

**AN EXPERT SYSTEM UTILITY PROTOTYPE FOR THE
EVALUATION OF ENVIRONMENTAL IMPACTS OF
CIVIL ENGINEERING PROJECTS**

Prepared by : K.G.Mercer
Postgraduate student in the Department of Civil Engineering

Prepared for: The Department of Civil Engineering at the University of
Cape Town

June 1994

Thesis prepared for the partial fulfilment of the requirements for the Degree of MSc in Civil Engineering.



The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

Declaration by Candidate

I, Kenneth George Mercer, hereby declare that this thesis is my own work and that it has not been submitted for a degree at another university.

Signed by candidate

K G Mercer

June 1994

To Bridget

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to the following :

My supervisors, Professor J B Martin and Professor K Mac Gregor, for their interest and advice during my period of post-graduate study.

My wife, Bridget, for her patience, support and help in typing and preparation of the manuscript.

Professor M O De Kock and the staff of the Department of Civil Engineering for their help and assistance.

My parents, for their support and assistance.

Wendy MacMillan, for her assistance in the preparation of the drawings and illustrations.

SYNOPSIS

The idea of this thesis is to create a prototype primary environmental impact assessment computer program for use by engineers in the planning of civil engineering projects. The idea is that this system will be used early in the planning phase of projects, to identify potential negative environmental impacts and there by iteratively modify proposals before further work on them is undertaken. The result would be a saving in time and money to clients by minimising abortive work.

The problem is highly complex in that there are almost limitless combinations of environmental characteristics available for selection for the evaluation of a project proposal. Each characteristic in turn, is able to assume a varying level of importance. The structure of the system developed to solve this problem uses a combination of an expert system shell and conventional computer programs.

The trend of modern environmental impact assessment methods is towards qualitative methods of assessment. As a result, the expert system shell SYNAPSE was chosen to provide a decision making facility for each environmental characteristic using heuristic techniques. Due to certain limitations of SYNAPSE a number of computer programs were developed to enable other supporting capabilities to be available to users. These programs were written in the Turbo Pascal language together with Turbo Vision, an object-orientated application framework for providing features such as windowing and event-driven regimens.

The thesis reviews expert system technology in detail followed by the concept of environmental impact assessment and corresponding assessment methodologies. The concept of using expert systems in environmental management is not new with several papers published to date. The thesis proposes a method whereby a set of rulebases (each specialising in a specific environmental characteristic) can be assembled for the specific problem at hand, and each environmental characteristic assigned a weighting according to the importance of that characteristic to the problem. The results of the SYNAPSE consultation are in the form of real value logic and are finally combined and presented to the user in a graphical qualitative form.

The programs were developed using certain design criteria, the most important of which was user friendliness. The program utility is rich with features including an installation program, help facilities, screen 'saving', pulldown menus, picklists, file viewers, editors and graphics. The major problem encountered in the development of the program utility was the memory requirements of the various computer programs. To solve this a sequence of operations had to be set up whereby

the program utility and SYNAPSE were swapped alternatively in and out of memory as required for analysis purposes. The thesis also deals with other aspects such as the interfacing of expert systems directly to databases and the manipulation of result data from SYNAPSE consultations.

Results of the system usage indicated that there does appear to be a promising future for this form of analysis. There are however several aspects that will require attention by future users of the system. These include the accuracy of the rulebases, the division of rulebase categories, the temptation to tamper with weightings to produce desired results, the subjective nature of many of the rulebase questions and the maintenance and continual refinement of the rulebases.

Finally proposals are made regarding future development work to this system. These include the selection of a new expert system shell and the formation of a team of experts from various fields to maintain rulebases relating to their own specialities. Furthermore it was recommended that a set of generalised guidelines be drawn up for helping with the preparation of different environmental analyses and another set of guidelines for answering questions during consultations. The final two recommendations were that of monitoring security of rulebases and developing an interface to allow the expert system to access data directly from databases.

