, providing access to the following tools:

4.1.2.1 The *Data Select* tool

The purpose of this tool is to select data from the database for displaying and viewing in a View to inspect the data. To add the data to a particular View, that View must be the selected item before switching to the *Data Select* tool. All the data, except the years, can be selected from a drop-down list. By selecting certain field values, a query will be built (Lynch 1994). Only one or all the fields can be selected at once. To make a query selecting all the data on the database, all the lists can be set at the 'All' option which makes the date range broad enough to incorporate all the data. To construct a date the day is specified in the drop-down list in *1*, the month in *2* and the year typed in the box at *3* in the format: 1998. The date range within which the data were gathered must be selected by using a start date before and an end day after the required date range. The program will automatically select the days between the chosen dates. When all the selections are made, press *OK* at *5* to continue, or *Quit* at *4* to cancel. To select data with a certain property, the button at *6* is clicked to add that functionality (Figure 3.15).

This property function can only be used if a specific type was selected from the *Type* field in Figure 3.14. This form selects the properties of the type selected in Figure 3.14 and displays them with the values allowed for the particular properties in a list. With this form a certain type of feature with specific properties can be selected. The user can, for example, select 'hut' as a type, 'stone' as the construction material and '10m²' as the area. A query that selects all the data with these properties from the database will then be constructed in order for the data to be selected. Once the task has been completed either *Cancel* or *OK* can be selected. *OK* will incorporate all the query selections made in Figure 3.14 and 3.15.

4.1.2.2 Spatial Data Editing Tool

The *Editing tool* (see Figure 3.17) is used to edit themes from the infrastructure database. This tool cannot be used to add a new feature; it can only edit existing points, lines and polygons.

To use this tool, themes from the database must be available in a view, while to add themes the *Infrastructure data Select* tool should be used. More than one window can be open in one project. Consequently the system was designed to address the last selected view. For this reason the user must select the desired *View* and then press button *1* to start using the Editing tool, after which a theme from the database is selected for editing from the drop-down list in *2*. The button at number *14* will be hidden when a point theme is selected, because a point theme has no use for it.

The following is a description of the function of each of the buttons in the Editing Tool:

- 3* Selects, moves and resizes graphics;
- 4* Goes to the full extent of all the themes;
- 5* Zooms to the extent of active themes;
- 6* Zooms to the extent of the selected features;
- 7* Zooms in on the centre of the display;
- 8* Zooms out from the centre of the display;
- 9* Goes back to the previous extent the user was viewing;
- 10* Drags the display in the direction the user moves the cursor;
- 11* Zooms out from a point the user clicks or zooms out to include a rectangle the user drags;
- 12* Zooms in at a point the user clicks on or zooms in on a rectangle the user drags;
- 13* Selects features in the active themes by pointing or dragging;
- 14* Adds, moves and deletes vertices of features; and
- 15* Saves the edits made and updates the database with the new data.

4.1.2.3 Spatial Data Adding Tool

The Spatial Data Adding Tool can be used to add spatial data to point features in the database (refer to Figure 3.16). The tool queries the database to find all the point features without spatial data and then adds these features to the list in the box at *1*. A point feature can be selected from the list to add spatial data to it. By pressing button *2* the cursor is activated for taking measurements. When the cursor is pointed to a view, it (the cursor) will change into a crosshair and a point on the view can then be selected. The co-ordinates of the selected point will appear in the boxes at *3* and *4*. The view in use must be projected, otherwise it will not show any co-ordinates. The co-ordinates can be edited, if required. To send the data to the database the OK button at *5* is clicked (Lupien, Hilbert & Paschak 1992). The button at *6* can be pressed to refresh the list box in *1* if more features in the database were added.

CHAPTER 5 SUMMARY, EVALUATION AND FURTHER DEVELOPMENT

The purpose of this study was to develop an infrastructure management support system for the Western Cape Nature Conservation Board. This section will review what was achieved and evaluate whether the aims set out at the beginning have been met.

The study gave a short background of the key concepts of an Infrastructure Management Support System. Then the mission of the WCNCB and their GIS Department was described. The managers expressed their desire for a system to show the location, type, age and service details of the infrastructure under their jurisdiction. From this needs analysis it was clear that GIS should play an integral role in the development of an Infrastructure Management Support System required by the reserve managers.

In summary, the primary goal was to design a system to help the managers with their task of managing the infrastructure of a reserve. An inventory of the infrastructure needs was established and the managers' tasks and computer skills were evaluated. This was followed by an assessment of the information technology environment at the WCNCB according to hardware, networks, databases and software. On the basis of that survey, it was decided to develop the IMSS for Windows 98 using the Microsoft Access 97 relational database and ArcView 3.1 GIS software. It was further decided that the network would be a LAN with a centralised database. A database and GUI was designed for all the components and the information flow was established. Once this task was completed, the system was implemented and a description given of the system from a user's point of view.

For a system to provide the required support to a manager, it should facilitate entering data on the infrastructure of nature reserves. The IMSS provides this functionality and stores the data in a relational database. In this database set-up the system has the ability to provide each feature with the following attributes:

- Three dimensional location;
- Type with maximum of five properties;
- Description;

- Age;
- Condition (more than one value can be stored connected to a date to track changes);
- Name(s) of the person(s) who carried out inspections;
- Photographs; and
- The name of the reserve that the data are related to.

The system was furthermore designed so that the user can, at any stage, add new features and properties to keep it updated. These data can be queried in ArcView in order to look at any part of the infrastructure data. A map can be displayed of the data with each feature's attribute data connected to it for the manager to use for analysis. The system provides a very user-friendly way of querying the data. Any spatial changes to the infrastructure data can also be done. To help the manager with decisions, three reports may be produced. The first is an inventory of all the features with their properties (refer to Figure 4.3). The second and third reports are pie charts showing the percentage of the inspections done by different people (Figure 4.2) and the condition of the infrastructure (Figure 4.1).

The development of an IMSS is an all-encompassing and complicated task, and many improvements could still be made to the current system. The following improvements and further developments are suggested:

- Presently spatial data can only be added by adding the feature into the database, then the dummy points must be edited in ArcView and thereafter the data are sent back to the database. This is cumbersome and can be improved by changing the system to detect an ID in the database and then adding the data directly from ArcView to the database, without entering the data into Access.
- Building more reports for the user to choose from will be useful and will further expand the functionality.
- The current system has no financial function to assist management with budgeting and financial planning. This would be a difficult function to incorporate, but it would be very useful to assist in making financial decisions relating to the infrastructure.
- To make the system more user-friendly, the labels for each feature can be standardised and added automatically when the data are displayed in ArcView.

The researcher is convinced that GIS is without any doubt the best software foundation for an Infrastructure Management Support System. It has great potential for incorporating spatial analytical and modelling capabilities that could be of incalculable value for improving nature reserve management.

CHAPTER 6 REFERENCES

- Amir, H & Warwick, V 1997. ArcView GIS/Avenue Programmer's Reference Second Edition. USA: Onword Press.
- Angehrn, AA, & Lüthi, HJ 1990. Intelligent Decision Support Systems: A Visual Interactive Approach. *Interfaces*, 20, 6, 17-28.
- Apers, PMG, Blanken, HM & Houtsma, MAW 1997. *Multimedia databases in perspective*. Great Britain: Springer.
- Armstrong, AP, & Densham, PJ 1990. Database organization strategies for spatial decision support systems. *International Journal of Geographical Information Systems* 4: 1, 3-20.
- Aronoff, S 1989. *Geographical information systems: a management perspective*. Ottawa: WDL Publications.
- Baldwin, R F 1984. *Operations Management in the forest Products Industry*. San Francisco: Miller Freeman Publications.
- Barker, FS 1996. *Access 95 Power Programming*. USA: Que ® Corporation,
- Batty, PM 1992. GIS data bases: a distributed future. *Mapping Awareness* 6,5: 34-37.
- Berry, J K 1993. *Beyond mapping: concepts, algorithms & uses in GIS*. Colorado: Fort Collins.
- Berry, J K & Ripple, WJ 1994. *The GIS Applications Book: Examples In Natural Resources A Compendium*. Bethesda MA: American Society for Photogrammetry and Remote Sensing.
- Blau, D 1993. GIS: The Future of Resource Management and Land Planning. *Electric Perspectives*, July/August: 554-55.

- Bromley, P 1990. *Countryside Management*. London: Chapman and Hall.
- Bromley, R & Coulson, M 1989. The value of corporate GIS to local authorities. *Mapping Awareness* 2, 5: 32-35.
- Burrough, PA 1988. *Principles of Geographic Information Systems for Land Resources Assessment*. Oxford: Clarendon Press.
- Buxton, R 1991. Geographic information in local government – steering in the right direction. *Mapping Awareness* 5,2: 39-41.
- Campbell, H 1992. Organisational issues and the implementation of GIS in Massachusetts and Vermont: some lessons for the United Kingdom. *Environment and Planning B: Planning and Design* 19,1: 85-95.
- Clarke, KC 1997. *Getting started with Geographical Information Systems*. California: Prentice-Hall.
- Crossland, MD, Wynne, BE & Perkins, WC 1995. Spatial Decision Support Systems: An overview of technology and a test of efficacy. *Decision Support Systems*, 14, 219-235.
- Dale, PF & Mclaughlin, JD 1989. *Land information management*. Oxford: Clarendon Press.
- Dangermond, J 1983. A classification of software components commonly used in geographic information systems. In Peuquet, DJ & Marble, DF (eds.) *Introductory readings in Geographic Information Systems*. Pp. 30-51. London: Taylor and Francis. (re-printed from Design and implementation of computer based Geographic Information Systems).
- Dangermond, J 1989. *Feature-oriented approach versus geo-relational approach*. Redlands, USA: Environmental Systems Research Institute.

- Dangermond, J 1991. Where Is Technology Leading Us? In: GIS Applications In Natural Resources. Colorado: GIS World Books.
- Date, CJ 1990. *An introduction to database systems*. Vol. 1. Massachusetts: Addison-Wesley.
- De Klerk, HM & Sutton, T 2000. *A strategy for the implementation of a Geographical Information System for Western Cape Nature Conservation*. Jonkershoek: Scientific Services Division.
- De Man, WHE 1990. Establishing a geographic information system in relation to its use: a process of strategic choices. In Peuquet, DJ & Marble, DF (eds.) *Introductory readings in Geographic Information Systems*. pp. 324-340 London: Taylor & Francis.
- Densham, PJ 1991. Spatial Decision Support Systems. *Geographical Information Systems, Volume 1: Principles*, edited by Maguire, DJ, Goodchild, MF and Rhind, DW, Longman, 403-412.
- Deitel, HM, Deitel, PJ & Nieto, TR 1999. *Visual Basic 6 How to Program*. New Jersey: Prentice-Hall.
- Demers, MN 1997. *Fundamentals of Geographical Information Systems*. New Mexico: John Wiley & Son, Inc.
- Denny, C 1998. *Sams Teach yourself Visual Basic 6 in 21 days*. Indiana: SAMS.
- Engelken, LJ 1993. Simple user interfaces protect database integrity. *GIS World* 6,2: 68.
- Environmental System Research Institute (ESRI) 1996. *Avenue Customization and application development for ArcView GIS*. USA: ESRI.
- Eom, S, Lee, S, & Kim, J 1993. The intellectual structure of Decision Support Systems (1971-1989). *Decision Support Systems* 10, 19-35.

- ESRI 1995. *Understanding GIS: The ARC/INFO method*. Cambridge: GeoInformation International.
- ESRI 1996. *Managing a GIS*. USA: Environmental Systems Research Institute.
- ESRI 1997. *Arcview Dialog Designer*. USA: ESRI.
- FAO conservation guide 1996. *Computer-assisted watershed planning and management*. USA: FAO.
- Freeman, C 1997 *Step by Step Microsoft Access 97*. USA: Microsoft Press.
- Getz, K, Litwin, P & Gilbert, M 1999. *Access 2000 Developer's Handbook* USA: SYBEX
- Grigg, NS 1988. *Infrastructure Engineering and Management*. New York: John Wiley and Sons.
- Guptill, SC 1996. The risks and rewards of GIS technology. *South African Journal of Geo-Information*, 17, (1): 3-9.
- Healey, RG 1991. Database Management Systems. In: Maguire, David *et al.* (Ed.) *Geographic Information Systems. A Longman Scientific and Technical Publication*. I: 251 - 267.
- Heit, M & Shortreid, A. (eds.) 1991. *GIS Applications in Natural Resources*. Colorado: GIS World Books.
- Heit, M, Dennison-Parker, H & Shortreid, A (eds.) 1996. *GIS Applications in Natural Resources 2*. Colorado: GIS World Books.
- Hentzen, W 1999. *Access 2000 programming from the ground up*. New York: Nordan BA.

- Hitt, W D 1985. *Management in Action Guidelines for Managers*. Columbus, Ohio: Battelle Press.
- Horton, WK 1991. *Illustrating computer documentation*. New York: Wiley.
- Horton, WK 1994. *Designing and writing online documentation, 2nd Edition*. New York: Wiley.
- Hudson, WR, Haas, R & Uddin, W 1997. *Infrastructure Management*. New York: McGraw-Hill.
- Jankowski, P 1995. Integrating geographical information systems and multiple criteria decision-making methods. *International Journal of Geographical Information Systems*: May-June, 1995: 9, 3, 251-73.
- Jones, J & Monk, S 1997. *Databases in Theory and Practice*. South Africa: UK International Thomson publishing.
- Lansdale, M W & Ormerod, T C 1995. *Understanding Interfaces a handbook of human – computer dialogue*. UK: Academic Press.
- Laurel, B 1990. *The art of human-computer interface design*. Reading, Mass: Addison-Wesley.
- Le Clercq, R & Thomson, D 1992. *Fundamentals of spatial information systems*. London: Academic Press.
- Lupien, AE, Hilbert, JA & Paschak, SH 1992. User interface lifecycle of industrial GIS applications. *Proceedings of the Fifth International Symposium on Spatial Data Handling*, Charleston, Vol. I: 30-39.
- Lynch, PJ 1994. Visual design for the user interface, Part 1: Design fundamentals. *Journal of Biocommunication* 21, 1: 22-30.

- Lynch, PJ 1994. Visual design for the user interface, Part 2: Design fundamentals. *Journal of Biocommunication* 21, 2: 6-15.
- Lyon, F 1989. Implementing a GIS, the Hampshire experience. *Mapping Awareness* 3,6: 19-21.
- Macleane, A (ed.) 1994. *Remote Sensing and Geographical Information Systems: an integration of technologies for resource management*. Bethesda, MA: American Society for Photogrammetry and Remote Sensing.
- Magellan System Corporation 1997. *User guide for the Magellan GPS ProMARK X and the Magellan GPS ProMARK X-CM*. California: Magellan System Corporation.
- Maguire, DJ 1991. An overview and definition of GIS. In Maguire, DJ, Goodchild, MF & Rhind, DW (eds) *Geographical Information Systems: Principles and Applications*. Vol. 1. : Longman Scientific and Technical.
- Mallach, EG 1994. *Understanding Decision Support Systems and Expert Systems*. USA: Irwin.
- Mallawaarachchi, T, Walker, PA, Young, MD, Smyth, RE, Lynch, HS & Dudgeon, G 1996. GIS-based Integrated Modelling Systems for Natural Resource Management. *Agricultural Systems*, 2: 169-189.
- Marble, DF 1990. Geographic Information Systems: an overview. In Peuquet, D.J. & Marble, D.F. (eds) *Introductory readings in Geographic Information Systems*. London: Taylor & Francis. pp. 8-17.
- Marcus, A 1992. *Geographic design for electronic documents and user interfaces*. ACM: Addison-Wesley Press.
- Masser, I 1992 Computers in Local governments: Malaysia. *Habitat International* 15, 4: 51-63.

- Microsoft Corporation 1992. *The windows interface: An Application design guide*. Redmond, WA: Microsoft.
- Miller, GT 1996. *Living in the environment*. Ninth edition. .USA: Wadsworth Publishing Company.
- National Science Foundation 1994. *Civil Infrastructure System Research*. Washington D.C.: National Science Foundation.
- National Science Foundation 1995. *Civil Infrastructure System - An Integrated Research Program*. Washington D.C.: National Science Foundation.
- Oxford Concise Dictionary 1999. Oxford, UK: Data Ltd.
- Paterson, WDO & Scullion, T 1990. *Information Systems for Road Management: Draft guidelines on System Design and Data Issues*. Technical Paper INU77, Washington DC: The World Bank, Infrastructure and Urban Development Department.
- Peeters, S & Jarosz, A 1996. The use of geographical Information Systems in the Utilization and Visualisation of Mining data. *Journal of the South African Institute of Mining and Metallurgy*. December 1996: 303-309.
- Poolman, J 1992. *GIS an IT Perspective*. Pretoria: Computer Foundation.
- Razavi, AH 1995. *Arcview Developer's Guide*. VSA: On Word.
- Riley, NW 1991. GIS and the management of change in Southern Africa. *Suid-Afrikaanse Tydskrif vir Geo-inligting* 16, 2: 40-44.
- Smith, GJ 1995. *Research Guidelines for Planning and Documentation*. Pretoria: Southern Book Publishers.
- Somers, R 1991 GIS in local government. *Cities* 8,1: 25-32.

Spooner, R 1992. GIS for land use management in local government: a vendor's perspective. *S9 update* 7: 22-29.

Uddin, W 1995. *Pavement Material Property Database for Pavement Management Applications*. Vol. 4 Philadelphia: American Society for Testing and Materials Databases.

Urenda, C 1992. PC-based GIS manages municipal water system. *GIS world*: 43-47.

Valenzuela, CR 1991. *Spatial databases. In Remote sensing and geographical information systems for resource management in developing countries*. Brussel: Kluwer.

Verhoef, E 1999. *Infrastructure definition. Design and Management of Infrastructure*, Amsterdam: Tudelft [Online].

Available :<http://www.infrastructures.tudelft.nl/infradef.html> [2001/06/22].

Webster's Revised Unabridged Dictionary 1998. Chicago: MICRA Inc.

Webster's Revised Unabridged Dictionary 2001. Chicago: Random House Inc.

Yeates, D, Shields, M & Helmy, D 1994. *Systems Analysis and design*. UK: Clays Ltd.

Zhang, Z, Dossey, T, Weissmann, J & Hudson, WR 1994. *GIS integrated Pavement and Infrastructure Management in Urban Areas. Transportation Research Record 1429*, Washington DC: National Research Council.

Personal communication

Alphabetise list

De Klerk, HM 2001. Information given to NM van Zyl during personal communications in connection with development of IMSS in 2001.

Laurens, L 2000. Information given to NM van Zyl on 10 December 2000 during a personal communication in connection with Infrastructure inventory list, Hottentots-Holland Nature Reserve.

Sutton, T 2000. Information given to NM van Zyl during personal communications in connection with development of IMSS in 2000.

Sutton, T 2001. Information given to NM van Zyl during personal communications in connection with development of IMSS in 2001.

Shaw, K 2001. Information given to NM van Zyl during a personal communication in connection with management in WCNCB.

Turner, AA 2001. Information given to NM van Zyl during personal communications in connection with development of IMSS in 2001.