TABLE OF CONTENTS

<u>Chapter</u>	<u>Page</u>
Title	i
Declaration	ii
Dedication	iii
Acknowledgments	iv
Synopsis	v
Table of Contents	vii
List of Illustrations	xii
Nomenclature	xv
1. Introduction	1.01
1.1 The Initial Motivation behind the Research Topic	1-02
1.2 General Statement of the Problem	1-03
1.3 Aim of the Thesis	1-03
1.4 Plan of Development	1-04
2 Literature Review of Artificial Intelligence	2-01
2.1 History	2-02
2.2 Overview of AI Application Areas	2-03
2.2.1 Game Playing	2-03
2.2.2 Automated Reasoning and Theorem Proving	2-03
2.2.3 Expert Systems	2-03
2.2.4 Natural Language Understanding and Semantic Modelling	2-04
2.2.5 Modelling Human Performance	2-04
2.2.6 Planning and Robotics	2-04
2.2.7 Languages and Environments in AI	2-05
2.2.8 Machine Learning	2-05
2.3 Summary	2-06
3. Rule-based Expert Systems	3-01
3.1 Introduction	3-02

3.2	Overview of 'Knowledge Engineering'	3-03
3.3	The difference between Expert Systems and Conventional Computer Programmes	3-05
3.4	The Structure of Expert Systems	3-05
3.4.1	Knowledge Base	3-05
3.4.2	Context	3-07
3.4.3	Inference Mechanism	3-07
3.4.4	Explanation Facility	3-07
3.4.5	Knowledge Acquisition Facility	3-07
3.4.6	User Interface	3-08
3.5	Problem Solving using Expert Systems	3-08
3.5.1	Types of Knowledge representation	3-08
3.5.1.1	Logic-based Representation	3-09
3.5.1.2	Rule-based Representation	3-09
3.5.1.3	Network-based Representation	3-10
3.5.2	Approaches to Problem Solving Strategy	3-11
3.5.2.1	Derivation	3-11
3.5.2.2	Formulation	3-11
3.5.3	Control Strategies	3-12
3.5.3.1	Forward Chaining	3-12
3.5.3.2	Backward Chaining	3-13
3.5.3.3	Mixed Chaining	3-13
3.5.3.4	Other Control Strategies	3-13
3.6	Dealing with Uncertainty in Expert Systems	3-14
3.7	Problems Relating to Expert System Technology	3-15
3.8	Selecting a Problem for Expert System Development	3-15
3.9	Testing and Evaluation of Expert System Performance	3-17
3.10	The SYNAPSE Expert System Shell	3-18
3.10.1	Overview	3-18
3.10.2	Inference Networks and Rules	3-19
3.10.3	SYNAPSE Mathematics and Uncertainty	3-20
4.	Environmental Impact Analysis and the Rulebase Development	4-01
4.1	Introduction	4-02
4.2	The Principals Behind IEM	4-02
4.3	The Integrated Environmental Management Procedure	4-03

4.3.1	Stage 1: Plan and Access	4-03
4.3.2	Stage 2: Decision	4-06
4.3.3	Stage 3: Implementation	4-06
4.4	The Role of the Engineer in IEM	4-08
4.5	Impact Assessment Methodology	4-09
4.6	The Proposed Methodology	4-12
4.7	Rulebase Design Criteria	4-14
4.8	The Knowledge Acquisition Procedure	4-15
4.9	Environmental Characteristics in Project Evaluation	4-16
4.9.1	Physical	4-16
4.9.2	Ecological	4-17
4.9.3	Potential Land Use/Landscape Character	4-17
4.9.4	Cultural Resources	4-18
4.9.5	Infrastructure	4-18
4.9.6	Social and Community Services	4-18
4.9.7	Pollution	4-19
4.9.8	Risk and Hazard	4-19
4.9.9	Health and Safety	4-20
4.9.10	Cumulative and Synergistic Effects	4-20
4.10	Project Evaluation	4-20
5.	Object-Orientated Programming and the Program Structure	5-01
5.1	Introduction	5-02
5.2	Object-Orientated Programming	5-02
5.2.1	Encapsulation	5-03
5.2.2	Inheritance	5-03
5.2.3	Polymorphism	5-03
5.3	Turbo Pascal and Turbo Vision	5-04
5.4	Basic Principals of the Program Operation	5-06
5.5	Design Criteria	5-07
5.6	Program Design and Structure	5-08
5.6.1	Overall Operational Review	5-08
5.6.2	Detailed Operational Breakdown	5-11
5.6.2.1	Object Hierarchy	5-11
5.6.2.2	Program Flow Charts	5-14
5.7	Data Storage Structures	5-19

5.7.1	Collections	5-19
5.7.2	Streams	5-21
5.7.3	Resource Files	5-23
5.8	Rulebase and Program Interfaces	5-25
5.9	Databases	5-27
5.10	Manipulation of Result Data	5-30
6.	An EIA Worked Example	6-01
6.1	Introduction	6-02
6.2	Environmental Assessment of Park Rynie Sewage Treatment Works	6-02
6.3	Summary of Report Assessment	6-07
6.4	Site Assessment Using the Expert System Utility	6-09
6.5	Comments on Results of Utilities Assessment	6-12
6.6	Comparison of the Computer Analysis With the Consulting Engineer's Analysis.	6-13
6.6.1	Overall Weighted Assesment	6-13
6.6.2	Comparison of Each Characteristic	6-14
6.6.3	Other Aspects of the Program	6-16
7.	Conclusions	7-01
7.1	Literature Survey of Expert System Technology	7-02
7.2	EIA and the Rulebase Development	7-02
7.3	Development of the Program Utility	7-03
7.4	Operational Review	7-04
8.	Recommendations	8-01
8.1	SYNAPSE	8-02
8.2	Rulebase Maintenance	8-02
8.3	Weightings and EIA Preparation	8-02
8.4	Rulebase Use	8-02
8.5	Security	8-02
8.6	Database Interfaces	8-03
8.7	Program Application	8-03

Chapter (Continued)

9. References	9-01
10. Bibliography	10-01

Appendix 1

Theory and Derivation of RVL Values

Appendix 2

AWK Program for the Interfacing of Rulebases and Databases

Appendix 3

Rulebase Listings

Appendix 4

Program Listings

Appendix 5

Implementation of the Expert System Utility

- 5.1 Introduction
- 5.2 Installation
- 5.3 Program Introduction
- 5.4 System Menu
- 5.5 Rulebase Management
- 5.6 Windows Menu
- 5.7 Databases Menu
- 5.8 Project Analysis Menu
- 5.9 Result Analysis Menu
- 5.10 Floating Menu
- 5.11 Help Facilities
- 5.12 Exiting the System

Appendix 6

Program Design Charts

LIST OF ILLUSTRATIONS

<u>Figures</u>	<u>Title</u>	<u>Page</u>
Chapter 3		
3.01	Exploratory Development Cycle for Rulebased Expert Systems	3-04
3.02	Structure of an Expert System	3-06
3.03	A Blocks World	3-09
3.04	Semantic Network Description of a Bluebird	3-10
3.05	Floor System Design Inference Network	3-12
3.06	Typical Structure of a SYNAPSE Inference Network	3-19
3.07	Forward and Backward Chaining Strategies	3-20
3.08	TRUE and FALSE Positions	3-21
3.09	The Number Line Showing and RVL Range	3-21
3.10	The Number Line Showing 'as good as' Values	3-22
3.11	Node Resolved as PROVED	3-22
3.12	Node Resolved as UNRESOLVED	3-22
Chapter 4		
4.01	Integrated Environmental Management Flow Diagram	4-04
4.02	Project Evaluation Using Multiple Rulebase Analysis	4-21
Chapter 5		
5.01	Program Operation	5-09
5.02	Program Operation for Running DOS Batchfiles	5-10
5.03	Turbo Pascal Object Hierarchy	5-12
5.04	Complete Program Object Hierarchy	5-15
5.05	Object Structure of TEditDialog2	5-18
5.15	Graphical Depiction of a TCollection Object	5-20
5.16	Graphical Depiction of the ExptList Stream	5-22
5.17	Resource File (Batch.rez) Structure	5-24
5.18(a)	Answer File - Plain ASCII Format	5-25
5.18(b)	Answer File - Delimited ASCII Format	5-25

5.19(a)	Result File - Plain ASCII Format	5-26
5.19(b)	Result File - Delimited ASCII Format	5-26
5.20	Flow Chart of Rulebase Information Retrieval for any Database Format	5-29
Screen 1	A Typical Turbo Vision Environment	5-05