CHAPTER 7 APPENDICES

APPENDICES	Page
APPENDIX 1 Avenue code used in the development of the IMSS	1
APPENDIX 2 Visual Basic for Application code used in the development of the IMSS	68
APPENDIX 3 Two SQL queries used in the development	90
APPENDIX 4 Installation procedure of IMSS on Personal Computer	91
APPENDIX 5 ArcView training for WCNCB staff	93

APPENDIX 1

```
'Name: CNCINF.ExtensionUnload
'date: 2001/07
'Author: Nico van Zyl
'Date: 2001/07
Dialog.DetachFromExtension(self)
```

```
'Name: CNCINFAdSpatData.Cancel
'Description: calls dlgStart'
'Author:Nico van Zyl
'Date: March 2001
'-----
' clear all dialog controls that are data bound
'-----
```

```
dlg = av.FindDialog("dlgAddSpatialData")

'dlg.SetServer(nil)
for each lbx in dlg.FindByClass(ListBox)
  'Remove linkage to any data sources used by the project
  lbx.Empty
end
for each cbx in dlg.FindByClass(ComboBox)
  'Remove linkage to any data sources used by the project
  cbx.Empty
end
```

DLG.close

av.FindDialog("dlgStart").open

```
'Name : CNCINFCalldlgAddSplData
'The next part is commented out and the call will be made from CNCINFRadioSelect
'Execute on menu Item 'add Spatial data' Click
'This script populates the dlgAddSpatialData dialog's cboNames combobox
```

```
mysql = "SELECT tblFeatures.Names, tblFeatures.FeatureID,
tblFeatures.FeatureTypeID, " +
```

```

        "tblPointsData.FeatureID, tblPointsData.LineID, tblPointsData.PolyID,
tblPointsData.Latitude, " +
        "tblPointsData.Longitude, tblPointsData.ZValue FROM (tblFeatures INNER
JOIN tblPointsData ON " +
        "tblFeatures.FeatureID = tblPointsData.FeatureID) INNER JOIN tblInspections
ON " +
        "tblFeatures.FeatureID = tblInspections.FeatureID WHERE
(((tblPointsData.LineID) Is Null) AND " +
        "((tblPointsData.PolyID) Is Null) AND ((tblPointsData.Latitude) Is Null) AND
" +
        "((tblPointsData.Longitude) Is Null));"

```

```
'msgbox.report(mysql.asstring,"TEST")
```

```
myDSN = "infrastructure"
```

```
theSQL=SQLCon.Find(myDSN)
```

```
myVtab = vtab.makesql(thesql,mysql)
```

```

'Check that at least 1 row was returned
if (myVTab.GetNumRecords < 1) then
    MsgBox.info ("There is no field on database without Spatial Data", "PointsData
Spatial Information")
    return nil
end

```

```
'=====
'This part gets all the info needed to populate the combobox
'=====
```

```
mynamefld = myvtab.findfield("Names")
```

```

dlg = av.FindDialog("dlgAddSpatialData")
'msgbox.info(dlg.getname.asstring, "Debug")

```

```

cbox1 = dlg.findbyname("cboNames")
'msgbox.info(cbox1.getname.asstring, "Debug")

```

```

' Syntax for next line : aComboBox.DefineUniqueFromVTab_
' (aVTab, aField, selectionOnly, caseSensitive, sortAscending)
'=====

```

```
'this part does the work of populating
'=====
```

```

cbox1.DefineUniqueFromVTab(myvtab, mynamefld, false, true, true)
cbox1.select

```

```
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
```

```
'Name: CNCINFComboSqlWriter
```

```
'Description: This script uses all the variables gathered from  
'           the combo boxes and creates a SQL from it  
'  
'
```

```
'Nico van Zyl
```

```
'2001 Januarie
```

```
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
```

```
'myType = self.get(0)  
'myCondition = self.get(1)  
'myReserveName = self.get(2)  
'myInspectBy = self.get(3)  
'myStartDate = self.get(4)  
'myEndDate = self.get(5)
```

```
myDialog = av.FindDialog("dlgSelectData")  
PropDialog = av.FindDialog("dlgProperties")  
myType = myDialog.FindByName("cboType").GetCurrentValue  
myCondition = myDialog.FindByName("cboCondition").GetCurrentValue  
myReserveName = myDialog.FindByName("cboReserveName").GetCurrentValue  
myInspectBy = myDialog.FindByName("cboInspectedBy").GetCurrentValue  
myProperty1 = PropDialog.FindByName("cboProperty1").GetCurrentValue  
myProperty2 = PropDialog.FindByName("cboProperty2").GetCurrentValue  
myProperty3 = PropDialog.FindByName("cboProperty3").GetCurrentValue  
myProperty4 = PropDialog.FindByName("cboProperty4").GetCurrentValue  
myProperty5 = PropDialog.FindByName("cboProperty5").GetCurrentValue
```

```
userCriteria = 0
```

```
mySQL = "SELECT tblType.Type, tblType.FeatureTypeID,  
tblReserves.ReserveName, tblInspections.Condition, " +  
"tblInspections.InspectedBy, tblFeatures.Description, tblFeatures.Names,  
tblFeatures.FeatureID, tblInspections.Date, " +  
"tblPropertyList.PropertyNames AS PropName1, tblFeatures.Property1,  
tblPropertyList_1.PropertyNames AS PropName2, " +  
"tblFeatures.Property2, tblPropertyList_2.PropertyNames AS PropName3,  
tblFeatures.Property3, tblPropertyList_3.PropertyNames AS " +  
"PropName4, tblFeatures.Property4, tblPropertyList_4.PropertyNames AS  
PropName5, tblFeatures.Property5, " +
```

```

"tblSpatialType.SpatialType, tblPointsData.Latitude, tblPointsData.Longitude FROM
((tblSpatialType INNER JOIN " +
"(tblPropertyList RIGHT JOIN (((tblType LEFT JOIN tblPropertyList AS
tblPropertyList_1 ON tblType.PropertyNameID2 = " +
"tblPropertyList_1.PropertyID) LEFT JOIN tblPropertyList AS tblPropertyList_2 ON
tblType.PropertyNameID3 = " +
"tblPropertyList_2.PropertyID) LEFT JOIN tblPropertyList AS tblPropertyList_4 ON
tblType.PropertyNameID4 = " +
"tblPropertyList_4.PropertyID) LEFT JOIN tblPropertyList AS tblPropertyList_3 ON
tblType.PropertyNameID5 = " +
"tblPropertyList_3.PropertyID) ON tblPropertyList.PropertyID =
tblType.PropertyNameID1) ON tblSpatialType.SpatialTypeID = " +
"tblType.SpatialTypeID) INNER JOIN (tblReserves INNER JOIN (tblFeatures LEFT
JOIN tblPointsData ON tblFeatures.FeatureID = " +
"tblPointsData.FeatureID) ON tblReserves.ReserveID = tblFeatures.ReserveID) ON
tblType.FeatureTypeID = " +
"tblFeatures.FeatureTypeID) LEFT JOIN tblInspections ON tblFeatures.FeatureID =
tblInspections.FeatureID " +
"WHERE "

```

```

*****

```

```

mysqlLine = "SELECT tblType.Type, tblType.FeatureTypeID,
tblReserves.ReserveName, tblInspections.Condition, " +
"tblInspections.InspectedBy, tblFeatures.Description, tblFeatures.Names,
tblFeatures.FeatureID, " +
"tblInspections.Date, tblPropertyList.PropertyNames AS PropName1,
tblFeatures.Property1, " +
"tblPropertyList_1.PropertyNames AS PropName2, tblFeatures.Property2,
tblPropertyList_2.PropertyNames AS " +
"PropName3, tblFeatures.Property3, tblPropertyList_3.PropertyNames AS
PropName4, tblFeatures.Property4, " +
"tblPropertyList_4.PropertyNames AS PropName5, tblFeatures.Property5,
tblSpatialType.SpatialType, " +
"tblPointsData.Latitude, tblPointsData.Longitude, tblLinesData.lineID FROM
((tblSpatialType INNER JOIN " +
"(tblPropertyList RIGHT JOIN (((tblType LEFT JOIN tblPropertyList AS
tblPropertyList_1 ON " +
"tblType.PropertyNameID2 = tblPropertyList_1.PropertyID) LEFT JOIN
tblPropertyList AS tblPropertyList_2 ON " +
"tblType.PropertyNameID3 = tblPropertyList_2.PropertyID) LEFT JOIN
tblPropertyList AS tblPropertyList_4 ON " +
"tblType.PropertyNameID4 = tblPropertyList_4.PropertyID) LEFT JOIN
tblPropertyList AS tblPropertyList_3 ON " +
"tblType.PropertyNameID5 = tblPropertyList_3.PropertyID) ON
tblPropertyList.PropertyID = " +
"tblType.PropertyNameID1) ON tblSpatialType.SpatialTypeID =
tblType.SpatialTypeID) INNER JOIN " +
"(tblReserves INNER JOIN ((tblFeatures INNER JOIN tblLinesData ON
tblFeatures.FeatureID = " +

```

```
"tblLinesData.featureID) INNER JOIN tblPointsData ON tblLinesData.lineID =
tblPointsData.LineID) ON " +
"tblReserves.ReserveID = tblFeatures.ReserveID) ON tblType.FeatureTypeID =
tblFeatures.FeatureTypeID) " +
"LEFT JOIN tblInspections ON tblFeatures.FeatureID = tblInspections.FeatureID
" +
"WHERE "
```

```
*****
```

```
mySqlPolygon = "SELECT tblType.Type, tblFeatures.Names,
tblFeatures.Description, tblReserves.ReserveName, " +
"tblInspections.Condition, tblInspections.InspectedBy, tblFeatures.FeatureID,
tblType.FeatureTypeID, " +
"tblInspections.Date, tblPropertyList.PropertyNames AS PropName1,
tblFeatures.Property1," +
"tblPropertyList_1.PropertyNames AS PropName2, tblFeatures.Property2,
tblPropertyList_2.PropertyNames AS " +
"PropName3, tblFeatures.Property3, tblPropertyList_3.PropertyNames AS
PropName4, tblFeatures.Property4, " +
"tblPropertyList_4.PropertyNames AS PropName5, tblFeatures.Property5,
tblSpatialType.SpatialType, " +
"tblPointsData.Latitude, tblPointsData.Longitude, tblpoly.PolyID FROM
((tblSpatialType INNER JOIN (tblPropertyList " +
"RIGHT JOIN (((tblType LEFT JOIN tblPropertyList AS tblPropertyList_1 ON
tblType.PropertyNameID2 = " +
"tblPropertyList_1.PropertyID) LEFT JOIN tblPropertyList AS tblPropertyList_2 ON
tblType.PropertyNameID3 = " +
"tblPropertyList_2.PropertyID) LEFT JOIN tblPropertyList AS tblPropertyList_4 ON
tblType.PropertyNameID4 = " +
"tblPropertyList_4.PropertyID) LEFT JOIN tblPropertyList AS tblPropertyList_3 ON
tblType.PropertyNameID5 = " +
"tblPropertyList_3.PropertyID) ON tblPropertyList.PropertyID =
tblType.PropertyNameID1) ON " +
"tblSpatialType.SpatialTypeID = tblType.SpatialTypeID) INNER JOIN (tblReserves
INNER JOIN " +
"((tblFeatures INNER JOIN tblPoly ON tblFeatures.FeatureID = tblPoly.FeatureID)
INNER JOIN " +
"tblPointsData ON tblPoly.PolyID = tblPointsData.PolyID) ON
tblReserves.ReserveID = " +
"tblFeatures.ReserveID) ON tblType.FeatureTypeID = tblFeatures.FeatureTypeID)
LEFT JOIN " +
"tblInspections ON tblFeatures.FeatureID = tblInspections.FeatureID " +
"WHERE "
```

```
*****
```

'this part construct the rest of the SQL "WHERE" statement

```
mySqlBld = ""
```

```
'This part looks at the cboType combobox to get the selection
```

```
if((myType = "All") or(myType = nil)) then
```

```
'do nothing
```

```
else
```

```
  usercreteria = 1
```

```
  mySqlBld = mySqlBld + "tblType.Type =" + " " + mytype + ""
```

```
end
```

```
if((mycondition = "all") or (mycondition = nil)) then
```

```
'do nothing
```

```
else
```

```
  usercreteria = usercreteria + 1
```

```
  if(usercreteria > 1) then
```

```
    mySqlBld = mySqlBld + " and " + "condition =" + "" + mycondition + ""
```

```
  else
```

```
    mySqlBld = mySqlBld + "condition =" + " " + mycondition + ""
```

```
  end
```

```
end
```

```
'This part looks at the cboReserveName combobox to get the selection
```

```
if((myReserveName ="all")or(myReserveName = nil)) then
```

```
'do nothing
```

```
else
```

```
  usercreteria = usercreteria + 1
```

```
  if(usercreteria > 1) then
```

```
    mySqlBld = mySqlBld + " and " + "ReserveName =" + "" + myreserveName + ""
```

```
  else
```

```
    mySqlBld = mySqlBld + "ReserveName =" + " " + myreserveName + ""
```

```

    end
end

if((myInspectBy ="all")or(myInspectBy = nil)) then
    'do nothing

else

    usercreteria = usercreteria + 1

    if(usercreteria > 1) then

        mySqlBld = mySqlBld + " AND " + "capturedBy =" + "" + myInspectBy + ""

    else

        mySqlBld = mySqlBld + "InspectedBy =" + " " + myInspectBy + ""
    end

end

=====
' get all the values from the date controls to build a string to send over the ODBC to
DB
=====

myStartDay = myDialog.findByName("cboStartDay").getCurrentValue
myStartMonth = myDialog.findByName("cboStartMonth").getCurrentValue
myStartYear = myDialog.findByName("txtStartYear").getText
myEndDay = myDialog.findByName("cboEndDay").getCurrentValue
myEndMonth = myDialog.findByName("cboEndMonth").getCurrentValue
myEndYear = myDialog.findByName("txtEndYear").getText

myStartDate = "#" + myStartYear.asstring + "/" + myStartMonth.asstring + "/" +
myStartDay.asstring + "#"
myEndDate = "#" + myEndYear.asstring + "/" + myEndMonth.asstring + "/" +
myEndDay.asstring + "#"

if((myStartYear = nil) or (myStartYear = ""))then
    ' do nothing
else

    usercreteria = usercreteria + 1
    if(usercreteria > 1) then

        mySqlBld = mySqlBld + " AND " + "[Date] between " + myStartDate + " and
" + myEndDate

    else

```

```
        mySqlBld = mySqlBld + " [Date] between " + myStartDate + " and " +
myEndDate

        end

end

if((myProperty1 ="all")or(myProperty1 = nil)) then
'do nothing

else

    usercreteria = usercreteria + 1

    if(usercreteria > 1) then
        mySqlBld = mySqlBld + " AND "+ "tblFeatures.Property1 =" + "" +
myProperty1 + ""

    else

        mySqlBld = mySqlBld + "tblValueList.Value =" + " " + myProperty1 + ""
    end

end

if((myProperty2 ="all")or(myProperty2 = nil)) then
'do nothing

else

    usercreteria = usercreteria + 1

    if(usercreteria > 1) then

        mySqlBld = mySqlBld + " AND "+ "tblFeatures.Property2 =" + "" +
myProperty2 + ""

    else

        mySqlBld = mySqlBld + "tblValueList_1.Value =" + " " + myProperty2 + ""
    end

end

if((myProperty3 ="all")or(myProperty3 = nil)) then
'do nothing

else

    usercreteria = usercreteria + 1
```

```
if(usercreteria > 1) then

    mySqlBld = mySqlBld + " AND "+ "tblFeatures.Property3 =" + "" +
myProperty3 + ""

    else

        mySqlBld = mySqlBld + "tblValueList_2.Value =" + " " + myProperty3 + ""
    end

end

if((myProperty4 ="all")or(myProperty4 = nil)) then
    'do nothing

else

    usercreteria = usercreteria + 1

    if(usercreteria > 1) then

        mySqlBld = mySqlBld + " AND "+ "tblFeatures.Property4 =" + "" +
myProperty4 + ""

        else

            mySqlBld = mySqlBld + "tblValueList_3.Value =" + " " + myProperty4 + ""
        end

    end

if((myProperty5 ="all")or(myProperty5 = nil)) then
    'do nothing

else

    usercreteria = usercreteria + 1

    if(usercreteria > 1) then

        mySqlBld = mySqlBld + " AND "+ "tblFeatures.Property5 =" + "" +
myProperty5 + ""

        else

            mySqlBld = mySqlBld + "tblValueList_4.Value =" + " " + myProperty5 + ""
        end

    end

end
```

```

if (usercriteria = 0) then
  msgbox.warning("At least one combobox must be selected!","Combobox
selection warning")
  return nil
  'av.run("CNCINFComboCombine")
else
end
'=====
'The "CNCINFSqlConnection" script is called in the next part,the Spatial type is
determined
'and split up so that it can be displayed in different themes.
'=====
'point SQL
mySqlPoint = mysql + mySqlBld + " AND tblSpatialType.SpatialType='Point';"
av.run("CNCINFSqlConnection",{mySqlPoint,"Point"})
'Line SQL
mySqlLine =mySqlLine + mySqlBld + " AND tblSpatialType.SpatialType='Line'
ORDER BY tblPointsData.PointID;"
'msgbox.report(mysqlLine.asstring,"test")
'return nil
av.run("CNCINFSqlConnection",{mySqlLine,"Line"})
'Polygon SQL
mySqlPoly = mySqlPolygon + mySqlBld + " AND
tblSpatialType.SpatialType='Polygon' ORDER BY tblPointsData.PointID;"
'msgbox.report(mysqlPoly.asstring,"test")
av.run("CNCINFSqlConnection",{mySqlPoly,"Polygon"})
'-----
'%%%%%%%%%%
'Name: CNCINF DissolvePoint
'Author: Nico van Zyl
'This script sends the points back to the database
'2001/06/06
'%%%%%%%%%%
'-----
' start editing the active Point theme
'-----
active = self.get(0)

myFtab= active.getFtab
myFtab.setEditable(true)

myThemes = av.getactivedoc.getthemes

'myLat = myFtab.findField("Latitude")
'myLon = myFtab.findField("Longitude")
myPID = myFtab.findField("PointID")
myfield = myFtab.FindField ("Shape")

```


'The variables needed to run the script

```
'this is a Temp bit to test the system
'theView = av.GetActiveDoc
'editThm = theView.GetEditableTheme
'active = theView.GetActiveThemes.Get(0)
'mySpatialType = "Line"
Active = self.get(0)
mySpatialType = self.get(1)
```

```
myFtab= active.getFtab
NumRecords = myFtab.GetNumRecords
ShpField = myFtab.FindField("Shape")
NameField = myFtab.FindField("PolyID")
myLineID = myFtab.FindField("LineID")
```

'connect to the database

```
myDSN = "infrastructure"
theSQL=SQLCon.Find(myDSN)
```

```
'SHOW THE STATUS BAR
av.ShowMsg ("Uploading polygon points...")
Cancelled = false
StatusIndex = 0
Count = 0
av.SetStatus (statusIndex)
```

'RETRIEVE THE POLYGON/LINE POINTS

```
ListCounts = List.Make
```

```
'Loop starts here
```

for each rec in myFtab

```
ThePolygon = myFtab.ReturnValue(ShpField,rec)
  TheList = ThePolygon.AsPolyline.AsList
  TheListCount = TheList.Count
  ListCounts.Add(TheListCount)
  ThePoints = TheList.Get(0)
'this counts the number of point in the shape
  ThePointsCount = ThePoints.Count - 1
```

```
'UPDATE THE STATUS BAR
  Count = Count + 1
  StatusIndex = (count/NumRecords)*100
```

```

Cancelled = false
av.ShowStopButton
Continued = av.SetStatus(StatusIndex)
if (not continued) then
  Cancelled = true
  MsgBox.Info("Process Interrupted.", "Stop")
  exit
end

```

'This part DELETES the Polygon records from the database so that you can add the new data without any trouble

```

if (mySpatialType = "Polygon") then
  NameField = myFtab.FindField("PolyID")

  'for each rec in myFtab

  Polygon_ID = myFtab.ReturnValue(NameField,rec)'sit PolyID hier in

  mySQL = "DELETE tblPointsData.*, tblPoly.FeatureID FROM tblPoly
INNER JOIN tblPointsData " +
  "ON tblPoly.PolyID = tblPointsData.PolyID WHERE (((tblPoly.PolyID)= "
+ Polygon_ID.asstring + "));"
  'msgbox.report(mySQL.asstring, "CNC.DBPolygon.dissolve - Delete -
debug")

  theSQL.executeSQL(mySQL.asstring)

  'RETRIEVE THE POLYGON POINTS
  for each pnt in 0..(ThePointsCount-1) 'av stores first and last vertex as the
same point,
  'so leave out last vertex
  ThePoint = ThePoints.Get(pnt)
  YCoord = ThePoint.GetY.AsString
  XCoord = ThePoint.GetX.AsString
  'myPolyType = polyFtab.ReturnValue(TypeField,rec) 'get the polygon
type

  'UPDATE THE STATUS BAR
  Count = Count + 1
  StatusIndex = (count/ThePointsCount)*100
  Cancelled = false
  av.ShowStopButton
  Continued = av.SetStatus(StatusIndex)
  if (not continued) then
    Cancelled = true
    MsgBox.Info("Process Interrupted.", "Stop")
    exit
  end

```

```
=====
'WRITE THE POLYGON DATA straight through to the table across the odbc link
=====
```

```

        mySQL = "INSERT INTO tblPointsData ( Latitude, Longitude, PolyID
) SELECT " + YCoord.asstring +
        " AS Expr1, " + XCoord.asstring + " AS Expr2, " + Polygon_ID.asstring
+" AS Expr3;"

```

```

        'msgbox.report(mySQL.asstring, "CNC.DBPolygon.dissolve - Update -
debug")
        theSQL.executeSQL(mySQL.asstring)
    end

```

```
=====
'This part DELETES the Line records from the database so that you can add the new
data without any trouble
=====
```

```
elseif (mySpatialType = "Line") then
```

```
    'for each rec in myFtab
```

```
        myLineIDVal = myFtab.returnValue(myLineID,rec)
```

```

        mySql = "DELETE tblPointsData.* FROM tblPointsData WHERE
tblPointsData.LineID= "+ myLineIDVal.asstring + ";"
        'msgbox.report(mySQL.asstring, "CNC.DBPolygon.dissolve - Delete -
debug")

```

```
        theSQL.executeSQL(mySQL.asstring)
```

```
=====
'RETRIEVE THE LINE POINTS
=====
```

```
count = 0
```

```
    for each pnt in 0..(ThePointsCount)
```

```
        ThePoint = ThePoints.Get(pnt)
```

```
        YCoord = ThePoint.GetY.AsString
```

```
        XCoord = ThePoint.GetX.AsString
```

```
        'myPolyType = polyFtab.ReturnValue(TypeField,rec) 'get the polygon type
```

```
        'Polygon_ID = myFtab.ReturnValue(LineNameField,pnt)'sit PolyID hier in
```

```

link
        'WRITE THE LINE DATA straight through to the table across the odbc