Chapter 6

6.01	Existing Land Usage in the Region	6-05
6.02	Alternative Sites Adjacent to Park Rynie With Respect to Land Usage.	6-06
6.03	Quantitative Evaluation of Site.	6-08
6.04	Batchrun for Site A	6-10
6.05	Batchrun for Site C	6-10
6.06	Analysis of Site A	6-11
6.07	Analysis of Site C	6-11

Appendix 5

Screen 1	Installation Program
Screen 2	Unpacking of System Files
Screen 3	Information Window Showing Program Name
Screen 4	Disclaimer Window
Screen 5	System Menu
Screen 6	System Colour Selection
Screen 7	Temporary Exit to DOS
Screen 8	Screen Saver
Screen 9	Rulebase Management Menu
Screen 10	List of Rulebases to View or Edit
Screen 11	Viewing/Editing of Multiple Rulebases
Screen 12	SYNAPSE Integrated Environment
Screen 13	Changing Directory
Screen 14	Windows Menu
Screen 15	Disorderly Arrangement of Rulebase Windows
Screen 16	Tiled Arrangement of Rulebase Windows
Screen 17	Cascaded Arrangement of Rulebase Windows
Screen 18	Database Menu

Screen 19	Project Analysis Menu
Screen 20	Batchrun Selection
Screen 21	Inputting New Batchrun Name
Screen 22	Selected Components and Weightings of a Batchrun
Screen 23	Expert System Control Centre
Screen 24	Listing of Registered Rulebases
Screen 25	Selecting a Rulebase to Modify Registration Information
Screen 26	Adding/Editing of Rulebase Registration Information
Screen 27	Selecting an Expert System Rulebase to Execute
Screen 28	Selecting a Batchrun to Execute
Screen 29	Typical SYNAPSE Question
Screen 30	Typical SYNAPSE Resolved Goal
Screen 31	Typical SYNAPSE Reasoning Process
Screen 32	Result Analysis Menu
Screen 33	Selection of Batchrun for Results Review
Screen 34	Graphical Presentation of Results
Screen 35	Results in Terms of RVL Values
Screen 36	Floating Menu
Screen 37	Typical Help Window
Screen 38	Exiting Confirmation Window

Appendix 6

5.06	System Menu - Structure
5.07	Rulebase Menu - View Rulebases
5.08	Rulebase Menu - Edit Rulebases
5.09	Rulebase Menu - SYNAPSE Integrated Environment
5.10	Rulebase Menu - Change Directory
5.11	Window Menu - Menu Commands
5.12(a)	Project Analysis Menu - Project Batch Management
5.12(b)	Project Analysis Menu - Project Batch Management(cont.)
5.13(a)	Project Analysis Menu - Expert System Man.(page 1)
5.13(b)	Project Analysis Menu - Expert System Man.(page 2)
5.13(c)	Project Analysis Menu - Expert System Man.(page 3)
5.13(d)	Project Analysis Menu - Expert System Man.(page 4)
5.13(e)	Project Analysis Menu - Expert System Man.(page 5)
5.14	Result Analysis Menu - Structure

NOMENCLATURE

Artificial Intelligence Terminology

AI :	Artificial Intelligence.
Artificial Intelligence :	A subfield of computer science concerned with the concepts and methods of symbolic inference by a computer and the symbolic representation of knowledge to be used in making inferences.
Backward Inference :	A control strategy for expert systems, in which the system will attempt to find a piece of information that leads to the establishment of some goal.
Cognition :	Knowing, perceiving or conceiving.
Domain :	Scope of the field of knowledge under investigation.
Forward Inference :	A control strategy for expert systems, in which the system will attempt to find a goal given the results of some questions.
ES :	Expert System.
Expert System :	A computer program which makes decisions that it can justify to humans.
Expert System Shell :	In a shell the knowledge and control elements of the program are separated and the knowledge base is empty (see figure 3.02). Knowledge engineers can take an 'empty shell' and create a new knowledge base appropriate to their application.
Expertise :	Specialised knowledge or judgement in a field.
Heuristics :	A heuristic is a useful but potentially fallible problem-solving strategy such as checking to make sure an unresponsive appliance is plugged in before assuming it is broken.