```



```
'variables
```

```
theView = av.GetActiveDoc
active = theView.GetActiveThemes.Get(0)
myFtab= active.getFtab
NumRecords = myFtab.GetNumRecords
ShpField = myFtab.FindField("Shape")
NameField = myFtab.FindField("Trail_name")
myDescription = myFtab.FindField("Descriptio")
```

```
'SHOW THE STATUS BAR
```

```
av.ShowMsg ("Uploading Line points...")
Cancelled = false
StatusIndex = 0
Count = 0
av.SetStatus (statusIndex)
```

```
'=====
```

```
'connect to the database
```

```
'=====
```

```
myDSN = "infrastructure"
theSQL=SQLCon.Find(myDSN)
```

```
ListCounts = List.Make
```

```
'=====
```

```
'Loop starts here
```

```
'=====
```

```
for each rec in myFtab
```

```
ThePolygon = myFtab.ReturnValue(ShpField,rec)
```

```
  TheList = ThePolygon.AsPolyline.AsList
```

```
  TheListCount = TheList.Count
```

```
  ListCounts.Add(TheListCount)
```

```
  ThePoints = TheList.Get(0)
```

```
'this counts the number of point in the shape
```

```
  ThePointsCount = ThePoints.Count - 1
```

```
  myNamesVal = myFtab.returnValue(NameField,rec)
```

```
  myDescriptionVal = myFtab.returnValue(myDescription,rec)
```

```
  mySQL = "INSERT INTO tblFeatures ( Names, Description) SELECT " +
myNamesVal.asstring +
```

```
  " AS Expr1, " + myDescriptionVal.asstring + " AS Expr2;"
```

```
  'msgbox.report(mySQL.asstring, "CNC.DBLine.dissolve - Update - debug")
```

```
  theSQL.executeSQL(mySQL.asstring)
```

```
  mySQL2 = "INSERT INTO tblLinesData ( featureID ) SELECT
tblFeatures.FeatureID FROM tblFeatures " +
```

```

"WHERE tblFeatures.Names=" + myNamesVal.asstring + " and
tblFeatures.Description = " +
myDescriptionVal.asstring +";"
'msgbox.report(mySQL2.asstring, "CNC.DBLine.dissolve - Update - debug")
theSQL.executeSQL(mySQL2.asstring)

count = 0
for each pnt in 0..(ThePointsCount)
    ThePoint = ThePoints.Get(pnt)
    YCoord = ThePoint.GetY.AsString
    XCoord = ThePoint.GetX.AsString
    'myPolyType = polyFtab.ReturnValue(TypeField,rec) 'get the polygon type
    'Polygon_ID = myFtab.ReturnValue(LineNameField,pnt)'sit PolyID hier in

    'WRITE THE LINE DATA straight through to the table accross the odbc
link

    mySQL3 = "SELECT tblLinesData.lineID FROM tblFeatures INNER
JOIN tblLinesData ON " +
    "tblFeatures.FeatureID = tblLinesData.featureID WHERE
tblFeatures.Names=" +
    myNamesVal.asstring + " and tblFeatures.Description = " +
myDescriptionVal.asstring +";"

    myVtab = vtab.makesql(thesql,mysql3)
    myLineID = myVtab.Findfield("LineID")
    TheLineID = myvtab.returnValue(myLineID,0)

    mySQL4 = "INSERT INTO tblPointsdata ( LineID, Latitude,
Longitude ) SELECT " + TheLineID.asstring +
    " AS Expr1, " + YCoord.asstring + " AS Expr2, " + XCoord.asstring
+ " AS Expr3;"
    theSQL.executeSQL(mySQL4.asstring)

    'theSQL.executeSQL(mySQL4.asstring)
    'UPDATE THE STATUS BAR
    Count = Count + 1
    StatusIndex = (count/ThePointsCount)*100
    Cancelled = false
    av.ShowStopButton
    Continued = av.SetStatus(StatusIndex)
    if (not continued) then
        Cancelled = true
        MsgBox.Info("Process Interrupted.", "Stop")
    exit
end

```

end

end

'%%'

'Name: **CNCINF DissolveThemePoint**

'Author :Nico van Zyl

'This script sends The Name, description and the XYcoords over ODBC for a
'Points theme.

'The %%% POINTS %%% theme must be selected!

'The 'Place_Name' and 'Descriptio' fields need to be Unique, TOGETHER, to upload
' the correct FID with each point.

'to the Infrastructure DB

'date: 2001/07/12

'%%'

'=====

'get the active Point theme in active view

'=====

theView = av.GetActiveDoc

active = theView.GetActiveThemes.Get(0)

myFtab= active.getFtab

myCount = myFtab.GetNumRecords

myPlaceName = myFtab.FindField("Place_Name")

myDescription = myFtab.FindField("Descriptio")

myfield = myFtab.FindField ("Shape")

'=====

'connect to the database

'=====

myDSN = "infrastructure"

theSQL=SQLCon.Find(myDSN)

'SHOW THE STATUS BAR

av.ShowMsg ("Uploading points...")

Cancelled = false

StatusIndex = 0

Count = 0

av.SetStatus (statusIndex)

'loop through the recs and get the Lat, long and FID values

for each Rec in myFtab

```
myShapeVal = myftab.returnvalue(myField,rec)
myXval= myShapeVal.GetX
myYval= myShapeVal.Gety
```

```
myPlaceNameVal = myFtab.returnValue(myPlaceName,rec)
myDescriptionVal = myFtab.returnValue(myDescription,rec)
```

```
mySQL = "INSERT INTO tblFeatures ( Names, Description) SELECT " +
myPlaceNameval.asstring +
    " AS Expr1, " + myDescriptionval.asstring + " AS Expr2;"
'msgbox.report(mySQL.asstring, "CNC.DBLine.dissolve - Update - debug")
theSQL.executeSQL(mySQL.asstring)
```

```
mySQL2 = "Select tblfeatures.FeatureID from tblFeatures Where
tblFeatures.Names =" + myPlaceNameVal.asstring + " and tblFeatures.Description =
" + myDescriptionVal.asstring +";"
'msgbox.report(mySQL2.asstring, "CNC.DBLine.dissolve - Update - debug")
myVtab = vtab.makesql(thesql,mysql2)
myFeatureID = myVtab.Findfield("FeatureID")
theFeatureID = myvtab.returnValue(myFeatureID,0)
```

```
mySQL3 = "INSERT INTO tblPointsdata ( FeatureID, Latitude, Longitude )
SELECT " + theFeatureID.asstring +
    " AS Expr1, " + myYval.asstring + " AS Expr2, " + myXval.asstring + " AS
Expr3;"
theSQL.executeSQL(mySQL3.asstring)
```

```
'UPDATE THE STATUS BAR
Count = Count + 1
StatusIndex = (count/myCount)*100
Cancelled = false
av.ShowStopButton
Continued = av.SetStatus(StatusIndex)
if (not continued) then
    Cancelled = true
    MsgBox.Info("Process Interrupted.", "Stop")
    exit
end
```

end

```
#####
'#
'# Author: Tim Sutton suttont@cncjnk.wcape.gov.za
'# Western Cape Nature Conservation Board
'# Scientific Services Division - GIS Section
'# Addapted by Nico van Zyl for Infra Tool
```

```
'# 2001 February
```

```
#####
```

```
' Name:CNCINFAdSpaDatReturnUserPoint
```

```
' Title:Obtain the coords of a point defined by clicking in view and display in  
dlgAddSpatialData dialog
```

```
' Topic:View document
```

```
' Detailed Description:
```

```
'-----
```

```
'Obtain the coords of a point defined by clicking in view. and display in  
dlgAddSpatialData dialog
```

```
'-----
```

```
'-----
```

```
' updates the x & y textbox values in dlgAddSpatialData
```

```
'-----
```

```
#####
```

```
'#
```

```
'# End of Header Section
```

```
'#
```

```
#####
```

```
p = av.getactivedoc.getprojection
```

```
myView = av.GetactiveDoc
```

```
myDisplay = myView.GetDisplay
```

```
myPoint = myDisplay.ReturnUserPoint
```

```
myUnprjPoint = myPoint.ReturnUnProjected(p)
```

```
aMrk = Symbol.Make(#SYMBOL_MARKER)
```

```
'msgbox.report(myUnprjPoint.asstring,"TEST")
```

```
myDisplay.DrawPoint (mypoint, aMrk)
```

```
'this can not be used because the view may be to big to re-display (time consuming)
```

```
'myview.draw(myDisplay)
```

```
'mythemes = myview.getActiveThemes
```

```
'mytheme = mythemes.get(0)
```

```
'mytheme.setvisible(false)
```

```
'mytheme.setvisible(true)
```

```

myPoint = myUnprjPoint
myX = myPoint.Getx
myY = myPoint.getY
'update the dialog x & y control contents
myDLG = av.FindDialog("dlgAddSpatialData")
myDLG.findbyName("txtLat").setText(myY.asstring)
myDLG.findbyName("txtLon").setText(myX.asstring)

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
'Name: CNCINFdlgDataSelect.Quit
'Author: Nico van Zyl
'Date: 2001/02
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dlg = av.findDialog("dlgSelectData")

```

```
dlg.close
```

```
dlg2 = av.findDialog("dlgStart")
```

```
dlg2.open
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
'Name: CNCINFDlgInfraDatOpen
'Nico van Zyl 2001 January
'This script starts up when the Data selection dialog box is openend.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```
'=====
'Show message in statusbar
'=====
```

```
av.ShowMsg("Populating comboboxes of form...")
av.showStopButton
while(true)

```

```
'=====
'This part gets the Condition, adds it to a list and display it in a combobox
'=====
```

```
mysql = " SELECT distinct tblInspections.Condition FROM tblInspections;"
```

```
myDSN = "infrastructure"
```

```
theSQL=SQLCon.Find(myDSN)
```

```
myVtab = vtab.makesql(thesql,mysql)
```

```

'Check that at least 1 row was returned
if (myVTab.GetNumRecords < 1) then
  MsgBox.warning ("The query or data set contains no data!","Error:
CNCInfrastructure.openSQL")
  return nil
end

mylist = {}
mynamefld = myVtab.findfield("Condition")

  for each rec in myVtab
    myname = myVtab.returnvalue(mynamefld,rec)
    mylist.add(myname)
  end

  mylist.insert("All")

cbox1 = self.findByName("cboCondition")
cbox1.definefromList(mylist)
' Syntax for next line : aComboBox.DefineUniqueFromVTab (aVTab, aField,
selectionOnly, caseSensitive, sortAscending)
'cbox1.DefineUniqueFromVTab(myvtab, mynamefld, false, true, true)
'cbox1.select

usercond2 = cbox1
'=====
'This part gets InspectedBy, adds it to a list and display it in a combobox
'=====

more = av.setstatus(1/5 * 100)

mysql = " SELECT distinct tblInspections.InspectedBy FROM tblInspections;"

myVtab = vtab.makesql(thesql,mysql)

'Check that at least 1 row was returned
if (myVTab.GetNumRecords < 1) then
  MsgBox.warning ("The query or data set contains no data!","Error:
CNCInfrastructure.openSQL")
  return nil
end

mylist = {}
mynamefld2 = myvtab.findfield("inspectedby")

  for each rec in myvtab

```

```

    myname = myvtab.returnvalue(mynamefld2,rec)
    mylist.add(myname)
end
mylist.insert("All")

```

```

cbox2 = self.findByName("cboInspectedBy")
cbox2.definefromList(mylist)

```

```

' Syntax for next line : aComboBox.DefineUniqueFromVTab (aVTab, aField,
selectionOnly, caseSensitive, sortAscending)
'cbox2.defineUniquefromVtab(myvtab, mynamefld2, false, true, true)
'cbox2.select

```

```

usercond4 = cbox2

```

```

'usercond4 = msgbox.choiceasstring(mylist,"select type","inspected by")

```

```

=====
' This part gets Reserve Name, adds it to a list and display it in a combobox

```

```

=====
more = av.setstatus(2/5 * 100)

```

```

mysql = " SELECT distinct tblReserves.ReserveName FROM tblReserves;"

```

```

myVtab = vtab.makesql(thesql,mysql)

```

```

'Check that at least 1 row was returned
if (myVTab.GetNumRecords < 1) then
    MsgBox.warning ("The query or data set contains no data!", "Error:
CNCInfrastructure.openSQL")
    return nil
end

```

```

mylist = {}
mynamefld3 = myvtab.findfield("reserveName")

```

```

for each rec in myvtab
    myname = myvtab.returnvalue(mynamefld3,rec)
    mylist.add(myname)
end
mylist.insert("All")

```

```

cbox3 = self.findByName("cboReserveName")
cbox3.definefromList(mylist)

```

```
' Syntax for next line : aComboBox.DefineUniqueFromVTab (aVTab, aField,
selectionOnly, caseSensitive, sortAscending)
'cbox3.defineUniquefromVtab(myvtab, mynamefld3, false, true, true)
'cbox3.select
```

```
usercond6 = cbox3
```

```
'usercond6 = msgbox.choiceasstring(mylist,"select type","reserve name")
```

```
'=====
'This part gets the TYPE, adds it to a list and display it in a combobox
```

```
'=====
more = av.setstatus(3/5 * 100)
```

```
mysql = " SELECT DISTINCT tblType.Type FROM tblType ORDER BY
tblType.Type;"
```

```
myVtab = vtab.makesql(thesql,mysql)
```

```
'Check that at least 1 row was returned
if (myVTab.GetNumRecords < 1) then
  MsgBox.warning ("The query or data set contains no data!","Error:
CNCInfrastructure.openSQL")
  return nil
end
```

```
mylist = {}
mynamefld4 = myvtab.findfield("type")
```

```
for each rec in myvtab
  myname = myvtab.returnvalue(mynamefld4,rec)
  mylist.add(myname)
end
mylist.insert("All")
```

```
cbox4 = self.findByName("cboType")
```

```
cbox4.definefromList(mylist)
```

```
' Syntax for next line : aComboBox.DefineUniqueFromVTab (aVTab, aField,
selectionOnly, caseSensitive, sortAscending)
'cbox4.defineUniquefromVtab(myVtab, mynamefld4, false, true, true)
'cbox4.select
```

```
usercond = cbox4
```

```

'usercond = msgbox.choiceasstring(mylist,"select type","Type")
'-----
' Right now update start and end date controls
'-----
more = av.setstatus(4/5 * 100)

cbox = self.FindByName("cboStartDay")
cbox.definefromlist({"","01","02","03","04","05","06","07","08","09","10","11","12",
"13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","29",
"30","31"})
cbox = self.FindByName("cboEndDay")
cbox.definefromlist({"","01","02","03","04","05","06","07","08","09","10","11","12",
"13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","29",
"30","31"})
cbox = self.FindByName("cboStartMonth")
cbox.definefromlist({"","01","02","03","04","05","06","07","08","09","10","11","12"
})
cbox = self.FindByName("cboEndMonth")
cbox.definefromlist({"","01","02","03","04","05","06","07","08","09","10","11","12"
})
self.FindByName("txtStartYear").setText("")
self.FindByName("txtEndYear").setText("")
'=====
' This part is added to clear the values from the dlgProperties dialog so that the
' value do not affect the query if you used it before
'=====

myDialog = av.finddialog("dlgProperties")
myProperty1 = myDialog.findByName("cboProperty1")
myProperty2 = myDialog.findByName("cboProperty2")
myProperty3 = myDialog.findByName("cboProperty3")
myProperty4 = myDialog.findByName("cboProperty4")
myProperty5 = myDialog.findByName("cboProperty5")
myProperty1.SetCurrentValue ("All")
myProperty2.SetCurrentValue ("All")
myProperty3.SetCurrentValue ("All")
myProperty4.SetCurrentValue ("All")
myProperty5.SetCurrentValue ("All")

'=====
' This part passes the variables to the next script (Not in use anymore)
'=====

more = av.setstatus(5/5 * 100)

'av.run("CNCINFComboSqlWriter",{usercond,usercond2,usercond4,usercond6,})
'av.run("CNCINFComboSqlWriter",{usercond,usercond2,usercond4,usercond6,userc
'ond3,usercond5})
more = false
if(not more) then
    break
end

```

```
end
av.ClearWorkingStatus
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
'Name: CNCINFdlgProp
'Author: Nico van Zyl February 2001
'
'This script adds the labels of the dlgProperties dialog interactively
'It is called from the dlgDataSelect dialog
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

=====
'finds the dialog and gets the value from the cboType combobox
=====
dlg = av.FindDialog("dlgInfraDataSelect")

mytype = dlg.findbyName("cboType")

mycbo = mytype.getcurrentvalue

'mytype.select
if (mycbo = nil) then
  av.FindScript("CNCINFOK").open
else

=====
'Close the current dialog
=====

dlg.close

=====
'Opens the new Dialog
=====
av.FindDialog("dlgProperties").open

dlg = av.FindDialog("dlgProperties")

myHeading = dlg.findbyName("lblPropertyHead")

myheading.SetLabel(mycbo + " Properties")
```

```
mysql = "SELECT distinct tblType.PropertyName1, tblType.PropertyName2, " +
"tblType.PropertyName3, tblType.PropertyName4, tblType.PropertyName5,
tblFeatures.Names " +
"FROM tblType INNER JOIN tblFeatures ON tblType.FeatureTypeID =
tblFeatures.FeatureTypeID " +
"WHERE "
```

```
=====
'Looks at the cboType combobox to get the selection and write a SQL statement
=====
```

```
if(mycbo = nil) then
  'do nothing
```

```
else
```

```
    mysql = mysql + "Type =" + "" + mycbo + " ;"
```

```
end
```

```
=====
'makes the connection to the database
=====
```

```
myDSN = "infrastructure"
```

```
theSQL=SQLCon.Find(myDSN)
```

```
myVtab = vtab.makesql(thesql,mysql)
```

```
=====
'gets info from vtab and sticks it in to a variable
=====
```

```
myPropField1 = myVtab.findfield("PropertyName1")
```

```
myPropField2 = myVtab.findfield("PropertyName2")
```

```
myPropField3 = myVtab.findfield("PropertyName3")
```

```
myPropField4 = myVtab.findfield("PropertyName4")
```

```
myPropField5 = myVtab.findfield("PropertyName5")
```

```
=====
'change variable
=====
```

```
dlg = av.FindDialog("dlgProperties")
```

```
=====
'gets label and make variable of it
=====
```

```
myProplbl1 = dlg.findbyname("lblPName1")
```

```
myProplbl2 = dlg.findbyname("lblPName2")
```

```
myProplbl3 = dlg.findbyname("lblPName3")
```

```
myProplbl4 = dlg.findbyname("lblPName4")
```

```
myProplbl5 = dlg.findbyname("lblPName5")
```

'if there is more than one value in the field it loops through them

```
=====
for each rec in myVtab
```

```
    mylbl1 = myvtab.returnValue(myPropfield1,rec).asString
```

```
    mylbl2 = myvtab.returnValue(myPropfield2,rec).asString
```

```
    mylbl3 = myvtab.returnValue(myPropfield3,rec).asString
```

```
    mylbl4 = myvtab.returnValue(myPropfield4,rec).asString
```

```
    mylbl5 = myvtab.returnValue(myPropfield5,rec).asString
```

```
=====
'sets the label to the value from vtab
=====
```

```
    myProplbl1.setlabel(mylbl1)
```

```
    myProplbl2.setlabel(mylbl2)
```

```
    myProplbl3.setlabel(mylbl3)
```

```
    myProplbl4.setlabel(mylbl4)
```

```
    myProplbl5.setlabel(mylbl5)
```

```
break
```

```
end
```

'Without spatial features

```
mySQL = "SELEct tblFeatures.Property1, tblFeatures.Property2,
```

```
tblFeatures.Property3, " +
```

```
"tblFeatures.Property4, tblFeatures.Property5, tblType.Type FROM tblType INNER
```

```
JOIN " +
```

```
"tblFeatures ON tblType.FeatureTypeID = tblFeatures.FeatureTypeID WHERE "
```

```
=====
'Looks at the cboType combobox to get the selection and write a SQL statement
=====
```

```
if(mycbo = nil) then
```

```
    'do nothing
```

```
else
```

```
    mysql = mysql +"Type =" + "" + mycbo + " ;"
```

```
end
```

```
msgbox.report(mysql.asstring,"TEST")
```

```
myVtab = vtab.makesql(thesql,mysql)
```

```

myProp1 = myVtab.findfield("Property1")
myProp2 = myVtab.findfield("Property2")
myProp3 = myVtab.findfield("Property3")
myProp4 = myVtab.findfield("Property4")
myProp5 = myVtab.findfield("Property5")

```

```

myPropcbo1 = dlg.findbyname("cboProperty1")
myPropcbo2 = dlg.findbyname("cboProperty2")
myPropcbo3 = dlg.findbyname("cboProperty3")
myPropcbo4 = dlg.findbyname("cboProperty4")
myPropcbo5 = dlg.findbyname("cboProperty5")

```

```
'for each rec in myVtab
```

```
    'mycbo1 = myvtab.returnValue(myProp1,rec).asstring
```

```
    'mycbo2 = myvtab.returnValue(myProp2,rec).asstring
```

```
    'mycbo3 = myvtab.returnValue(myProp3,rec).asstring
```

```
    'mycbo4 = myvtab.returnValue(myProp4,rec).asstring
```

```
    'mycbo5 = myvtab.returnValue(myProp5,rec).asstring
```

```
=====
'sets the to the value from vtab
=====
```

```
'aComboBox.DefineUniqueFromVTab (aVTab, aField, selectionOnly, caseSensitive,
sortAscending)
```

```

myPropcbo1.DefineUniqueFromVTab(myVtab,myProp1,false,false,false)
myPropcbo2.DefineUniqueFromVTab(myVtab,myProp2,false,false,false)
myPropcbo3.DefineUniqueFromVTab(myVtab,myProp3,false,false,false)
myPropcbo4.DefineUniqueFromVTab(myVtab,myProp4,false,false,false)
myPropcbo5.DefineUniqueFromVTab(myVtab,myProp5,false,false,false)

```

```
end
```

```
'%%%%%%%%%%
```

```
'Script calls CNCINFOK to start Query
```

```
'Name: CNCINFdlgPropertiesOK
```

```
'Author: Nico van Zyl
```

```
'Date 2001/06/21
```

```
'%%%%%%%%%%
```

```
dlg2 = av.findDialog("dlgProperties")
```

```
dlg2.close
```

```
dlg2 = av.findDialog("dlgSelectData")
```

```
dlg2.open
```

```
myActiveDoc = av.getactivedoc
msgbox.info(myActiveDoc.asstring,"debug")
```

```
myvalues = av.run("CNCINFOK",nil)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Name: CNCINFdlgPropertiesOpen
```

```
' This script is executed on the open of 'dlgProperties'.
```

```
' It configures the dialog and calls the script to populate the comboboxes
```

```
,
```

```
' Author: Nico van Zyl
```

```
' Date: 2001/06/18
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
=====
```

```
'Show message in statusbar
```

```
=====
```

```
av.ShowMsg("Populating comboboxes of form...")
```

```
av.showStopButton
```

```
while(true)
```

```
'Variables
```

```
myDialog = av.finddialog("dlgSelectData")
```

```
myType = myDialog.findByName("cboType")
```

```
myTypeValue = myType.GetCurrentValue
```

```
myPropHead = self.findByName("lblPropertyHead")
```

```
=====
```

```
'this part sets the title to display the Type that is selected
```

```
=====
```

```
self.SetTitle ("The Properties of " + myTypeValue)
```

```
=====
```

```
'This part connects to the database and gets the data needed to populate the labels.
```

```
=====
```

```
mySql = "SELECT tblType.Type, tblPropertyList.PropertyNames AS Property1, " +
```

```
"tblPropertyList_1.PropertyNames AS Property2, tblPropertyList_2.PropertyNames
```

```
AS Property3, " +
```

```
"tblPropertyList_3.PropertyNames AS Property4, tblPropertyList_4.PropertyNames
AS Property5 " +
"FROM tblPropertyList RIGHT JOIN (((tblType LEFT JOIN tblPropertyList AS
tblPropertyList_1 ON " +
"tblType.PropertyNameID2 = tblPropertyList_1.PropertyID) LEFT JOIN
tblPropertyList AS " +
"tblPropertyList_2 ON tblType.PropertyNameID3 = tblPropertyList_2.PropertyID)
LEFT JOIN " +
"tblPropertyList AS tblPropertyList_3 ON tblType.PropertyNameID4 =
tblPropertyList_3.PropertyID) " +
"LEFT JOIN tblPropertyList AS tblPropertyList_4 ON tblType.PropertyNameID5 =
tblPropertyList_4.PropertyID) " +
"ON tblPropertyList.PropertyID = tblType.PropertyNameID1 WHERE
tblType.Type=" + myTypeValue.asstring + ";"
```

```
'msgbox.report(mySql,"Debug")
myDSN = "infrastructure"
theSQL=SQLCon.Find(myDSN)
myVtab = vtab.makesql(thesql,mySql)
```

```
'=====
'this is the property variables from the dialog
'=====
```

```
myCombo1 = self.findByName("cboProperty1")
myCombo2 = self.findByName("cboProperty2")
myCombo3 = self.findByName("cboProperty3")
myCombo4 = self.findByName("cboProperty4")
myCombo5 = self.findByName("cboProperty5")
```

```
'=====
'this is the label variables from the dialog
'=====
```

```
myLabel1 = self.findByName("lblPName1")
myLabel2 = self.findByName("lblPName2")
myLabel3 = self.findByName("lblPName3")
myLabel4 = self.findByName("lblPName4")
myLabel5 = self.findByName("lblPName5")
```

```
'=====
'This part sets all the combos and labels invisible before continuing
'=====
```

```
myLabel1.SetVisible (False)
myLabel2.SetVisible (False)
myLabel3.SetVisible (False)
myLabel4.SetVisible (False)
myLabel5.SetVisible (False)
```

```
myCombo1.SetVisible (False)
myCombo2.SetVisible (False)
myCombo3.SetVisible (False)
```

```
myCombo4.SetVisible (False)
myCombo5.SetVisible (False)
```

```
'this is the Vtab property fields variables
myPropfld1 = myVtab.findfield("Property1")
myPropfld2 = myVtab.findfield("Property2")
myPropfld3 = myVtab.findfield("Property3")
myPropfld4 = myVtab.findfield("Property4")
myPropfld5 = myVtab.findfield("Property5")
```

```
'Gets the values in the fields of the database
myPropVal1 = myVtab.returnvalue(myPropfld1,0)
myPropVal2 = myVtab.returnvalue(myPropfld2,0)
myPropVal3 = myVtab.returnvalue(myPropfld3,0)
myPropVal4 = myVtab.returnvalue(myPropfld4,0)
myPropVal5 = myVtab.returnvalue(myPropfld5,0)
```

```
ComboSQL = "SELECT tblPropertyList.PropertyNames, tblValueList.Value FROM
tblPropertyList " +
"INNER JOIN tblValueList ON tblPropertyList.PropertyID =
tblValueList.PropertyNameID WHERE " +
"tblPropertyList.PropertyNames="
```

```
'=====
'The rest of the script tests if it must call CNCINFPopulateCombos and then call it
'=====
```

```
more = av.setstatus(1/5 * 100)

if ((myPropVal1 = nil) or (myPropVal1 = "")) then
    more = av.setstatus(5/5 * 100)
    more = false
    if(not more) then
        av.ClearWorkingStatus

        return nil
    end
end
av.run("CNCINFPopulateCombos", {myPropVal1,theSql,ComboSQL,myCombo1,my
Label1})

if ((myPropVal2 = nil) or (myPropVal2 = "")) then
    more = av.setstatus(5/5 * 100)
    more = false
    if(not more) then
        av.ClearWorkingStatus

        return nil
    end
end
```

```
end
av.run("CNCINFPopulateCombos",{myPropVal2,theSql,ComboSQL,myCombo2,my
Label2})
more = av.setstatus(2/5 * 100)

if ((myPropVal3 = nil) or (myPropVal3 = "")) then
  more = av.setstatus(5/5 * 100)
  more = false
  if(not more) then
    av.ClearWorkingStatus
```

```
  return nil
end
```

```
end
av.run("CNCINFPopulateCombos",{myPropVal3,theSql,ComboSQL,myCombo3,my
Label3})
more = av.setstatus(3/5 * 100)

if ((myPropVal4 = nil) or (myPropVal4 = "")) then
  more = av.setstatus(5/5 * 100)
  more = false
  if(not more) then
    av.ClearWorkingStatus
```

```
  return nil
end
```

```
end
av.run("CNCINFPopulateCombos",{myPropVal4,theSql,ComboSQL,myCombo4,my
Label4})
more = av.setstatus(4/5 * 100)

if ((myPropVal5 = nil) or (myPropVal5 = "")) then
  more = av.setstatus(5/5 * 100)
  more = false
  if(not more) then
    av.ClearWorkingStatus
```

```
  return nil
end
```

```
end
av.run("CNCINFPopulateCombos",{myPropVal5,theSql,ComboSQL,myCombo5,my
Label5})
more = av.setstatus(5/5 * 100)
more = false
if(not more) then
```

```

    break
  end
end
av.ClearWorkingStatus

```

```

'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
Name: CNCINFdlgStart.Quit
'This closes the dialog box
'Nico van Zyl
'April 2001
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
dlg = av.findDialog("dlgStart")

```

```
dlg.close
```

```

'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
Name: CNCINFExit
'Author: Nico van Zyl
'Date: 2001/03
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'

```

```
exit
```

```
self.closeall
```

```

'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
'This script gets the sql for the selected feature without spatial data in a combobox
and 'then gets the spatial point from a user defined point on a view and sends it to the
Access DB.
Name: CNCINFGetSpatialData
'Nico van Zyl
'2001 February
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'

```

```

dlg = av.FindDialog("dlgAddSpatialData")
cbox1 = dlg.findbyname("cboNames")

```

```
'gets value
```

```

mycbo = cbo1.getvalue

'aDisplay.DrawPoint (aPoint, aSymbol)

'construct SQL
' mysql = " SELECT tblFeatures.Names, tblFeatures.FeatureTypeID,
tblpointsdata.featureID, " +
' "tblpointsdata.PointID, tblPointsData.Latitude, tblPointsData.Longitude,
tblPointsData.ZValue " +
' "FROM (tblFeatures INNER JOIN tblPointsData ON tblFeatures.FeatureID =
tblPointsData.PointID) " +
' "INNER JOIN tblInspections ON tblFeatures.FeatureID = tblInspections.FeatureID
WHERE " +
' "tblPointsData.Latitude Is Null AND tblPointsData.Longitude Is Null "

mysql = "SELECT tblFeatures.Names, tblFeatures.FeatureID,
tblFeatures.FeatureTypeID, " +
      "tblPointsData.pointID,tblPointsData.FeatureID, tblPointsData.LineID,
tblPointsData.PolyID, tblPointsData.Latitude, " +
      "tblPointsData.Longitude, tblPointsData.ZValue FROM (tblFeatures INNER
JOIN tblPointsData ON " +
      "tblFeatures.FeatureID = tblPointsData.FeatureID) INNER JOIN tblInspections
ON " +
      "tblFeatures.FeatureID = tblInspections.FeatureID WHERE
tblPointsData.LineID Is Null AND " +
      "tblPointsData.PolyID Is Null AND tblPointsData.Latitude Is Null AND " +
      "tblPointsData.Longitude Is Null "
if (mycbo = nil) then
  'do nothing
  MsgBox.Warning ("There is no data in the Names field." + nl + nl + "It is probably
because there is no records in database without Spatial data", "No Spatial Data")

  exit
else
  mysql = mysql + "AND tblFeatures.Names=" + "" + mycbo + ""

'msgbox.report(mysql.asstring, "TEST")

'make ODBC connection
myDSN = "infrastructure"

theSQL=SQLCon.Find(myDSN)

myVtab = vtab.makesql(thesql,mysql)

end
=====
'now pull the sql data from Access onto a table in Arcview
=====

```

```
mytable = "AddSpatial"
```

```
myTable=Table.Make(myVtab)
av.GetProject.AddDoc(mytable)
```

```
' Show the table
' mytable.Get Win.Open
  myTable.SetName("AddSpatial")
```

```
if (myTable = NIL) then
  return nil
end
```

```
myVTab=myTable.getVtab
myDBName = myTable.GetName
```

```
=====
'It now gets the variables from the txtboxes and the values of the ID fields
' and adds them in objects.
=====
```

```
myfeatureIDfield = myvtab.findfield("featureID")
mypointIDfield = myvtab.findfield("pointID")
'mypointIDval.getvalue(myfeatureIDfield)
myfeatureIDval = myVtab.ReturnValue(myfeatureIDfield,0)
mypointIDval = myVtab.ReturnValue(mypointIDfield,0)
```

```
myDLG = av.FindDialog("dlgAddSpatialData")
mytxtLat = myDLG.findbyName("txtLat")
mytxtLon = myDLG.findbyName("txtLon")
mytxtLatval = mytxtLat.getText
mytxtlonval = mytxtLon.getText
```

```
~~~~~
'This part constructs a SQL from the objects made in the previous section
~~~~~
```

```
mysql = "UPDATE tblPointsData SET tblPointsData.Latitude =" +
mytxtLatval.asstring +
", tblPointsData.Longitude =" + mytxtLonval.asstring + " WHERE
(((tblPointsData.featureID)=" +
myfeatureIDval.asstring + ") AND ((tblPointsData.PointID)=" +
mypointIDval.asstring+ "));"
```

```
'msgbox.report(mysql.asstring, "SqlTest")
```

```
theSQL.executeSQL(mysql)
```

```
'myVTab.SetEditable(true)
```

```
,
```

```
' if (myVTab.IsEditable) then
```

```
'   myVTab.SetValue(myLatfield,0,mytxtLat)
```

```
'   myVTab.SetValue(myLatfield,0,mytxtLat)
```

```
' end
```

```
' myVTab.SetEditable(FALSE)
```

```
~~~~~
```

```
'now it must get the data added from DB
```

```
'add it to the view
```

```
'at the moment I have not decided if it is really necessary
```

```
~~~~~
```

```
=====
```

```
'this is the first one that makes the sql connection
```

```
=====
```

```
mysql = "SELECT tblPointsData.featureID, tblPointsData.Latitude,
```

```
tblPointsData.Longitude, " +
```

```
"tblPointsData.PointID FROM tblPointsData WHERE (((tblPointsData.featureID)="
```

```
+
```

```
myfeatureIDval.asstring + ") AND ((tblPointsData.PointID)=" +
```

```
myPointIDval.asstring + "));"
```

```
'msgbox.report(mysql.asstring, "SqlTest")
```

```
myDSN = "infrastructure"
```

```
theSQL=SQLCon.Find(myDSN)
```

```
'msgbox.report(mysql.asstring, "TEST")
```

```
myVtab = vtab.makesql(thesql,mysql)
```

```
'Check that at least 1 row was returned
```

```
if (myVTab.GetNumRecords < 1) then
```

```
  MsgBox.warning ("The query or data set contains no data!", "Error:  
CNCINFCombosqlWriter")
```

```
  return nil
```

```
end
```

```
=====
```

'now pull the sql data from Access onto a table in Arcview

```
mytable = "Infrastructure"
```

```
myTable=Table.Make(myVtab)
av.GetProject.AddDoc(mytable)
```

```
' Show the table
' mytable.GetWin.Open
myTable.SetName("infrastructure")
```

```
if (myTable = NIL) then
  return nil
end
```

```
myVTab=myTable.getVtab
myDBName = myTable.GetName
```

' Now run an event theme process to add the theme to the current view

```
myxstring = "longitude"
myystring = "latitude"
myXfield = myVTab.Findfield(myXstring)
myYfield = myVTab.Findfield(myYstring)
```

```
myXYsrc = XYname.make(myVtab,myXfield,myYfield)
myNewTheme = theme.make(myXYsrc)
av.getActiveDoc.addTheme(myNewTheme)
```

```
myname = MsgBox.Input ("Type in a name for the new Theme", "Theme
Input", "Theme" )
```

```
myNewTheme.SetName(myname)
```

'This updates the combobox after new data has been added.

```
myObj = "to make the next part work"
av.run("CNCINFCallDlgAddSplData",myObj)
```

```

=====
' Clean Up - remove the event table
=====
myTable = av.GetProject.FindDoc(myDBName)

av.GetProject.RemoveDoc(myTable)
av.PurgeObjects

=====
' All completed
=====
return "OKAY"
=====

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
' Script to install my extension
' Tim Sutton 1998
' Script Name: CNCINFInstallScript
' Adapted by: Nico van Zyl 2001/07/04
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
' The self object for this script is the extension
' The first object for this script is the menu

' Check if there is an active project

if (av.GetProject = nil) then
  return nil
end

=====
' show the splash screen
=====
msgbox.banner("P:\Infrastructure\apr\Dialog_backgrounds\CNCINFInfrasplashGIF.G
IF".asfilename, 5, "")

=====
' Add in the custom menu items
=====

' PROJECT MENU: Add the Infrastructure menu after the Graphics menu

myMenuBar = av.GetProject.FindGui("Project").GetMenuBar
myMenu = myMenuBar.FindByLabel("Theme")

if (myMenu <> nil) then
' myMenuBar.Add(self.Get(0), myMenuBar.GetControls.Find(myMenu))
' else

```

```
' myMenuBar.Add(self.get(0), 999)
'end
```

```
' VIEW MENU: Add the CNC Utilities menu after the CNC Dbases menu
```

```
myMenuBar = av.GetProject.FindGui("View").GetMenuBar
myMenu = myMenuBar.FindByLabel("Graphics")
```

```
if (myMenu <> nil) then
  myMenuBar.Add(self.Get(0), myMenuBar.GetControls.Find(myMenu))
else
  myMenuBar.Add(self.get(0), 999)
end
```

```
' VIEW MENU: Add the CNC Utilities menu after the Field menu
'
```

```
'myMenuBar = av.GetProject.FindGui("Table").GetMenuBar
' myMenu = myMenuBar.FindByLabel("Field")
'
```

```
'if (myMenu <> nil) then
' myMenuBar.Add(self.Get(2), myMenuBar.GetControls.Find(myMenu))
' else
' myMenuBar.Add(self.get(2), 999)
'end
```

```
'DataSetup = av.run("CNC.ODBC.hidemenu", "")
'HideMenu = av.run("CNCutilities.AdvancedToggle", "") 'will turn on when problems
are sorted out
'AddIcons = av.run("CNCutilities.ButtonsMake", "")
'Msgbox.Info("CNC Utilities Extension Loaded. This extension was written by Tim
Sutton. " +
""You are free to use, modify or copy this extension, although no warranty or fitness
to " +
""purpose is either expressed or implied. Use entirely at your own risk.", "CNC
Utilities Extension")
```

```
'%%%%%%%%%%
'This script connects a theme's Ftab in a view to a table in Arcview
'requirements: The view must be active with the theme
'Script Name: CNCINFJoinFtabVtab
'Creator: Nico van Zyl
'2001/05/25
'klasiel@yahoo.com
'%%%%%%%%%%
'Passed variables
  myShpThemeName = self.get(0)
  myTableName = self.get(1)
```

```

'-----
'This part gets the Ftab
'-----
  myDoc = av.getactivedoc 'the view with the theme in must be active

  shpFtab = mydoc.FindTheme (myShpThemeName)
  myShpFtab = shpFtab.getFtab
'-----
'This part gets the Vtab
'-----
  myTable = av.getProject.findDoc(myTableName)
  myVtab = myTable.getvtab
'-----
'This part makes the tables notEditable
'(Join don't want tables to change after join)
'-----
  myshpFtab.SetEditable(false)
  myVtab.SetEditable(false)
'-----
'Find the fields to use in join
'-----
  FID = myshpFtab.findField("FID")
  FeatureID = myVtab.findField("FeatureID")

'Remove any existing joins on the theme ftab
  myshpftab.UnjoinAll

'-----
'Joining the two tables
'-----
  myshpftab.join(FID,myVtab,FeatureID)

'-----
'Clean up any copies of the working tables
'-----
  myDoc = av.findDoc(mytableName)
  if (myDoc <> nil) then
  av.GetProject.RemoveDoc(myDoc)
  end

```

```

'%%%%%%%%%%
' Script Name: CNCINFLoad
' Function: Carry out any tasks needed the first time the extension is loaded
' Author: Nico van Zyl 2001
'%%%%%%%%%%
'nothing at the moment

```

```

'%%%%%%%%%%

```

Name: **CNCINFOK**

'This scripts does some error checking before it runs the "CNCINFComboSqlWriter"

'script

,

'Author : Nico van Zyl

'Date: February 2001

'%%%

dlg1 = av.findDialog("dlgSelectData")

dlg1.open

cbox = dlg1.findByName("cboType")

myType = cbox.GetCurrentValue

=====

' Get the active document, which should be a view. Bail if not a view:

=====

theView=av.GetActiveDoc

if (theView.Is(View).Not) then

MsgBox.Warning("You must select a view before Pressing OK", "Select View
Warning")

return nil

end

"do some error checking

,

'if (myType = nil) then

' return nil

'end

,

"This gets the values from the comboboxes and make them variables

,

'cbox = self.GetDialog.findByName("cboCondition")

'myCondition = cbox.GetCurrentValue

,

"do some error checking

,

'if (myCondition = nil) then

' return nil

'end

,

"This gets the values from the comboboxes and make them variables

,

'cbox = self.GetDialog.findByName("cboReserveName")

'myReserveName = cbox.GetCurrentValue

,

"do some error checking

,

'if (myReserveName = nil) then

' return nil

```

'end
,
"This gets the values from the comboboxes and make them variables
,
'cbox = self.GetDialog.findByName("cboInspectedBy")
'myInspectedBy = cbox.GetCurrentValue
,
"do some error checking
,
'if (myInspectedBy = nil) then
' return nil
'end

=====
' Check if the date fields entered are valid / populated
=====

myStartDay = dlg1.findByName("cboStartDay").GetCurrentValue

myStartMonth = dlg1.findByName("cboStartMonth").GetCurrentValue
myStartYear = dlg1.findByName("txtStartYear").getText
myEndDay = dlg1.findByName("cboEndDay").GetCurrentValue
myEndMonth = dlg1.findByName("cboEndMonth").GetCurrentValue
myEndYear = dlg1.findByName("txtEndYear").getText

myStartDate = myStartYear.asstring + myStartMonth.asstring + myStartDay.asstring
myEndDate = myEndYear.asstring + myEndMonth.asstring + myEndDay.asstring

'myList = {myStartDate,myEndDate, myType, myCondition, myReserveName,
myInspectedBy}
'msgbox.listasstring(myList,"Debug","Debug")

if ((myStartDate.count < 8) or (myEndDate.count < 8)) then
  myAnswer = msgbox.yesno("The date ranges you have entered are invalid. Would
you like to continue without using a date range?", "Individual Extension", true)
  if (myAnswer = false) then
    return nil
  else
    myStartDate = nil
    myEndDate = nil

  end
end

=====
'calls the CNCINFComboSqlWriter script
=====