Inference Network :	Hierarchal representation of knowledge in the form of rules forming an algorithm through which information is processed to reach a conclusion.
IE :	Integrated Environment.
Knowledge :	Specific information about a subject.
Knowledge Base :	A collection of facts and other information concerning a domain.
Reasoning :	The act or process of drawing conclusions from facts or evidence.
Rule :	A conditional statement made up of two parts. The first being the idea, fact or decision to be established, the second is the list of conditions that should apply in order to establish the first.
Rulebase :	A collection of rules concerning a particular subject area.
RVL :	Real Value Logic.
Search :	A problem solving technique that systematically explores a space of problem states ie intermediate steps in a reasoning process.
Uncertain Reasoning :	The act or process of drawing conclusions from evidence which cannot totally or accurately be determined.

Programming Terminology

AWK :	Aho, Weinberger, Kerrighan - Pattern matching programming language.
Ancestor :	An object from which another object is descended.
Descendent :	An Object that inherits all its data fields and methods from its immediate ancestor, which may have inherited properties from its own ancestor.

DOS :	Disk Operating System - developed by Microsoft.
Encapsulation :	The combination of data and the code which acts on the data into one self-contained object.
Expert System Utility :	A program utility that features an expert system as one of its components. (see Program Utility).
Inheritance :	Ability of new objects to inherit the characteristics and capabilities of previously defined objects.
Instance :	A variable of an object type.
Method :	A procedure or function definition in an object type. Methods describe the operations that objects know how to perform.
Object :	A variable (that may be addressed by a pointer) that has the characteristics and capabilities of polymorphism, inheritance and encapsulation.
OOP :	Object-Orientated Programming.
Stream :	Collection of objects on their way somewhere.
Polymorphism :	A concept referring to the ability of objects to assume different forms at run time.
Program Utility :	A term used to describe a combination of various computer program applications that run consecutively to solve a common problem.
TP :	Turbo Pascal, High-level object-orientated programming language developed by Borland.
TV :	Turbo Vision, an object-orientated application framework (that is a feature of Turbo Pascal) for providing windowing and event-driven capabilities plus much more.
Utility :	See Program Utility.

Virtual method : A procedure or function by which polymorphic objects are created.

Environmental Impact Analysis

EIA : Environmental Impact Analysis

IEM : Integrated Environmental Management

EES : Environmental Evaluation System

WRAM: Water Resources Assessment Methodology

CHAPTER 1

INTRODUCTION

*"The real danger of the computing age is not
that computers will think like people, but
that people will think like computers"*

Frank Romano (US Publisher)

Contents

1. Introduction

- 1.1 The Initial Motivation Behind the Research Topic
- 1.2 General Statement of the Problem
- 1.3 Aim of the Thesis
- 1.4 Plan of Development

INTRODUCTION

1.1 The Initial Motivation Behind the Research Topic

Often environmental impact assessments (EIA) on any proposed urban or rural development are completed late into the planning phase and thus have a limited influence on the overall design of the project. Civil engineers generally do not possess sufficient environmental expertise to be able to make a significant contribution in these early phases, even though their overall awareness may be high, hence the need for easily accessible assistance.

The original idea was that this thesis should consist largely of developing a computer program to assist engineers in assessing the environmental impact of proposed projects. With the aid of this computer program engineers can begin to assess projects in the conceptual stage. Using an iterative design approach ideas and changes in the design can be incorporated and the analysis re-run until a sufficiently positive environmental assessment is made. As far as possible the program was to be made versatile, user friendly and easily modifiable to incorporate future changes, new ideas or changes to legislation. The envisaged benefits of the program would be a more effective planning process. Environmental problems could be identified early and the design modified or the project relocated before any major design or work is undertaken, thereby minimising both risks and costs to the client.

In accordance with modern environmental impact assessment trends that concentrate on qualitative methods of analysis, it was decided that a heuristic form of computer evaluation should be used. This form of decision making process can best be implemented using an expert system in which expert's "skills" are captured and modelled using rules that the expert system can re-use for each consultation.