```

```
'myvalues =
av.run("CNCINFComboSqlWriter",{myType,myCondition,myReserveName,myInspectedBy,myStartDate,myEndDate})

av.run("CNCINFComboSqlWriter",nil)
```

```
'Name: CNCINFopendlgProperties
'This script opens the dlgProperties form
'author : Nico van Zyl
'date"2001/04/28
```

```
myDialog = av.finddialog("dlgSelectData")
myType = myDialog.findByName("cboType")
myTypeValue = myType.GetCurrentValue
```

```
if (myTypeValue = "All") then
  msgbox.Warning("You must select a value in Dropdownlist 'Type' to display the
properties." + nl + nl + " ' All ' is not sufficient" ,"Warning")
  return nil
end
```

```
dlg = av.findDialog("dlgProperties")
```

```
dlg.open
```

```
'Name: CNCINFpointselect
'This script gets the userpoint from a view and sticks it in the textbox provided for
them
'
'Nico van Zyl March 2001
'
```

```
myView = av.GetactiveDoc
myDisplay = myView.GetDisplay
myPoint = myDisplay.ReturnUserPoint
```

```

myTheme = myView.getActiveThemes.get(0)
if (myTheme.getFtab.returnValueString(myTheme.getFtab.findField("Shape"),1) <>
"Polygon") then
  return nil
end

```

```

if (myTheme.CanSelect) then
  myTheme.SelectByPoint(myPoint, #VTAB_SELTYPE_NEW)
else
  return nil
end

```

```

myBitmap = myTheme.getFtab.getSelection

```

```

'check that at least 1 record is selected

```

```

if (myBitmap.count > 0) then

```

```

  for each rec in myBitmap
    myFld = myTheme.getFtab.findField("Shape")
    myShape = myTheme.getFtab.returnValue(myFld,rec)
  end
else
  return nil
end

```

```

myPoint = myShape.ReturnCenter

```

```

'myUnprjPoint = myPoint.ReturnUnProjected(p)

```

```

myPoint = myUnprjPoint

```

```

'gets the point and puts them in the textbox

```

```

myX = myPoint.Getx

```

```

myY = myPoint.getY

```

```

'update the dialog x & y control contents

```

```

myDLG = av.FindDialog("dlgGetCoord")

```

```

myDLG.findbyName("txtLat").setText(myY.asstring)

```

```

myDLG.findbyName("txtLon").setText(myX.asstring)

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

Name: CNCINFPopulateCombos

```

```

This script is called from CNCINFdlgPropertiesOpen

```

```

Author: Nico van Zyl

```

```

Date: 2001/06/20

```

```

'

```

```

'Tasks script do:

```

```

' Unhides Labels

```

```

' Unhide Comboboxes

```

```

' Makes a SQL connection to DB

```

```

' sets the Labels to a value from DB

```

' It Populates comboboxes

'%%%

'=====

'passed Variables form "CNCINFPropertiesOpen"

'=====

```
myPropVal = self.get(0)
theSql = self.get(1)
comboSql = self.get(2)
myValueCbo = self.get(3)
myLabel = self.get(4)
```

' This parts makes the label and Combobox visible if it is being used

'=====

```
myLabel.SetVisible (True)
myValueCbo.SetVisible (True)
myLabel.SetLabel (myPropVal.asstring)
```

' this part makes the SQL statement to get the data to populate combobox

'=====

```
mySql = comboSQL + myPropVal.asstring + "";"
'send SQL to DB
myVtab = vtab.makesql(thesql,mysql)
```

'Check that at least 1 row was returned
if (myVTab.GetNumRecords < 1) then

```
    MsgBox.warning ("The query or data set contains no data!", "Error:
CNCInfrastructure.openSQL")
    return nil
end
```

'clean the list

```
mylist = {}
'find the field to make the data list from
myValuefld = myvtab.findfield("Value")
```

' go through the records in Vtab to stick the values in a list

'=====

```
for each rec in myvtab
    myValue = myvtab.returnvalue(myValuefld,rec)
    mylist.add(myValue)
end
```

' insert the values in a ComboBox

'=====

```
mylist.insert("All")
myValueCbo.definefromList(mylist)
```

'=====

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
'Name: CNCINFRadioSelect
```

```
' This script opens the dialog the user clicked from the radio buttons
```

```
,
```

```
'Nico van Zyl March 2001
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
' close this dialog and open the next one
```

```
Infrastructuredlg = ""
```

```
'Find out which radio button is selected
```

```
dlg = av.FindDialog("dlgStart")
```

```
if (dlg.FindByName("Radio1").isSelected = true) then
```

```
  Infrastructuredlg = "dlginfraDataSelect"
```

```
end
```

```
if (self.GetDialog.FindByName("Radio2").isSelected = true) then
```

```
  Infrastructuredlg = "dlgAddSpatialData"
```

```
end
```

```
if (self.GetDialog.FindByName("Radio3").isSelected = true) then
```

```
  Infrastructuredlg = "edit data"
```

```
end
```

```
'now close this dialog and call the open script for the next
```

```
'Close up current dialog
```

```
self.getDialog.close
```

```
dlg = av.findDialog(Infrastructuredlg)
```

```
dlg.open
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
'Name: CNCINFSaveEdits
```

```
'This script saves and stop editing the data when you click the "Save and Stop Edit"
```

```
'button in the "dlgEditData" dialog. It calls other scripts"CNCINFDissolvePoint" and
```

```
""CNCINFDissolvePolygon/Line" to dissolve the data.
```

```
'Nico van Zyl
```

```
'March 2001
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
'Variables needed to run this script
```

```
theView = av.GetActiveDoc
```

```
editThm = theView.GetEditableTheme
```

```
Active = theView.GetActiveThemes.Get(0)
```

```

myFtab= active.getFtab
polyFtab = myFtab
SpatialType = PolyFtab.findField("SpatialTyp")
mySpatialType = polyFtab.ReturnValue(SpatialType,0)
if (myspatialType = nil) then
  msgbox.info("The active theme is most likely not in edit mode. Select the theme
from the 'Editing Tool' Listbox", "Editing Warning")
  return nil
end

if (editThm <> nil) then
  'we need to stop editing this theme

  doSave = MsgBox.YesNoCancel("Save Edits to "+editThm.GetName+
                              "?", "Stop Editing", true)
  if (doSave = nil) then
    return nil
  end

  if (editThm.StopEditing(doSave).Not) then
    'save failed, remain editing this theme
    MsgBox.Info ("Unable to Save Edits to "
                + editThm.GetName +
                ", please use the Save Edits As option", "")
    return nil

  end

  'save succeeded
  theView.SetEditableTheme(NIL)

  if (editThm = active) then

    ' user wanted to stop editing the active theme, were done

  end
else
  MsgBox.Info ("There is no theme being edited at the moment", "Edit Warning")
  return nil

end

if (active.GetFTab.IsBeingEditedWithRecovery) then

  'user wants to edit the active theme in the view, but its
  'table doc is already being edited - force the
  'user to stop editing the table

  doSave = MsgBox.YesNoCancel("Save Edits to the table for "+

```

```

active.GetName+"?", "Stop Editing", True)

if (doSave = nil) then
    return nil
end

if (active.GetFTab.StopEditingWithRecovery(doSave).Not) then
    MsgBox.Info ("Unable to Save Edits, please use the Save Edits As option", "")
    return nil 'unable to save, remain editing
end
end
'=====
' The scripts that are called in this part is desolving the features and sending it back to
the DB.
'=====

if (mySpatialType = "Point")then
    av.run("CNCINFDissolvePoint",{Active,mySpatialType})
else
    av.run("CNCINFDissolvePolygon/Line",{Active,mySpatialType})
end

'-----
' This part is not used anymore. With this method the edited view is not refreshed and
the
' changes is not vissible. The method of using more than one temp file is used. The
cleaning
' is done by the SetScratch command.
'=====
' This part deletes all the files assosiated with Shape files. This prosess is done to keep
' the directory clean from temp files.
'=====

'myName = active.asstring.asFileName
'theView.DeleteTheme(active)
'file.delete(myName)
,

'mysrcFileName = "$home\Edit.shp".asFileName
'destFileName = filename.Make ("$home\BackupEdit.shp")
'defaultName = FileName.Make("$HOME").MakeTmp("BackupEdit", "shp")
'destFileName = FileDialog.Put( defaultName, "*.shp", "New Fire Shape File" )
,
' if (destFileName = nil) then
'     exit
' end
'mySrcName = srcName.make(destFileName.asstring)
,
'myFullName = destFileName.GetName

```

```

'mysrcName = myFullName.AsSrcName
'
'
'File.Copy (mysrcFileName, destFileName)
'
'myFullName = destFileName.GetFullName
'n = srcname.make(mysrcName)
'theTheme = Theme.Make(n)
'theView.AddTheme (theTheme)
'myDialog = av.FindDialog("dlgEditData")
'myTheme = myDialog.findByName("cbo ThemeList").getCurentValue
'theView.AddTheme (myTheme)
'
'myTheme.SetName (mySpatialType.asstring)
'
'myTheme.SetVisible (true)
'myFtabT = myTheme.getFtab
'myThemes = av.GetActiveDoc.GetThemes
'myFtabT.Flush
'myFtabT.refresh
'
'-----
'=====
' This part Refresh the view
'=====
myThemes = av.GetActiveDoc.GetThemes

for each t in myThemes

    myFtab = t.getFtab
    myFtab.refresh

end

```

```

'%%%%%%%%%%
'Name: CNCINFShapeConvert
'This script is fired on the select of a item in the combobox in dlgEditData
'It was constructed from two system scripts.
'It uses the info from the combobox and selects the Theme that could be edited
'Nico van Zyl
'2001 March
'%%%%%%%%%%
'=====

```

'Error Checking

'Get the active document, which should be a view. Bail if not a view:

```
=====
theView=av.GetActiveDoc
if (theView.Is(View).Not) then
  MsgBox.Warning("You must select a view before selecting a Theme from
List", "Select View Warning")
  return nil
end
```

'declare the VARIABLES to get the info from the dialog box

```
=====
myDialog = av.FindDialog("dlgEditData")
myTheme = myDialog.FindByName("cboThemeList").GetCurrentValue
myView = av.GetActiveDoc
editThm = myView.GetEditableTheme
myInputFTab = myTheme.getFTab
'Make variables for the tools for editing
myPointControl = myDialog.FindByName ("toolPoint")
myLinePolyControl = myDialog.FindByName ("toolLinePoly")
```

'Test if there is a theme that is being edited

```
=====
if (editThm <> nil) then
  'we need to stop editing this theme before we can go on
  msgbox.warning("You can not start editing a new theme while another is still
edited!" + nl +
  "Press Save/Edit before editing a new Theme", "Warning")
  return nil
```

end

'This part determines where the data will be written

```
=====
newFtab =
myInputFTab.Export("$home\Edit.shp".asFileName,Shape,myInputFTab.GetSelection.Count>0)
```

'This command generates a temp file and then sets it to be deleted after close
myTempFile = "\$HOME".asString.asFileName.MakeTMP("Edit", "shp")

'myTempFile.setScratch(True)

```
newFtab =
myInputFTab.Export(myTempFile,Shape,myInputFTab.GetSelection.Count>0)
```

```
mySrcName=srcName.make(myTempFile.asstring)
```

```
'mySrcName=srcName.make("$home>Edit.shp")
```

```
=====
'make a theme and add it to the view
=====
```

```
myNewTheme = theme.make(mySrcName)
myView.addtheme(myNewTheme)
myNewTheme.SetName (myTheme.asstring)
```

```
=====
'start editing the active theme
=====
```

```
myView.SetEditableTheme(myNewTheme)
```

```
'display the theme to edit
myNewTheme.setactive(true)
myNewTheme.SetVisible (True)
```

```
=====
>Delete the theme from which the shape was made
=====
```

```
myView.deleteTheme(myTheme)
```

```
=====
'This part shows the correct controls to use for the spatial type selected in a theme
=====
```

```
myTag = myTheme.getObjectTag
'msgbox.info(mytag.asstring,"Object tag")
```

```
if (myTag = "Point") then
  myPointControl.SetVisible (True)
  myLinePolyControl.SetVisible (False)
```

```
elseif ((myTag = "Line") or (myTag = "Polygon")) then
```

```
  myPointControl.SetVisible (true)
  myLinePolyControl.SetVisible (True)
  myLinePolyControl.SetEnabled (true)
'elseif (myTag = "Polygon") then
'myPointControl.SetVisible (False)
'myLinePolyControl.SetVisible (True)
```

```
else
  'do nothing
```

```

return nil
end

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Name: CNCINFSplashStart
' This script is not in use at the moment
' It is used to add a banner to the Infrastructure tool
'created: 2001/07/04
'Author" Nico van Zyl
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
myBannerPath =
"P:\Infrastructure\apr\Dialog_backgrounds\CNCINFInfrasplashGIF.GIF"
myfilename = myBannerPath.asFileName

msgbox.Banner(myfilename,2,"Infrastructure Tool")

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Name: CNCINFSqlConnection
'Author: Nico van Zyl
'Created: 2001/02/25
'This script gets the data from the Infrastructure database
'and makes themes for Point, Line and Poly features to add to the view
'in the correct geometry format.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
mysql = self.get(0)
mySpatialVal = self.get(1)

=====
'This part connects to the database and makes a Vtab from it
=====

'Clean up any previous copies of the working tables before connecting to the database
myDoc = av.findDoc("myVtab")
'if (myDoc <> nil) then
'av.GetProject.RemoveDoc(myDoc)
'end

myDSN = "infrastructure"

theSQL=SQLCon.Find(myDSN)
msgbox.report(mysql.asstring,"TEST")
myVtab = vtab.makesql(thesql,mysql)
'Check that at least 1 row was returned
if (myVtab.GetNumRecords < 1) then
Msgbox.info ("Three Queries are made for: Points, Lines and Polygons" + nl + "The
" + mySpatialVal + " dataset contains no data ", "Information on Themes added to
view")
return nil

```

end

```
'-----
' Output Name : Enter the name for the output theme
'-----
myFileName = "$HOME".asString.asfilename.MakeTMP("connect", "shp")
'myFileName = FileDialog.Put (defname,"*.shp", "Enter output file name")
if (myFileName = nil) then return nil end
```

```
'=====
' now split up for the different spatial types: point -> Poly and Line
'=====
```

```
if (mySpatialVal <> "Point") then

    myLatFld = myVtab.Findfield("Latitude")
    myLongFld = myVtab.Findfield("Longitude")
    myInFIDFld = myVtab.Findfield("FeatureID")
```

```
    PointList = list.make
    FIDList = list.make
```

```
'-----
' Prepare the output Line and Poly file
'-----
```

```
'make a new ftab
if (mySpatialVal="Line") then
    shpFtab= ftab.MakeNew(myFileName, PolyLine)
else
    shpFtab= ftab.MakeNew(myFileName, Polygon)
end
```

```
shpFtab.addfields({field.make("ID",#FIELD_LONG ,6,0)})
shpFtab.addfields({field.make("FID",#FIELD_LONG ,6,0)})
```

```
myOutShape = shpFtab.Findfield("Shape")
myOutFID = shpFtab.findfield("FID")
myFirstPoint = nil
myLastFID = myvtab.returnValue(myInFIDFld,0)
'myLastFID = nil
myFirstFID = nil
```

```
'-----
' main loop starts here
'-----
```

```
for each rec in myVtab
    'msgbox.info(rec.asstring,"Debug")
```

```

'Get the lat and long data from the access database and make a point
Lat = myvtab.returnValue(myLatFld,rec)
Long = myvtab.returnValue(myLongFld,rec)
myPoint = point.make(Long,Lat)

'get the feature ID from the access database
myFID = myvtab.returnValue(myInFIDFld,rec)
if (myFirstFID = nil) then myFirstFID = myFID end
if (myFID <> myFirstFID) then myLastFID = myFID end

'Getting the first point in the list to use in Polygon
if (PointList.IsEmpty = true) then myFirstPoint = myPoint end

'check if we must finish a feature off and start a new feature
if (myFID <> myFirstFID) then
  ' Add a final point to close the polygon if needed
  if (mySpatialVal="Polygon") then

    'close off the polygon!
    PointList.add(myFirstPoint)
    'create a new rec on the output FTAB
    polyrec = shpFTab.AddRecord
    'now make a POLYGON shape
    pl = Polygon.Make( {PointList} )
  else
    'presume a line feature is being made!
    'create a new rec on the output FTAB
    polyrec = shpFTab.AddRecord
    'now make a polyline shape
    pl = Polyline.Make( {PointList} )
  end

  'msgbox.info(myRecNum.asstring,"Debug")
  'dump the shape we have built into the output FTAB
  shpFTab.setvalue(myOutShape,PolyRec,pl)
  'set the Feature ID for this shape as it is stored in the access table
  'polyrec = shpFTab.AddRecord

  shpFTab.setvalue(myOutFID,PolyRec,myFirstFID)
  'prepare for the next poly
  PointList = {}
  'add the first point to the list of the next Poly
  pointList.add(myPoint)
  myFirstPoint = myPoint
  myFirstFID = myLastFID
else

'Add the current point to the current point list
pointList.add(myPoint)
end

```

end

```

-----
'this part is used to finish of the last one in a polygon
PolyRecAddOne = 0
' Add a final point to close the polygon if needed
if (mySpatialVal="Polygon") then
  'close off the polygon!
  PointList.add(myFirstPoint)
  'create a new rec on the output FTAB
  polyrec = shpFTab.AddRecord

  'now make a POLYGON shape
  pl = Polygon.Make( {PointList} )
  'msgbox.info(myRecNum.asstring,"Debug")
  'dump the shape we have built into the output FTAB
  shpFtab.setvalue(myOutShape,Polyrec,pl)
  'set the Feature ID for this shape as it is stored in the access table
  myOutFID.SetEditable (True)
  shpFtab.setvalue(myOutFID,Polyrec,myLastFID)
  'prepare for the next poly
  PointList = {}
  shpFTab.Flush

```

else

```

'presume a line feature is being made!
'create a new rec on the output FTAB
polyrec = shpFTab.AddRecord
'now make a polyline shape
pl = Polyline.Make( {PointList} )

shpFtab.setvalue(myOutShape,Polyrec,pl)
shpFtab.setvalue(myOutFID,Polyrec,myLastFID)

```

end

```

-----
' main loop ends here
-----

```

```

-----
' Add the data to a theme and to the view
-----

```

'This part adds the vtab into a permanent table.