As expert systems lack certain support features, it was decided to develop an integrated system combining both the unique advantages of expert systems together with the capabilities of conventional computer programs. It was envisaged that when the system was completed it would provide a tool for engineers, environmentalists and other parties to conduct primary environmental impact analysis with a relatively high degree of accuracy.

1.2 General Statement of the Problem

Once a project has been conceived and planning work begun, traditional environmental impact studies have tended to be conducted separately from the engineering planning and design. The final decision for the project implementation is usually based on the results of the design process such as costs, construction time, financial feasibility etc. on the one hand and a separate but parallel EIA process on the other. Should the EIA produce an unfavourable result the project could suffer delays, cancellation, interruptions or even redesign resulting in wastage of resources.

The EIA process itself uses many environmental characteristics. These include the assessment of pollution, social impacts, physical changes to the landscape, ecological effects on the fauna and flora, legality and many others. Moreover for each project there will be a different set of characteristics that will be relevant in the analysis. For example the characteristics for evaluating a new housing complex adjacent to a lagoon will be completely different to those used for a factory located inland in an established industrial area. Furthermore the relative importance of each of these characteristics will vary from project to project. For example the significance (effect) of air pollution from a new factory on public health will be totally different if it is located adjacent to a residential area or sited at a remote location far from densely populated areas.

It is thus clearly evident that there exist almost limitless possible combinations of design factors and weightings for each project evaluated. We thus have a situation which is extraordinarily difficult to quantify or model in any way. One might ask, 'why bother to write a computer utility to evaluate these problems?'. The answer to this question lies in the fact that there are potentially significant savings in both time and money if the system as described could be used to identify potentially environmentally sensitive areas at an early stage in the project life cycle. Full EIA investigations are usually lengthy and costly undertakings, therefore there exists the need for a fast, cheap and accurate utility for carrying out primary environment impact assessments. This utility should be suitable for use by individuals other than specialists in this field to obtain expert quality assessments. The means by which this will be achieved will be by the use of an expert system.

1.3 Aims of the Thesis

- ° To investigate the development and structure of a computer programme which will be capable of processing the data for alternative proposals as well as the many variables in an Environmental Impact Assessment.

- ° To investigate the methods used in Environmental Impact Assessments in accordance with the Integrated Environmental Management Guidelines with respect to the requirements set out in Part 1.2 above.
- ° To develop a computer programme system which will meet the above requirements.
- ° To show how the system is used on an Environmental Impact Assessment.
- ° To present conclusions and recommendations on the use of the programme and possible further development which could be implemented in the future.

1.4 Plan of Development

Chapter 2

Expert systems have their roots in the field of artificial intelligence and thus this thesis begins by a brief overview of this topic. The chapter begins by reviewing the history of artificial intelligence after which the most important fields other than expert systems are briefly described.

Chapter 3

Chapter 3 is devoted to the review of the most important aspects of expert systems. The first of these is the concept of knowledge engineering whereby knowledge is captured from experts and expert systems are developed. Section 3 explains the differences between expert systems and conventional computer programs. The following two sections present the structure of a typical expert system and how knowledge is represented using networks and how information is processed in order to reach a conclusion.

An important aspect of expert systems is that of uncertainty which is dealt with in section 6. Other aspects presented are problems relating to expert system technology, selecting an appropriate problem for expert system development, the testing and evaluation of expert systems and finally the SYNAPSE expert system. SYNAPSE is the expert system shell selected for use in this thesis. Important aspects dealt with are that of SYNAPSE inference networks and rules and SYNAPSE mathematics and uncertainty.

Chapter 4

In this chapter environmental impact assessment and the development of rulebases is dealt with. The principles of Integrated Environmental Management (IEM) are presented together with the procedures and guidelines laid out for this purpose. A very important participant in IEM is the engineer whose role is reviewed in detail. Various methods used in environmental impact assessment are reviewed and this is followed by the presentation of a proposed methodology for assessing problems using expert systems.

Further aspects dealt with in this chapter include the rulebase design criteria, the knowledge acquisition procedure and the breakdown of environmental characteristics for use in project evaluation. For each different environmental characteristic a separate expert system rulebase was developed. The chapter is completed by a discussion on how these rulebases are combined together with weightings in order to undertake an evaluation of an environmental problem.

Chapter 5

As a result of the limitations of the SYNAPSE expert system, a large computer program utility was written to enable users to implement and plan the primary EIA of their projects as easily as possible. Extensive use was made of up-to-date programming techniques such as object orientated code, event driven control and turbo vision to ensure ease of future modification and code management. This chapter describes these new concepts together with the programming language and application framework used.

The chapter goes on to cover the basic principles of program operation and design criteria including the detailed operational breakdowns. This is followed by a description of the data structures used and the program interfaces together with further modifications that can be included at a later date. The chapter concludes by a description of how the result data produced by SYNAPSE consultations are processed so as to be presented in a user friendly manner.

Chapter 6

This chapter presents a detailed picture of how the computer utility works using images of captured screens. Each aspect of the utility is listed with explanations of how it works.

Chapter 7

This chapter shows how a user would go about using this system to evaluate a typical environmental problem. As an example the chapter uses a 'real life' problem and the expert who originally conducted the environmental impact study re-analyses the problem using this computer utility and the results are compared.

Chapter 8

In this chapter conclusions are drawn regarding problems encountered during the development of the system together with comments on operational aspects.

Chapter 9

A list of recommendations is drawn up based on the conclusions. These comments should form a starting point for further development work on this project.

CHAPTER 2

LITERATURE REVIEW OF ARTIFICIAL INTELLIGENCE

*"Men stumble over the truth from time to time,
but most pick themselves up and hurry off
as if nothing has happened."*

Sir Winston Churchill (British statesman)

Contents

- 2.1 History**
- 2.2 Overview of AI Application Areas**
 - 2.2.1 Game Playing
 - 2.2.2 Automated Reasoning and Theorem Proving
 - 2.2.3 Expert Systems
 - 2.2.4 Natural Language Understanding and Semantic Modelling
 - 2.2.5 Modelling Human Performance
 - 2.2.6 Planning and Robotics
 - 2.2.7 Languages and Environments in AI
 - 2.2.8 Machine Learning
- 2.3 Summary**

LITERATURE REVIEW OF ARTIFICIAL INTELLIGENCE

2.1 History

The original foundation stones of Artificial Intelligence (AI) can be traced back to philosophers and discoverers such as Rene Descartes and Galileo. Their early work in researching the division between human thinking and reality and the development of mathematics to describe the 'truths' about the natural world showed that the mind and the physical world could be separated, existing on their own and hence could be studied in themselves. Once thinking had come to be regarded as a form of computation, its formalisation and eventual mechanisation were but the next logical steps.

Charles Babbage is regarded as the earliest researcher in AI (Morrison and Morrison-19). Babbage developed a 'difference engine' for computing the values of certain polynomial functions and was the forerunner of the later 'analytical engine' (Scientific American-20). This 'engine', although never completed (until recently by researchers) was a general purpose programmable computing machine. Babbage's inspiration was to apply the knowledge of his day to liberate humans from the drudgery of arithmetic calculations. George Boole is credited with being the first mathematician to create a formal language (Boole-21). Although the fundamental role of Boolean algebra in computer logic circuitry is well known, Boole's original goal was to investigate the laws of reasoning. Further AI researchers built on this early work. These included Gottlob Frege (Frege-22) who created a mathematical specification language, Russell and Whitehead whose goal it was to derive the whole of mathematics through purely formal operations on a collection of axioms and finally Alfred Tarski (Tarski-23) who created a theory of reference.

This early work up to the early 20th century provided the intellectual prerequisite for the study of Artificial Intelligence, but it was not until the introduction of the digital computer that AI became a viable scientific discipline. By the end of the 1940's electronic digital computers could provide the amount of power and memory required by intelligent programs and it was now possible to implement formal reasoning systems on a computer and empirically test their sufficiency for exhibiting intelligence (Luger-1).

Most AI work conducted in the last 25 years has been in the development of programming languages and philosophies such as production systems, object-based systems and network representations. Much emphasis is currently being placed on developing systems that can exhibit the intellectual flexibility that approaches that of the intelligence of living organisms.