```

mytable = "Infrastructure"

```

```

myTable=Table.Make(myVtab)
av.GetProject.AddDoc(mytable)

' Show the table
'mytable.GetWin.Open
myTable.SetName(mySpatialVal)

if (myTable = NIL) then
  return nil
end

'shpFtab.removeRecord(polyRec)
shpFtab.Flush

mySrcName= shpftab.GetSrcName
myTheme2=theme.make(mySrcName)
av.getActiveDoc.addTheme(myTheme2)

  myName = MsgBox.Input ("Type in a name for the new " + mySpatialVal + "
Theme", "Theme Input", mySpatialVal )
  if (myName = nil) then
    return nil
  end

  myTheme2.SetName(myName)
  myTheme2.SetObjectTag(mySpatialVal)

'-----
'This part calls the script that joins the Vtab and Ftab
'-----
  av.run("CNCINFJoinFtabVtab
",{myName,mySpatialVal})

'-----
'This is the point part that adds the points to the view
'-----
else
  myxstring = "longitude"
  myystring = "latitude"
  myXfield = myVtab.Findfield(myXstring)
  myYfield = myVtab.Findfield(myYstring)

  myXYsrc = XYname.make(myVtab,myXfield,myYfield)
  myNewTheme = theme.make(myXYsrc)

```

```

    av.getActiveDoc.AddTheme(myNewTheme)
    myName = MsgBox.Input ("Type in a name for the new Theme", "Theme
Input",mySpatialVal )
    if (myName = nil) then
        return nil
    end

    myNewTheme.SetName(myName)
    myNewTheme.SetObjectTag(mySpatialVal)

end

```

```

=====
' This part is added to clear the values from the dlgProperties dialog so that the
' value do not affect the query if you used it before
=====

```

```

myDialog = av.FindDialog("dlgProperties")
myProperty1 = myDialog.FindByName("cboProperty1")
myProperty2 = myDialog.FindByName("cboProperty2")
myProperty3 = myDialog.FindByName("cboProperty3")
myProperty4 = myDialog.FindByName("cboProperty4")
myProperty5 = myDialog.FindByName("cboProperty5")
myProperty1.SetCurrentValue ("All")
myProperty2.SetCurrentValue ("All")
myProperty3.SetCurrentValue ("All")
myProperty4.SetCurrentValue ("All")
myProperty5.SetCurrentValue ("All")

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Name: CNCINFStartAdSpatDat
'calls "dlgAddSpatialData"
'Nico van Zyl April 2001
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

dlg = av.FindDialog("dlgAddSpatialData")

```

```

dlg.open

```

```

dlg = av.FindDialog("dlgStart")

```

```

dlg.close

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Name: CNCINFStartdlgInfDatSel
'calls "dlgInfraDataSelect"
'Nico van Zyl Maart 2001

```

```
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
```

```
dlg = av.findDialog("dlgSelectData")
```

```
dlg.open
```

```
dlg2 = av.findDialog("dlgStart")
```

```
dlg2.close
```

```
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
```

```
Name: CNCINFstartdlgStart.dlgEditData.Exit
```

```
'This closes the dialog box ('dlgEditData')
```

```
'and opens the 'dlgStart' dialog
```

```
'Nico van Zyl
```

```
'April 2001
```

```
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
```

```
'This script save the data before exit.
```

```
doSave = MsgBox.YesNoCancel("Do you first want to upload the data before exiting  
?", "Exit", true)
```

```
if (doSave = nil) then
```

```
    return nil
```

```
elseif (dosave = false) then
```

```
    dlg = av.findDialog("dlgStart")
```

```
    dlg.open
```

```
else
```

```
    av.run ("CNCINFSaveEdits","variable")
```

```
    dlg = av.findDialog("dlgStart")
```

```
    dlg.open
```

```
end
```

```
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
```

```
Name: CNCINFStartdlgStart.onClose
```

```
'Author: Nico van Zyl
```

```
'Date: 2001/03
```

```
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
```

```
dlg = av.findDialog("dlgStart")
```

```
dlg.open
```

```

'Name: CNCINFstartEditData
'calls "dlgEditData"
'Nico van Zyl Maart 2001

```

```
dlg = av.findDialog("dlgEditData")
```

```
dlg.open
```

```
dlg2 = av.findDialog("dlgStart")
```

```
dlg2.close
```

```

'Name: CNCINFStartEditing
'This script reads the object tags and then make a listbox to show the
'user the Themes that is coming from the ODBC in a combobox
'Nico van Zyl
'2001 March

```

```

' Get the active document, which should be a view. Bail if not a view:
theView=av.GetActiveDoc
if (theView.Is(View).Not) then
  MsgBox.Warning("You must select a view before pressing 'Get Themes'", "Select
View Warning")
  return nil
end

```

```

theView = av.GetActiveDoc
aList = List.Make
emptylist = true
for each t in theView.GetThemes
  myTag=t.getObjectTag
  'if (myTag = "ODBC Theme") then
  if (myTag <> nil) then
    aList.Add(t)
    emptylist = false
  end
end
if (emptylist=true) then
  return nil
end

```

```
myDialog = av.FindDialog("dlgEditData")
```

```
myCombo = myDialog.findByName("cboThemeList")
```

```
myCombo.definefromList(alist)
```

```
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
' Script to UNinstall my extension
' Tim Sutton 1998
' Script Name: CNCINFUnInstallScript
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
'The self object for this script is the extension
```

```
'Check if there is an active project
```

```
if (av.GetProject = nil) then
  return nil
end
```

```
'No need to uninstall the extension if the project is closing
```

```
if (av.GetProject.IsClosing) then
  return nil
end
```

```
'Remove Custon buttons
```

```
'DeleteIcons = av.run("CNC.ButtonsDelete", "")
```

```
'PROJECT MENU : Remove the CNCInfrastructure menu
```

```
'myMenuBar = av.GetProject.FindGui("Project").GetMenuBar
'myMenuBar.Remove(self.Get(0))
```

```
'VIEW MENU : Remove the CNC Utilities menu
```

```
myMenuBar = av.GetProject.FindGui("View").GetMenuBar
myMenuBar.Remove(self.Get(0))
```

```
'-----
'The following script segment was kindly provided by
'Zook, Cody/CVO [czoek@CH2M.com]
'
```

```
'It removes object dependancies from all dialogs
'
```

```
for each aDoc in av.GetProject.GetDocs
```

```
  if (aDoc.Is(DialogEditor)) then
```

```
    av.ShowMsg("Removing linkages from dialog: "+aDoc.GetName.AsString)
```



```

myProject = av.GetProject

' First do a check and make certain that all scripts are properly compiled
'-----
countScripts = 0
countNotComp = 0
badList = list.make

for each d in av.GetProject.GetDocs
  if (d.Is(SEd).NOT) then
    continue
  end
  countScripts = countScripts + 1

  if (d.IsCompiled.NOT) then
    badlist.add(d.getName)
    countNotComp = countNotComp + 1
    continue
  end
end

If (countNotComp > 0) then
  MsgBox.listasstring (badlist, countScripts.asstring + " Script(s) checked, " +
countNotComp.asstring + " scripts uncompiled.", "CNC Extension Builder Error")
  return Nil
end

'-----
'Get install and uninstall scripts

myInstallScript = myProject.FindDoc("cncINFInstallScript").getScript
myUnInstallScript = myProject.FindDoc("cncINFUnInstallScript").getScript

'Make sure the above scripts are compiled

if ((myInstallScript = nil) or (myUnInstallScript = nil)) then MsgBox.info("Install or
UnInstall scripts not compiled", "")
  return nil
end

' Make the extension file

myExtension =
Extension.make("$AVEXT/CNCINF_Infrastructure.avx".asFileName, "CNC
Infrastructure Extension", myInstallScript, myUninstallScript,
{"$AVBIN/avdlog.dll".asFilename})

'Set the CanUnload script
myCanUnloadScript = myProject.FindScript("CNCINF.CanUnload")
myExtension.SetCanUnloadScript(myCanUnloadScript)

```

```
'Set the unload script to clean up dialog designer
myUnloadScript = myProject.FindScript("CNCINFExtensionUnload")
myExtension.SetUnloadScript(myUnloadScript)
```

'PROJECT MENU : Get the new menu item (Infrastructure) and add it to the extension

```
'myViewMenuBar = myProject.FindGui("Project").getMenuBar
'myMenuItem = myViewMenuBar.FindByLabel("Infrastructure")
'myExtension.add(myMenuItem)
```

'VIEW MENU : Get the new menu item (CNC Utilities) and add it to the extension

```
myViewMenuBar = myProject.FindGui("View").getMenuBar
myMenuItem = myViewMenuBar.FindByLabel("Infrastructure")
myExtension.add(myMenuItem)
```

'Table MENU : Get the new menu item (CNC Utilities) and add it to the extension

```
'myViewMenuBar = myProject.FindGui("Table").getMenuBar
'myMenuItem = myViewMenuBar.FindByLabel("Field")
'myExtension.add(myMenuItem)
```

' Add all scripts prefixed with CNCINF to the extension

```
myScriptList = list.make
for each d in myProject.getDocs
  if (d.is(SEd) and (d.GetName.Left("CNCIN".count + 1) = "CNCINF")) then
    myExtension.Add(d.getScript)
    myScriptList.add(d.GetName)
  end
end
```

```
'-----
'The following script segment was kindly provided by
'Zook, Cody/CVO [czook@CH2M.com]
```

'It removes object dependancies from all dialogs

```
'
for each aDoc in av.GetProject.GetDocs
  if (aDoc.Is(DialogEditor)) then
    av.ShowMsg("Removing linkages from dialog: "+aDoc.GetName.AsString)
    System.RefreshWindows
    dlg = aDoc.GetDialog

    dlg.SetServer(nil)
```

```

for each lbx in dlg.FindByClass(ListBox)
  'Remove linkage to any data sources used by the project
  lbx.Empty
end
for each cbx in dlg.FindByClass(ComboBox)
  'Remove linkage to any data sources used by the project
  cbx.Empty
end
end
end
'remove database dependencies
alist = sqlcon.getconnections
for each i in alist.clone
  abool = i.islogin
  if (abool) then
    i.logout
  end
end
end
'-----

```

'Modify the lines below if you need to use any dialogs

```

'Add the Metabrowser dialog
myDialog = av.finddialog("dlgEditdata")
myExtension.add(myDialog)
myDialog = av.finddialog("dlgProperties")
myExtension.add(myDialog)
myDialog = av.finddialog("dlgSelectData")
myExtension.add(myDialog)
myDialog = av.finddialog("dlgStart")
myExtension.add(myDialog)
myDialog = av.finddialog("dlgAddSpatialData")
myExtension.add(myDialog)

```

```

myLoadScript = myProject.FindScript("CNCINFload")
myExtension.setLoadScript(myLoadScript)

```

'Add a description to appear in the extensions dialog

```

myExtension.setAbout("Cape Nature Conservation Infrastructure tool: build 19. Nico
van Zyl")

```

' Now commit the changes

```

myExtension.commit

```

'clean up any unnecessary object dependancies such as for the avdialogue editor

' code below taken from
<http://www.primenet.com/~piersen/arcview/avtips/avxbomb.htm>
'error in last line to be sorted out before this will work

```
'RFileName = FileName.Make("$USEREXT/cnc_utilities.avx")
'WFileName = FileName.Make("$USEREXT/cnc_utilities.tmp")
'RFile = LineFile.Make(RFileName,#FILE_PERM_READ)
'WFile = LineFile.Make(WFileName,#File_PERM_WRITE)
'WFile.SetScratch(TRUE)
'  while (RFile.IsAtEnd.Not)
'    buf = RFile.ReadElt
'    if (buf.Contains("ObjectTag:").not) then
'      WFile.WriteElt(buf)
'    end
'end
'RFile.Close
'WFile.Flush
'File.Copy(WFilename, RFileName)
'WFile.Close
'
```

```
msgbox.listasstring(myScriptList,"The following scripts were added, totalling " +
myScriptList.count.asstring + " to the AVX", "CNC Infrastructure")
```

APPENDIX 2

MS Access Modules

Form_frmImage

Option Compare Database

Option Explicit

Private Sub Form_Current()

On Error GoTo Current_error

If IsNull(Me.txtPhotoPath) = False And Me.txtPhotoPath <> "" Then

 Me.ImageFeatures.Picture = Me.txtPhotoPath

Else

 Me.ImageFeatures.Picture = ""

End If

Exit Sub

Current_error:

 'load a default image when an error occurs

 Me.ImageFeatures.Picture = "r:\Photos\image.jpg"

End Sub

Private Sub cmdClose_Click()

On Error GoTo Err_cmdClose_Click

 DoCmd.Close

Exit_cmdClose_Click:

 Exit Sub

Err_cmdClose_Click:

 MsgBox Err.Description

 Resume Exit_cmdClose_Click

End Sub

Form_FrmInspection

Option Compare Database

Option Explicit

Private Sub cmdPicture_Click()

On Error GoTo Err_cmdPicture_Click

 Dim myImage As String

 Dim stDocName As String

 Dim stLinkCriteria As String

 stDocName = "frmImage"

```
stLinkCriteria = "[photoPath]=" & "" & Me![photoPath] & ""
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdPicture_Click:
```

```
Exit Sub
```

```
Err_cmdPicture_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdPicture_Click
```

```
'myImage = Forms!frmImage.txtPhotoPath
```

```
'Forms!frmImage.imageFeature.Picture = myImage
```

```
'Me.Refresh
```

```
End Sub
```

```
Private Sub cmdReportCond_Click()
```

```
On Error GoTo Err_cmdReportCond_Click
```

```
Dim stDocName As String
```

```
stDocName = "rptCondition"
```

```
DoCmd.OpenReport stDocName, acPreview
```

```
Exit_cmdReportCond_Click:
```

```
Exit Sub
```

```
Err_cmdReportCond_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdReportCond_Click
```

```
End Sub
```

```
Private Sub cmdReportInspect_Click()
```

```
On Error GoTo Err_cmdReportInspect_Click
```

```
Dim stDocName As String
```

```
stDocName = "rptInspectedByPie"
```

```
DoCmd.OpenReport stDocName, acPreview
```

```
Exit_cmdReportInspect_Click:
```

```
Exit Sub
```

```
Err_cmdReportInspect_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdReportInspect_Click
```

```
End Sub
```

```
Form_frmMain
```

Option Compare Database
Option Explicit

Private Sub general()

' This section evaluates the property names of the tblType table.

Dim myDB As Database
Dim myRST As Recordset
Dim mySql As String

Set myDB = CurrentDb()

'check that the combo actually has info in first

If IsNull(Me.cboNames) Then

Exit Sub

End If

mySql = "select * from qryDebug " _
& "WHERE (((tblType.FeatureTypeID)=" & Me.FeatureTypeID & ");"

'Debug.Print mySql

' This part sets the database that you can only add a new record

Set myRST = myDB.OpenRecordset(mySql, dbOpenDynaset, dbReadOnly)

With myRST

'property 1 viable check

If IsNull(!Prop1) Then

Me.lblproperty1.Visible = False

Me.txtProperty1.Visible = False

Else

Me.txtProperty1.Visible = True

Me.lblproperty1.Visible = True

Me.lblproperty1.Caption = !Prop1

End If

'property 2 viable check

If IsNull(!Prop2) Then

Me.lblProperty2.Visible = False

Me.txtProperty2.Visible = False

Else

Me.txtProperty2.Visible = True

Me.lblProperty2.Visible = True

Me.lblProperty2.Caption = !Prop2

End If

'property 3 viable check

If IsNull(!Prop3) Then

Me.lblproperty3.Visible = False

Me.txtProperty3.Visible = False

Else

Me.txtProperty3.Visible = True

Me.lblproperty3.Visible = True

Me.lblproperty3.Caption = !Prop3

End If

```

'property 4 viable check
  If IsNull(!Prop4) Then
    Me.lblproperty4.Visible = False
    Me.txtProperty4.Visible = False
  Else
    Me.txtProperty4.Visible = True
    Me.lblproperty4.Visible = True
    Me.lblproperty4.Caption = !Prop4
  End If
'property 5 viable check
  If IsNull(!Prop5) Then
    Me.lblProperty5.Visible = False
    Me.txtProperty5.Visible = False
  Else
    Me.txtProperty5.Visible = True
    Me.lblProperty5.Visible = True
    Me.lblProperty5.Caption = !Prop5
  End If

.Close
End With
End Sub

Private Sub cboNames_AfterUpdate()
'moved...
  Dim spatialTypeID As String
  Dim mySpatialTypeID As String
  mySpatialTypeID = Me.txtFeatureTypeID
  spatialTypeID = DLookup("SpatialTypeID", "tblType", "FeatureTypeID=" &
mySpatialTypeID)

  If spatialTypeID = 1 Then 'is a point so no need to do anything
Form_frmPointsData.RecordSource = "tblPointsdata"
Form!frmPointsData.Requery
  ElseIf spatialTypeID = 2 Then
Form_frmPointsData.RecordSource = "qryLineSpatialData"
Form!frmPointsData.Requery
'Form_frmPointsData.AllowEdits
  ElseIf spatialTypeID = 3 Then
Form_frmPointsData.RecordSource = "qryPolySpatialData"
Form!frmPointsData.Requery
  End If

End Sub

Private Sub cboNames_Change()
Dim editOk As Boolean

```

```
'If editOk = True Then
```

```
    If (Me.txtNames.Enabled = False) Then
```

```
        Me.txtProperty1.Enabled = True
        Me.txtProperty2.Enabled = True
        Me.txtProperty3.Enabled = True
        Me.txtProperty4.Enabled = True
        Me.txtProperty5.Enabled = True
        Me.txtNames.Enabled = True
        Me.txtNotes.Enabled = True
        Me.cboReserveName.Enabled = True
        Call general
```

```
    End If
```

```
'Else
```

```
    '    Call general
```

```
'End If
```

```
'Else
```

```
    'Me.ActiveControl.Value = Me.ActiveControl.OldValue
```

```
'End If
```

```
End Sub
```

```
Private Sub cboNames_GotFocus()
```

```
Me.Refresh
```

```
End Sub
```

```
Private Sub Form_AfterUpdate()
```

```
' This section evaluates the property names of the tblType table.
```

```
    Dim myDB As Database
```

```
    Dim myRST As Recordset
```

```
    Dim mySql As String
```

```
    Set myDB = CurrentDb()
```

```
'check that the combo actually has info in first
```

```
If IsNull(Me.cboNames) Then
```

```
    Exit Sub
```

```
End If
```

```
mySql = "select * from qryValueListCombo " & "WHERE (((tblPropertyList.PropertyNames)=' & Me.lblproperty1.Caption & '));"
```

```
Debug.Print mySql
```

```
'Debug.Print mySql
```

```
' This part sets the database that you can only add a new record
```

```
Set myRST = myDB.OpenRecordset(mySql, dbOpenDynaset, dbReadOnly)
```

```
With myRST
```

```

'property 1 viable check
  If IsNull(!Value) Then
    'Me.lblproperty1.Visible = False
    'Me.txtProperty1.Visible = False
  Else
    Me.txtProperty1.Visible = True
    'Me.lblproperty1.Visible = True
    'Me.txtProperty1.Caption = !Value
    'Forms!Employees!cmbNames.RowSource = "EmployeeList"
    Me.txtProperty1.RowSource = !PropertyNames
  End If

  .Close
End With

```

```
End Sub
```

```

' this stops cbonames from giving a error when adding a new field
Private Sub Form_Current()
Dim mySpatialTypeID As String
'Dim txtProperty1 As ComboBox
If IsNull(Me.cboNames) Then
'do nothing
'Me.txtProperty1.Enabled = False

Else
'General selects and show the labels that mathes the cboNames
Call general
' This part disables the txtboxes so that you can not edit the data
Me.txtProperty1.Enabled = False
Me.txtProperty2.Enabled = False
Me.txtProperty3.Enabled = False
Me.txtProperty4.Enabled = False
Me.txtProperty5.Enabled = False
Me.txtNames.Enabled = False
Me.txtNotes.Enabled = False
Me.cboReserveName.Enabled = False

```

```

Dim spatialTypeID As String
mySpatialTypeID = Me.txtFeatureTypeID
spatialTypeID = DLookup("SpatialTypeID", "tblType", "FeatureTypeID=" &
mySpatialTypeID)

```

```

If spatialTypeID = 1 Then 'is a point so no need to do anything
Form_frmPointsData.RecordSource = "tblPointsdata"
Form!frmPointsData.Requery
  ElseIf spatialTypeID = 2 Then
  Form_frmPointsData.RecordSource = "qryLineSpatialData"
  Form!frmPointsData.Requery
  'Form_frmPointsData.AllowEdits
  ElseIf spatialTypeID = 3 Then
  Form_frmPointsData.RecordSource = "qryPolySpatialData"
  Form!frmPointsData.Requery
End If
End If

```

```

Dim myDB As Database
Dim myRST As Recordset
Dim mySql As String
Dim mySql2 As String
Dim mySql3 As String
Dim mySql4 As String
Dim mySql5 As String

```

```

Set myDB = CurrentDb()
'check that the cboNames combo actually has info in first
If IsNull(Me.cboNames) Then
  Exit Sub
End If

```

```

'Populates the comboboxes to show the values allowed for each property
mySql = "select * from qryValueListCombo " _
& "WHERE (((tblPropertyList.PropertyNames)=" & Me.lblproperty1.Caption &
""));"
Debug.Print mySql
Me.txtProperty1.RowSource = mySql
Me.txtProperty1.Requery
mySql2 = "select * from qryValueListCombo " _
& "WHERE (((tblPropertyList.PropertyNames)=" & Me.lblProperty2.Caption &
""));"
Debug.Print mySql
Me.txtProperty2.RowSource = mySql2
Me.txtProperty2.Requery
mySql3 = "select * from qryValueListCombo " _
& "WHERE (((tblPropertyList.PropertyNames)=" & Me.lblproperty3.Caption &
""));"
Debug.Print mySql
Me.txtProperty3.RowSource = mySql3
Me.txtProperty1.Requery
mySql4 = "select * from qryValueListCombo " _
& "WHERE (((tblPropertyList.PropertyNames)=" & Me.lblproperty4.Caption &
""));"

```

```

Debug.Print mySql
Me.txtProperty4.RowSource = mySql4
    Me.txtProperty1.Requery
mySql5 = "select * from qryValueListCombo " _
& "WHERE (((tblPropertyList.PropertyNames)=" & Me.lblProperty5.Caption &
""));"
Debug.Print mySql
Me.txtProperty5.RowSource = mySql5
    Me.txtProperty1.Requery

```

```
End Sub
```

```
Private Sub cmdNewFeature_Click()
On Error GoTo Err_cmdNewFeature_Click

```

```
    DoCmd.GoToRecord , , acNewRec

```

```
Exit_cmdNewFeature_Click:
Exit Sub

```

```
Err_cmdNewFeature_Click:
MsgBox Err.Description
Resume Exit_cmdNewFeature_Click

```

```
End Sub
```

```
Private Sub Form_Current_()
Call general

```

```
End Sub
```

```
Private Sub Form_Load()

```

```
End Sub
```

```
Private Sub txtProperty1_BeforeUpdate(Cancel As Integer)
Call general

```

```
End Sub
```

```
Private Sub cmdAddRecord_Click()
On Error GoTo Err_cmdAddRecord_Click

```

```
    DoCmd.GoToRecord , , acNewRec

```

```
Exit_cmdAddRecord_Click:
```

```
Exit Sub
```

```
Err_cmdAddRecord_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdAddRecord_Click
```

```
End Sub
```

```
Private Sub cmdAddFeature_Click()
```

```
On Error GoTo Err_cmdAddFeature_Click
```

```
Dim stDocName As String
```

```
Dim stLinkCriteria As String
```

```
stDocName = "frmPropertyName"
```

```
stLinkCriteria = "[Type]=" & "" & Me![txtNames] & ""
```

```
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdAddFeature_Click:
```

```
Exit Sub
```

```
Err_cmdAddFeature_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdAddFeature_Click
```

```
End Sub
```

```
Private Sub cmdNewFeature2_Click()
```

```
On Error GoTo Err_cmdNewFeature2_Click
```

```
Dim stDocName As String
```

```
Dim stLinkCriteria As String
```

```
stDocName = "frmPropertyName"
```

```
stLinkCriteria = "[Type]=" & "" & Me![cboNames] & ""
```

```
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdNewFeature2_Click:
```

```
Exit Sub
```

```
Err_cmdNewFeature2_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdNewFeature2_Click
```

```
End Sub
```

```
Private Sub cmdQuit_Click()
```

```
On Error GoTo Err_cmdQuit_Click
```

```
DoCmd.Quit
```

```
Exit_cmdQuit_Click:
    Exit Sub
```

```
Err_cmdQuit_Click:
    MsgBox Err.Description
    Resume Exit_cmdQuit_Click
```

```
End Sub
```

```
Private Sub cmdOpenPropertyNames_Click()
    On Error GoTo Err_cmdOpenPropertyNames_Click
```

```
    Dim stDocName As String
    Dim stLinkCriteria As String
    If IsNull(Me.cboNames) = True Then
        On Error GoTo cmdOpenPropertyNames_Click

        Dim stDocName As String
        Dim stLinkCriteria As String

        stDocName = "frmPropertyName"
        stLinkCriteria = "[FeatureTypeID]=" & [null]
        DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdOpenPropertyNames_Click:
    Exit Sub
```

```
Err_cmdOpenPropertyNames_Click:
    MsgBox Err.Description
    Resume Exit_cmdOpenPropertyNames_Click
```

```
End Sub
```

```
Else
```

```
    stDocName = "frmPropertyName"
    stLinkCriteria = "[PropertyID]=" & Me![cboNames]
    DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdOpenPropertyNames_Click:
    Exit Sub
```

```
Err_cmdOpenPropertyNames_Click:
    MsgBox Err.Description
    Resume Exit_cmdOpenPropertyNames_Click
```

```
End If
```

```
End Sub
```

```
Private Sub cmdExit_Click()
```

On Error GoTo Err_cmdExit_Click

DoCmd.Close

Exit_cmdExit_Click:
Exit Sub

Err_cmdExit_Click:
MsgBox Err.Description
Resume Exit_cmdExit_Click

End Sub

'%%%

Private Sub cmdNewFeatureSel_Click()

On Error GoTo Err_cmdNewFeatureSel_Click

'%%%

Dim stDocName As String

Dim stLinkCriteria As String

If IsNull(Me.cboNames) = True Then

On Error GoTo Err_cmdNewFeatureSel_Click

stDocName = "frmPropertyName"

DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_cmdNewFeatureSel_Click:
Exit Sub

Err_cmdNewFeatureSel_Click:
MsgBox Err.Description
Resume Exit_cmdNewFeatureSel_Click

Else

stDocName = "frmPropertyName"

stLinkCriteria = "[FeatureTypeID]=" & Me![cboNames]

DoCmd.OpenForm stDocName, , , stLinkCriteria

End If

End Sub

Private Sub cmdReport_Click()
On Error GoTo Err_cmdReport_Click

Dim stDocName As String

```
stDocName = "rptFeature_Property_Value_SortType"  
DoCmd.OpenReport stDocName, acPreview
```

```
Exit_cmdReport_Click:  
Exit Sub
```

```
Err_cmdReport_Click:  
MsgBox Err.Description  
Resume Exit_cmdReport_Click
```

```
End Sub
```

Form_frmNewProperty

```
Option Compare Database  
Option Explicit
```

```
Private Sub cmdExit_Click()  
On Error GoTo Err_cmdExit_Click
```

```
DoCmd.Close
```

```
Exit_cmdExit_Click:  
Exit Sub
```

```
Err_cmdExit_Click:  
MsgBox Err.Description  
Resume Exit_cmdExit_Click
```

```
End Sub
```

```
Private Sub cmdOpenPropertyName_Click()  
On Error GoTo Err_cmdOpenPropertyName_Click
```

```
Dim stDocName As String  
Dim stLinkCriteria As String
```

```
stDocName = "frmPropertyName"  
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdOpenPropertyName_Click:  
Exit Sub
```

```
Err_cmdOpenPropertyName_Click:  
MsgBox Err.Description  
Resume Exit_cmdOpenPropertyName_Click
```

```
End Sub
```

Form_frmPointsData

Option Compare Database

Option Explicit

Private Sub Form_AfterInsert()

Dim db As Database

Dim rsline As Recordset

Dim rsPoly As Recordset

Dim rsLineID As Recordset

Dim rsPolyId As Recordset

Dim rsFeatureID As Recordset

Dim rsPoint As Recordset

Dim spatialTypeID As String

Dim PolyID As String

Dim LineID As String

Dim mySpatialTypeID As String

Dim myFeatureID As String

Dim myLineID As String

Dim myLineIDSql As String

Dim myPolyIDSql As String

Dim mySqlUpdate As String

Dim myPointID As String

Dim qdf As QueryDef

Set db = CurrentDb

mySpatialTypeID = Forms!frmMain.txtFeatureTypeID

myFeatureID = Forms!frmMain.txtFeatureId

myPointID = Me.PointID

spatialTypeID = DLookup("SpatialTypeID", "tblType", "FeatureTypeID=" &
mySpatialTypeID)

Set db = CurrentDb

If spatialTypeID = 1 Then 'is a point so no need to do anything

ElseIf spatialTypeID = 2 Then 'is a line so add line rec

Set rsline = db.OpenRecordset("tblLinesData", dbOpenDynaset)

'this part adds the featureID from the tblFeatures to the table

```

With rsline
.MoveLast
.FindFirst FeatureID = myFeatureID
If .NoMatch Then

```

```

rsline.AddNew
rsline!FeatureID = myFeatureID
rsline.Update
End If
End With

```

```

LineID = DLookup("LineID", "tblLinesData", "FeatureID=" & myFeatureID)
myLineIDSql = "SELECT tblPointsData.LineID FROM tblPointsData WHERE
tblPointsData.PointID=" & myPointID & ";"

```

```

Set rsLineID = db.OpenRecordset(myLineIDSql, dbOpenDynaset)

```

```

rsLineID.Edit
rsLineID!LineID = LineID
rsLineID.Update
rsLineID.Close

```

```

ElseIf spatialTypeID = 3 Then

```

```

Set rsPoly = db.OpenRecordset("tblPoly", dbOpenDynaset)

```

```

'this part adds the featureID from the tblFeatures to the table

```

```

rsPoly.AddNew
rsPoly!FeatureID = myFeatureID
rsPoly.Update

```

```

PolyID = DLookup("PolyID", "tblPoly", "FeatureID=" & myFeatureID)
myPolyIDSql = "SELECT tblPointsdata.PolyID FROM tblPointsData WHERE
tblPointsData.PointID=" & myPointID & ";"

```

```

Set rsPolyId = db.OpenRecordset(myPolyIDSql, dbOpenDynaset)
rsPolyId.Edit
rsPolyId!PolyID = PolyID
rsPolyId.Update
rsPolyId.Close

```

```

End If
End Sub

```

Form_frmPropertyName

```

Option Compare Database
Option Explicit

```

```

Private Sub cmbtblType_Change()

```

```

End Sub

```

```
Private Sub Command16_Click()  
On Error GoTo Err_Command16_Click
```

```
    DoCmd.GoToRecord , , acNewRec
```

```
Exit_Command16_Click:  
    Exit Sub
```

```
Err_Command16_Click:  
    MsgBox Err.Description  
    Resume Exit_Command16_Click
```

```
End Sub
```

```
Private Sub cmbFind_Click()  
On Error GoTo Err_cmbFind_Click
```

```
    Screen.PreviousControl.SetFocus  
    DoCmd.DoMenuItem acFormBar, acEditMenu, 10, , acMenuVer70
```

```
Exit_cmbFind_Click:  
    Exit Sub
```

```
Err_cmbFind_Click:  
    MsgBox Err.Description  
    Resume Exit_cmbFind_Click
```

```
End Sub
```

```
Private Sub Command20_Click()  
On Error GoTo Err_Command20_Click
```

```
    DoCmd.Quit
```

```
Exit_Command20_Click:  
    Exit Sub
```

```
Err_Command20_Click:  
    MsgBox Err.Description  
    Resume Exit_Command20_Click
```

```
End Sub
```

```
Private Sub cmbCloseForm_Click()  
On Error GoTo Err_cmbCloseForm_Click
```

```
DoCmd.Close
```

```
Exit_cmbCloseForm_Click:  
Exit Sub
```

```
Err_cmbCloseForm_Click:  
MsgBox Err.Description  
Resume Exit_cmbCloseForm_Click
```

```
End Sub
```

```
Private Sub Command56_Click()  
On Error GoTo Err_Command56_Click
```

```
Dim stDocName As String  
Dim stLinkCriteria As String
```

```
stDocName = "frmNewProperty"
```

```
stLinkCriteria = "[PropertyNames]=" & "" & Me![cbotblType] & ""  
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_Command56_Click:  
Exit Sub
```

```
Err_Command56_Click:  
MsgBox Err.Description  
Resume Exit_Command56_Click
```

```
End Sub
```

```
Private Sub Command58_Click()  
On Error GoTo Err_Command58_Click
```

```
Dim stDocName As String  
Dim stLinkCriteria As String
```

```
stDocName = "frmNewProperty"
```

```
stLinkCriteria = "[PropertyNames]=" & "" & Me![cboPropertyName1] & ""  
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_Command58_Click:  
Exit Sub
```

```
Err_Command58_Click:  
MsgBox Err.Description  
Resume Exit_Command58_Click
```

```
End Sub
```

```
Private Sub Command59_Click()
On Error GoTo Err_Command59_Click
```

```
    Dim stDocName As String
    Dim stLinkCriteria As String
```

```
    stDocName = "frmNewProperty"
```

```
    stLinkCriteria = "[PropertyID]=" & Me![cboPropertyName1]
    DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_Command59_Click:
    Exit Sub
```

```
Err_Command59_Click:
    MsgBox Err.Description
    Resume Exit_Command59_Click
```

```
End Sub
```

```
Private Sub cboPropertyName1_GotFocus()
Me.Refresh
End Sub
```

```
Private Sub cboPropertyName2_GotFocus()
Me.Refresh
End Sub
```

```
Private Sub cboPropertyName3_GotFocus()
Me.Refresh
End Sub
```

```
Private Sub cboPropertyName4_GotFocus()
Me.Refresh
End Sub
```

```
Private Sub cboPropertyName5_GotFocus()
Me.Refresh
End Sub
```

```
Private Sub cmdNewProperty1_Click()
On Error GoTo Err_cmdNewProperty_Click
If IsNull(Me.cboPropertyName1) = True Then
```

```
    MsgBox " There must be a value in the textbox on the left. Use the ' NEW
PROPERTY ' button to add a Property", vbExclamation, "Can not edit the value of
nothing"
```

```
    Me.cboPropertyName1.SetFocus
    'Cancel = True
```

```
End
End If
```

```
    Dim stDocName As String
```

```
Dim stLinkCriteria As String
```

```
stDocName = "frmNewProperty"
```

```
stLinkCriteria = "[PropertyID]=" & Me![cboPropertyName1]  
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdNewProperty_Click:
```

```
Exit Sub
```

```
Err_cmdNewProperty_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdNewProperty_Click
```

```
End Sub
```

```
Private Sub cmdNewProperty2_Click()
```

```
On Error GoTo Err_cmdNewProperty_Click
```

```
If IsNull(Me.cboPropertyName2) = True Then
```

```
MsgBox " There must be a value in the textbox on the left. Use the ' NEW  
PROPERTY ' button to add a Property", vbExclamation, "Can not edit the value of  
nothing"
```

```
Me.cboPropertyName2.SetFocus
```

```
End
```

```
End If
```

```
Dim stDocName As String
```

```
Dim stLinkCriteria As String
```

```
stDocName = "frmNewProperty"
```

```
stLinkCriteria = "[PropertyID]=" & Me![cboPropertyName2]  
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdNewProperty_Click:
```

```
Exit Sub
```

```
Err_cmdNewProperty_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdNewProperty_Click
```

```
End Sub
```

```
Private Sub cmdNewProperty3_Click()
```

```
On Error GoTo Err_cmdNewProperty_Click
```

```
If IsNull(Me.cboPropertyName3) = True Then
```

```
MsgBox " There must be a value in the textbox on the left. Use the ' NEW  
PROPERTY ' button to add a Property", vbExclamation, "Can not edit the value of  
nothing"
```

```
Me.cboPropertyName3.SetFocus
```

```

End
End If
    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = "frmNewProperty"

    stLinkCriteria = "[PropertyID]=" & Me![cboPropertyName3]
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_cmdNewProperty_Click:
    Exit Sub

Err_cmdNewProperty_Click:
    MsgBox Err.Description
    Resume Exit_cmdNewProperty_Click

End Sub
Private Sub cmdNewProperty4_Click()
On Error GoTo Err_cmdNewProperty_Click

If IsNull(Me.cboPropertyName4) = True Then

    MsgBox " There must be a value in the textbox on the left. Use the ' NEW
PROPERTY ' button to add a Property", vbExclamation, "Can not edit the value of
nothing"
    Me.cboPropertyName4.SetFocus
End
End If
    Dim stDocName As String
    Dim stLinkCriteria As String

    stDocName = "frmNewProperty"

    stLinkCriteria = "[PropertyID]=" & Me![cboPropertyName4]
    DoCmd.OpenForm stDocName, , , stLinkCriteria

Exit_cmdNewProperty_Click:
    Exit Sub

Err_cmdNewProperty_Click:
    MsgBox Err.Description
    Resume Exit_cmdNewProperty_Click

End Sub
Private Sub cmdNewProperty5_Click()
On Error GoTo Err_cmdNewProperty_Click

If IsNull(Me.cboPropertyName5) = True Then

```

```
MsgBox " There must be a value in the textbox on the left. Use the ' NEW
PROPERTY ' button to add a Property", vbExclamation, "Can not edit the value of
nothing"
```

```
Me.cboPropertyName5.SetFocus
End
End If
```

```
Dim stDocName As String
Dim stLinkCriteria As String
```

```
stDocName = "frmNewProperty"
```

```
stLinkCriteria = "[PropertyID]=" & Me![cboPropertyName5]
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdNewProperty_Click:
Exit Sub
```

```
Err_cmdNewProperty_Click:
MsgBox Err.Description
Resume Exit_cmdNewProperty_Click
```

```
End Sub
Private Sub cmdNewPropertyDo_Click()
On Error GoTo Err_cmdNewPropertyDo_Click
```

```
Dim stDocName As String
Dim stLinkCriteria As String
```

```
stDocName = "frmNewProperty"
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdNewPropertyDo_Click:
Exit Sub
```

```
Err_cmdNewPropertyDo_Click:
MsgBox Err.Description
Resume Exit_cmdNewPropertyDo_Click
```

```
End Sub
Private Sub cmdNewFeature_Click()
On Error GoTo Err_cmdNewFeature_Click
```

```
DoCmd.GoToRecord , , acNewRec
```

```
Exit_cmdNewFeature_Click:
Exit Sub
```

```
Err_cmdNewFeature_Click:
```

```
MsgBox Err.Description
Resume Exit_cmdNewFeature_Click
```

```
End Sub
```

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
```

```
    If IsNull(Me.cboSpatialType) = True Then
```

```
        MsgBox "A value in Spatial Type is required to add a record!", vbExclamation,
"Spatial Type Required!"
        Me.cboSpatialType.SetFocus
        Cancel = True
```

```
    End If
```

```
End Sub
```

```
Private Sub cmdExit_Click()
```

```
On Error GoTo Err_cmdExit_Click
```

```
    DoCmd.Close
```

```
Exit_cmdExit_Click:
```

```
Exit Sub
```

```
Err_cmdExit_Click:
```

```
    MsgBox Err.Description
    Resume Exit_cmdExit_Click
```

```
End Sub
```

Form_frmStart

```
Option Compare Database
```

```
Option Explicit
```

```
Private Sub cmdEnter_Click()
```

```
On Error GoTo Err_cmdEnter_Click
```

```
    Dim stDocName As String
    Dim stLinkCriteria As String
```

```
    stDocName = "frmMain"
    DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdEnter_Click:
```

```
Exit Sub
```

```
Err_cmdEnter_Click:
```

```
    MsgBox Err.Description
```

```
Resume Exit_cmdEnter_Click
```

```
End Sub
```

```
Private Sub cmdNewCreteria_Click()
```

```
On Error GoTo Err_cmdNewCreteria_Click
```

```
Dim stDocName As String
```

```
Dim stLinkCriteria As String
```

```
stDocName = "frmPropertyName"
```

```
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdNewCreteria_Click:
```

```
Exit Sub
```

```
Err_cmdNewCreteria_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdNewCreteria_Click
```

```
End Sub
```

```
Private Sub cmdNewProperties_Click()
```

```
On Error GoTo Err_cmdNewProperties_Click
```

```
Dim stDocName As String
```

```
Dim stLinkCriteria As String
```

```
stDocName = "frmNewProperty"
```

```
DoCmd.OpenForm stDocName, , , stLinkCriteria
```

```
Exit_cmdNewProperties_Click:
```

```
Exit Sub
```

```
Err_cmdNewProperties_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdNewProperties_Click
```

```
End Sub
```

APPENDIX 3

SQL queries

qryLineSpatialData (query to select the correct fields for Line Spatial Data)

```
SELECT tblPointsData.PointID, tblPointsData.Latitude, tblPointsData.Longitude,  
tblPointsData.ZValue, tblPointsData.LineID, tblLinesData.featureID  
FROM (tblFeatures INNER JOIN tblLinesData ON tblFeatures.FeatureID =  
tblLinesData.featureID) INNER JOIN tblPointsData ON tblLinesData.lineID =  
tblPointsData.LineID;
```

qryPolySpatialData (query to select the correct fields for Polygon Spatial Data)

```
SELECT tblPointsData.PointID, tblPointsData.Latitude, tblPointsData.Longitude,  
tblPointsData.ZValue, tblPoly.FeatureID, tblPointsData.PolyID  
FROM (tblFeatures INNER JOIN tblPoly ON tblFeatures.FeatureID =  
tblPoly.FeatureID) INNER JOIN tblPointsData ON tblPoly.PolyID =  
tblPointsData.PolyID;
```

APPENDIX 4

Installing IMSS on PC

Creating default (P,Q) drive mappings on stand alone pc's

Because of the dispersed nature of our organisation and the fact that we do not have good network connectivity, a standard system for deploying data to reserves was required. The drive letters P and Q were selected as default locations for storing database and GIS data respectively.

In order to accomplish this drive mapping process on stand alone PC's, so-called 'virtual drives' need to be created. The method used to do this will differ if you are using Windows 9x or Windows NT.

7.1) Windows 9x users

Create two subdirectories under d:\ as follows:

```
d:\gisdata
d:\dbases
```

Edit the file entitled c:\autoexec.bat using the notepad text editor that was installed along with Windows. If you are unable to find a file entitled autoexec.bat, create a blank notepad document, and save it to this name when you have completed the edits as outlined below.

This is a example of what must be added to the Autoexec.bat:

```
subst P: d:\gisdata
subst r: d:\dbases
```

Reboot your computer. You should now see two additional drives (P & Q) when you open Windows Explorer.

7.2) Windows NT users

Windows NT has built in networking features that will allow you created shared folders which other users on the network can access. These networked folders can also be mapped to 'virtual drives'. A neat trick is to simply map your own shared folders to virtual drives on the same computer. To do this,

Create two subdirectories under d:\ as follows:

```
d:\gisdata
d:\dbases
```

In Windows Explorer, right-click on the d:\gisdata folder and select 'sharing' from the popup menu. Check the 'shared as option'.

8) Creating default drive mappings in a networked environment

9) Installing CNC database on standalone PC's:

Copy files 'InfrastructureTool' and 'InfrastructureData' into dir: P:\InfrastrutureTool

If you for some reason want to change the location of the data do the following

Copy the 'InfrastructureTool' and InfrastructureData file onto any location and link the two together by executing 'InfrastructureTool' with the shift button pressed, delete all the tables within it and relink it to the current location of the 'InfrastructureData' file with: file – Get External Data – link tables.

10) Create ODBC DSN (Data Source Name) Entries. This process is described in section 3.3.2 of thesis.

11) copy the CNCINF_Infrastructure.avx file into the ArcView directory:
ESRI\AV_GIS\ARCVIEW\EXT32

Installation complete.

APPENDIX 5

The full text of the notes for initial ArcView training course presented between May and June 1999 have been included. However, certain large pictures have been removed to prevent this document from becoming unwieldy. The full version of the notes (with all pictures), if required, can be obtained from the GIS staff.



ArcView Training for CNC

Tim Sutton and Helen de Klerk



DAY 1: Basics of GIS theory

What is GIS?

- a system (of hardware, software, and data) that allows the visualisation **and analysis** of geographically referenced data (Dueker 1979:106 cited in Maguire 1991, Clarke 1997).
- ‘an organised collection of computer hardware, software, geographic data, and personnel designed to efficiently capture, store, update, manipulate, analyse, and display all forms of geographically reference information (ESRI 1995)

Types of GIS data:

Vector

- Points (e.g. locality data for species; locality of bore holes and buildings)
- Lines & polylines (e.g. roads and rivers)
- Polygons (e.g. boundaries of communities or vegetation types)
- Z (height)

Raster

- Landsat:
25m pixel resolution
Gives error of up to 200m in mountainous area or 50 m in flat areas (increases to edge of footprint)
Consists of 3 bands. We have coloured these bands as follows using false colour:
Band 3 – visible red (absorbed by vegetation) – Display Colour: Blue
Band 4 – near infrared (highly reflected by vegetation) – Display Colour: Green
Band 5 – medium infrared (absorbed by vegetation) – Display Colour: Red

Orthophotos (digital aerial photo's – not the same as paper orthophoto – no contours)

3.75m pixel resolution

Gives up to 300m in mountainous areas or 20-30 m in flat areas

Consists of 3 bands. We have recoloured with a technique called linear ramping to remove extreme differences in exposure so that at a large scale (zoomed in) the colour

is smooth over the mosaiced (tiled) images. However, this technique results in the image having a 'greyed out' appearance at a small scale (zoomed out).

Attribute data

All spatial files in a GIS are linked to internal attribute tables. In ArcView, tools exist to link spatial data to external data (e.g. SOB data in the Access database)

Input tools: getting the data into the computer

Digitisers

Digitising amounts to tracing features on a paper map, such as the range of a species or the extent of a habitat type, using a 'puck' (or keypad), which is like a mouse. The paper map is laid on a digitising tablet, which contains a fine electronic wire grid that registers the relative location of the crosshairs on the keypad.

Scanners

HUD

GPS

Projections and datums: representing a 3-D spherical world on a 2-D flat map

Projection

The earth is not a perfect sphere (the meridian is longer than the equator), so straightforward geometry cannot be used to project the spherical earth onto a flat map (piece of paper or computer screen). Projections are used to do the mathematics involved. However, no one projection can perfectly represent the area, shape, angles, and distances of a spherical world on a flat map. CNC has standardised on the Lambert (Lo21) projection as it gives the most accurate area and distance readings over the whole province. The image data supplied to you is already projected to Lambert, but vector data is stored in geographic. Tim has developed a tool that will automatically project the vector data into Lambert for you (CNC Utilities\Use Lambert).

An additional aspect to the problem of representing a spherical world on a flat map is that the earth's surface is not smooth (which is what the mathematics of the projections assume), but has relief (i.e. mountains and valleys. To account for this, a datum (based on a spheroid) is used. The datum to be used is determined by the Department of Land Affairs (Trig. Surveying?). Up until recently, the datum was Clarke (Cape) 1880. However, in 1999 the standard was changed to WGS84. WGS84 represents an improvement, being based on new methods and more sophisticated equipment. Note that all data supplied to you is already in WGS84 format.

The ArcView manual covers these topics in Chapter 9 (p. 149)

Geographic coordinates (lat/lon)

Cartesian coordinates (xy)

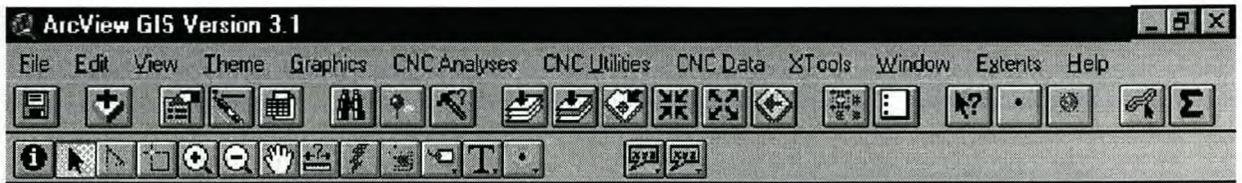
Azimuthal

Cylindrical
 Conic
 Datums (Aaaaargh again!)
 Cape Datum (Modified Clarke 1880)
 WGS84

Exploring the data supplied to you

Use Windows Explorer to navigate to the 'Q: drive'. **Remember that the Q and P drives are 'virtual drives' mapped through the 'C:\My documents' folder. Be very careful never to move or delete the 'C:\My documents' folder.** All Image data is stored in the 'Q:\3_Lambert_WGS84' directory. All vector data is stored in the 'Q:\2 Geographic' directory. In these two directories you will find a number of sub-directories, in which are house various directories for specific GIS data layers. If you navigate into a directory for a particular data layer, you will find a .doc or .txt file which will tell you about that data layer. For example, in Q:\2 Geographic\Landuse\Conservation\Provincial Reserves\Reserves Wcape\CNC Boundaries you will find the file 'ca_cnc.doc'. This file will give you details on the acts under which these property holdings have been proclaimed, who did the digitising, and how the data was checked.

A few basics on Windows



The Menu Bar

The Button Bar (a button only stays active for one execution of the task it represents. If you want to repeat the action, you have to select it again.)

The Tool Bar (select a tool once and it will stay active until you select another tool)

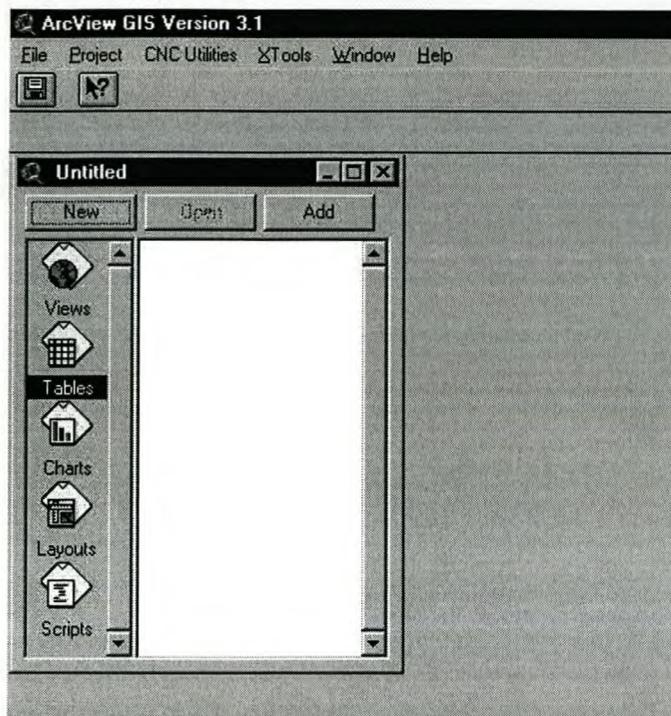
Minimise

Maximise

Close window

Use the drop down list under the 'Window' menu option to switch between the project, view, table, and layout windows.

The basics of ArcView



This is the Project window.

- **Views:** for visualisation and query of spatial data
- **Tables:** for visualisation and query of non-spatial data
- **Charts:** rather use EXCEL or PowerPoint to make graphs and charts
- **Layouts:** for arranging map data and query results to print out
- **Scripts:** for increasing the functionality of the programme (you can largely ignore this function)

Use the 'File' menu to give you project a new name (i.e. rename the 'untitled' window) and to **save you work regularly – ArcView has NO automated save function.**

Also in the 'File' menu you will find an 'Extensions' option. Make sure that the following extensions are ticked on and that there is a tick in the 'default box'

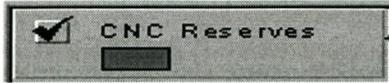
- CNC Extensions (Tims tweeks)
- ER Mapper (to enable you to read Chris's fancy images)
- Xtools (to calculate meters and hectares)

Adding data to the view and reclassifying the data (changing the symbolisation)

Double click on VIEW
Click on the 'add theme' icon



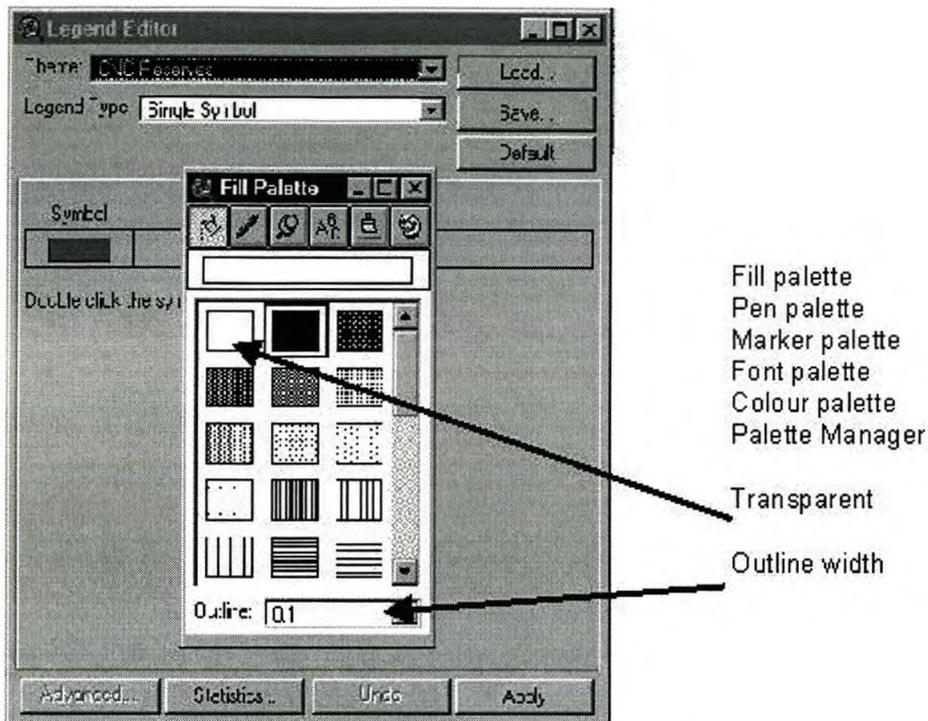
- Navigate to Q:\2 Geographic\Landuse\Conservation\Provincial Reserves\Reserves Wcape\CNC Boundaries\ca_cnc.shp. A button will appear in the View window. Check the tick box:



Use the 'identify' and 'pan' buttons to find your reserve, and then use the 'zoom in' button to draw a box around your reserve and zoom in to the extent of that box



Make reserves transparent and the boundary green as follows:
Double click on the CNC Reserves layer (shown above) anywhere but on the check box. This will bring up the Legend Editor. Double click on the 'Symbol' box. This will bring up the 'Palette' window. Choose the transparent option as depicted below. Change the 'Outline' width to be thicker (about 2). Click on the Colour palette and change the **outline** colour to green.



- Use the 'add theme' button again to add Q:\2 Geographic\Natural Environment\Biotic\Flora\Vegetation Types\General\veg_wc.shp

You will no longer be able to see the CNC Reserves layer, as the Vegetation layer will be sitting above it, obscuring all layers below. Click and drag the CNC Reserve layer to the top of the list



Recolour the layer according to the vegetation type as follows:
Double click on the Vegetation layer (anywhere but on the check box). This will bring up the Legend Editor. Click on the drop down list in the 'Legend Type' box

and choose 'Unique Value'. Click on the drop down list in the 'Values Field' and choose 'veg_type'. Once you have the colour scheme of your choice selected, click apply.

- Use the 'add theme' button to add Q:\2 Geographic\Water Management\Rivers50\rivers50.shp

Recolour the layer using one of the standard river legend files as follows:

Double click on the rivers50 layer (anywhere but on the check box). This will bring up the Legend Editor. Click on the 'Load' button at the top right of the Legend Editor window. Navigate to Q:\2 Geographic\Water Management\Rivers50 choose river50_type.avl for the rivers to be classified according to whether they are perennial or non-perennial, or choose river50_riversys.avl for the rivers to be classified (coloured) according to the river system in which they occur.

- Use the 'add theme' button to add Q\3_Lambert_WGS94\landsat_ortho_comb.alg.

Note that in order to see this file in the Q\3_Lambert_WGS94 directory you will need to select 'Image Data Source' from the 'Data Source Types' box at the bottom left of the 'add theme' window. You will notice that when you check this layer on that it does not appear in the view window. This is because all image data is stored in Lambert projection while the vector data is stored in unprojected (geographic) format. This problem is easily rectified by choosing 'CNC Utilities\Use Lambert'.

Querying the data and using tables

Identifying records in the table that were selected in the view

Use the 'Select Feature' tool to select a reserve in the CNC Reserve layer. Click on the 'Open Theme Table' button to open the attribute table for the CNC Reserve layer. Use the 'Promote' button to promote the record for the reserve polygon that you selected in the view to the top of the list. You can then read off all the details from the table for the reserve polygon that you selected.



- Database query tool

Use the 'Query Builder' button to select all reserves that fall under the Waterval office ([Res_centre] = "Waterval"). Because we selected the fill of the CNC Reserves layer to be transparent, we cannot see which reserves were selected. So, double click on the CNC Reserve layer and use the Legend Editor to make the fill solid (see notes above). When you click on apply, the reserves that meet the criteria of ([Res_centre] = "Waterval") will be highlighted in yellow.

- Spatial selection using select tool



- Theme by theme query

Adding your own data (HUD)

Zoom into your reserve. For digitising new data, you want to work at least at 1:50,000 (and preferably at 1:10,000) scale, otherwise the data you digitise may be many meters out on the ground. Check the scale that you are working at in the. Use the landsat_ortho_comb to locate a few patches of aliens.

Add a blank data layer to your view by using View\New Theme. In the 'New Theme' window select feature type as polygon, and save the file as my_aliases.shp (for example) in the **Q4 my GIS data**\ directory. You will see that there is a dashed box around the check box of my_aliases.shp. This indicates that you are in editing mode. Use the 'Draw Polygon' tool to trace the outline of a few alien patches.

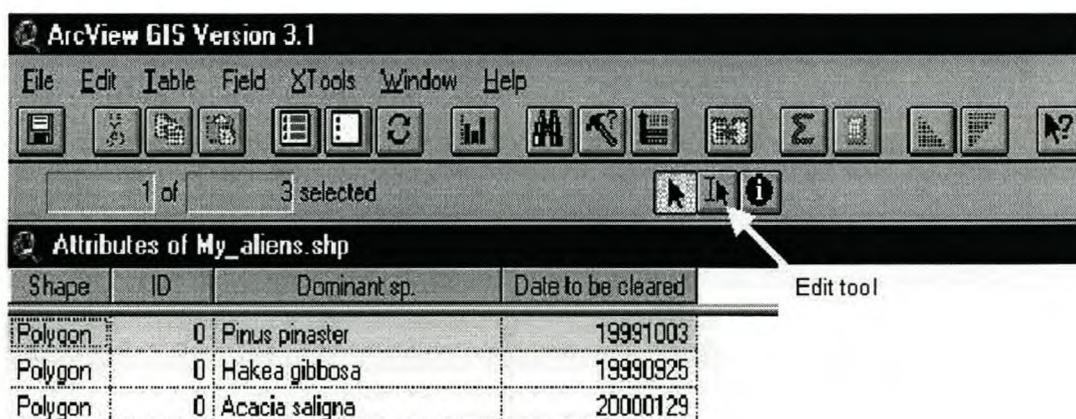


Should you wish to edit the boundary of one of your polygons, use the 'Vertex' tool to either click on existing vertices and move them to the desired location, or to click on a line between two vertices to create a new vertex which can then be positioned at the desired location.



The process of creating a new shape file automatically creates a blank attribute table for that new shape file. If you want to add data for the shape file such as dominant species and date to be cleared, use the 'Select Feature' tool to select the alien polygon for which you wish to add data. Use the 'Open attribute Table' button to open the attribute. In the tables window, use Edit/Add Field to add a new column. In the Name field, enter 'Dominant sp.', in the Type field use the drop down list to select string (=text), and make the Width about 50. Use the 'Edit' tool to enter the data into the Dominant sp. column for the polygon you have selected (e.g. *Pinus pinaster*). Go back to the View window and select the next alien polygon. Go to the Tables window and enter the dominant species for that polygon. Repeat the procedure until you have entered data in the Dominant sp. column for all of the alien polygons that you digitised.

To add a second column for the date by which the alien polygon (FWWP task) should be cleared, go to the tables window, use Edit/Add Field to add a new column. In the Name field, enter 'Date to be cleared', in the Type field use the drop down list to select date. Be warned that ArcView is very fussy about the format in which dates should be entered, namely YYYYMMDD (see Page 191 of your manuals). Go to the window and select one of the alien polygons. Use the 'Edit' tool to enter the data into the 'Date to be cleared' column for the polygon you have selected (e.g. 19991003 for the 3rd of October 1999). Repeat the procedure until you have entered data in the 'Date to be cleared' column for all of the alien polygons that you digitised. Your table will look something like this:

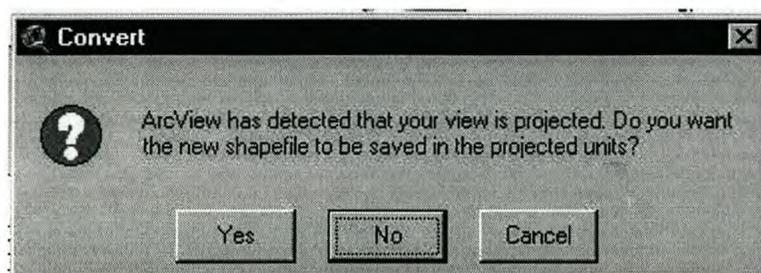


Remember to save your edits as you work either in the View window through Theme\Save Edits or, in the Tables window by using Table\Save Edits. When you have finished adding both spatial (polygons) and non-spatial (attributes) data, use either the Theme\Stop Editing in the View window, or the Table\Stop Editing in the Tables window.

Creating new data from existing data

- Create a conservancy from the Cadastral data layer as follows:
Add the cadastral layer Q:\2 Geographic\Social and Cultural Regions\Cadastral\Cadastre.shp. Make sure that the Cadastral layer is active. Use the 'Feature Select' tool to select all the cadastral units which are to make up the

conservancy (hold the shift button down to select more than one feature at a time). Select 'Theme\Convert to Shapefile' to save just the selected cadastral units to a new data layer. **Remember to save the new layer to Q\4 My GIS data.** Once you have saved the new data layer, the following message window will pop up:



It is important that you choose the 'NO' option. A second message window will then pop up asking if you would like to add the shapefile as a theme to the view. You can reply 'Yes' to this.

Labelling data in a view

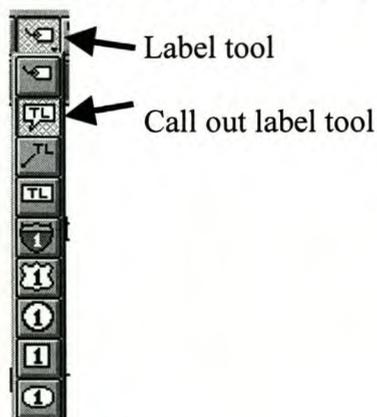
- Automatically label all reserves by reserve name

Make sure that the CNC Reserve layer is active. Also make sure that either no or all features in the data layer are selected (e.g. 'Theme\Clear Selected Features') otherwise only those features that are selected will be labelled. There are two ways to select the label field by which the layer should be labeled. The first is to go to 'Theme\Properties\Text Labels' and to use the drop down list in the 'Label Field' to choose 'Res_name'. Then use 'Theme\Autolabel'. This method will often have to be used for new data layers that you have created (either through digitising or by querying existing data). The second is to go straight to 'Theme\Autolabel' and again use the drop down list in the 'Label Field' field to select Res_name (or Manager if you would rather having a map depicting who the various reserves are managed by).

If you wish to remove these labels use 'Theme\Remove Labels'.

- Label 3 of the reserves individually with 'callout' labels according to their telephone numbers

Use 'Theme\Properties\Text Labels' to select Tel from the drop down list in the 'Label Field'. Then click on the bottom right hand corner of the 'Label' tool to select



the callout type of label.
 To rotate/angle text on map
 Hilite:~:; double click

Importing data from Spreadsheets

Make sure that your spreadsheet is in the following format:

DS	M S	SS	DE	M E	SE	Species recorded
33	04	21	19	05	21	Leopard
33	10	14	19	02	12	Dassie
33	08	47	19	20	18	Meerkat

In Excel save your spreadsheet file to Q\4 My GIS data first in Excel format (.xls) (e.g. species.xls) and then use the 'File\Save As' option to save your file in 'Comma delimited' format (.csv). ArcView requires spreadsheet data to be in comma delimited format but to be called 'txt' rather than the 'csv' that Excel produces. So in Windows Explorer navigate to where you have saved the file (Q\4 My GIS data). Rename the file to have a 'txt' extension (e.g. rename species.csv to species.txt). When you hit enter, Explorer will give you an error message – you can ignore it.

In Arc View use the 'CNC utilities\Import DMS text file' option to import the species.txt spreadsheet file. Follow the instructions that the programme gives you. This will open a file in the view call species.dbf. You can save this as a shape file using 'Theme\Convert to Shapefile'

The 'CNC utilities\Import DMS text file' option converts degrees-minutes-seconds (dms) data to decimal-degree (dd) format which is the format required by ArcView. It uses the following equation to do so:

$$DD = (D + (M/60) + (S/3600))$$

Layouts and printing your maps and analysis results

Get all the data you want on the map added in the view, symbolised correctly, and labelled where required. In the project window click on the 'Layout' icon and then click on the 'New' Button. This will bring up a new layout with an empty page. **It is very important that the very first thing you do is to set the page size in 'Layout\Page Set up'**. Preferably do **not** leave this option as 'Same as Printer' (e.g. make is A4). Use the 'View Frame' tool to draw a box on the page indicating the extent of the page that you wish your map to fill. This will pop up a 'View Frame Properties' dialogue box. Choose the view in which the data you wish to print resides. Click on the bottom right hand corner of the 'View Frame' tool to drop down other view options 'Legend Frame', 'Scale', and 'North Arrow'.



To add a heading to the map, use the 'Text' tool. The font, text size, text style, and even colour can be edited by accessing the normal palettes through 'Window\Show Symbol Window'



Also use the 'Text' tool to add an information box to your map stating the projection, datum, and scale of the map, as well as who created it and when it was created (date). You can also put a note as to what directory the project is stored in.