2.2 Overview of AI application Areas

As in most sciences, AI is divided up into a number of subdisciplines. These subdisciplines have been applied to different applications but nevertheless share an essential approach to problem solving. The following is an outline of the major application areas and thus contributions to AI as a whole.

2.2.1 Game Playing

A large amount of early AI research began with game playing such as checkers, chess and 16 puzzle. The games have properties that make them ideal for AI research topics. These include a well defined set of rules (ie a structured problem), board configurations that can be easily represented on a computer and an opponent whose moves cannot be reliably anticipated. Games can generate large search spaces requiring techniques called heuristics to explore all the alternatives available. Much of what is termed intelligence lies in the heuristics used by humans to solve problems (Luger-1).

2.2.2 Automated Reasoning and Theorem Proving

This field is often considered to be the oldest branch of AI. It is the most fruitful branch of this discipline, responsible for much of the early work in formalising search algorithms and developing formal representation languages such as predicate calculus and the logic programming language, Prolog (Shapiro-24). Despite many setbacks and inefficiencies involved in automated theorem proving, many important problems such as the design and verification of logic circuits, verification of the correctness of computer programs and the control of complex systems are well suited to this type of approach.

2.2.3 Expert Systems

A result of early work in AI was the realisation of the importance of domain-specific knowledge (Lindsay et al-25). For example a medical doctor is not effective at diagnosing illness solely because she possesses some inborn problem solving skills, she is effective because she knows a lot about medicine. Expert knowledge is a combination of a theoretical understanding of the problem and a collection of heuristic problem - solving rules that experience has shown to be effective in the domain. Expert systems are built by using the knowledge from a human expert and coding it in a way that the computer may apply to similar problems (Luger-1).

Some noted and well documented examples are Dendril, Mysin, Prospector, and Internist. It is important to note that most expert systems have been written for relatively specialised, well structured and clearly defined domains. Problems that use common sense are much more difficult to solve by these means. Expert systems are discussed further in chapter 4.

2.2.4 Natural Language Understanding and Semantic Modelling

One of the fundamental goals of AI is to create programs that are capable of understanding human language. The understanding of natural language is considered a fundamental aspect of human intelligence and the ability of programs to understand natural language would have a significant impact on the user-friendliness of computers (Winograd-26). There has been considerable effort to reach this goal, however only limited success has been achieved. The understanding of natural language goes beyond merely breaking down sentences into their individual parts. Real understanding depends on extensive background knowledge of the discourse and idioms used and the ability to apply general knowledge to resolve the omissions and ambiguities that are a normal part of speech (Luger-1).

2.2.5 Modelling Human Performance

The majority of current AI programs, although using human intelligence as a benchmark do not work in the same organisational patterns as the human mind. AI programs are generally engineered to solve problems without regard for their similarities to human mental architecture. A good example of this is Expert Systems, which although obtaining their knowledge from humans, do not really attempt to simulate human thinking processes. Furthermore, programs using non-human approaches are usually more successful than their human counterparts in problem solving. Nevertheless, systems that explicitly model aspects of human problem solving are an important area of research in AI. Human performance modelling has been used as a very powerful tool for building and testing theories of human cognition and this has given psychologists a new way of explaining the human mind (Luger-1).

2.2.6 Planning and Robotics

Planning is an important part of programming a robot to perform a sequence of actions while responding to changes in its environment. It attempts to co-ordinate a sequence of actions that will enable the robot to achieve some task, such as to move across an obstacle-filled room.

Planning is difficult due to the vast number of potential move sequences by the robot and the potential number of paths through the obstacles. The development of a program that can intelligently discover the best path without being overwhelmed by the number of possibilities requires sophisticated AI techniques (Luger-1).

2.2.7 Languages and Environments in AI

The evolution of programming languages and development environments has been an important by-product of continued research in AI. These programming environments include knowledge structuring techniques such as object orientated programming and expert system frameworks. As most AI applications are very large, programs which support modular development such as LISP and PROLOG have helped to develop program size and complexity. These programs include useful tools such as trace packages which allow programs to reconstruct the execution of algorithms and heuristically guided search. The structures of the languages which were developed for AI programming closely follow the theoretical principles of AI theory (Luger-1).

2.2.8 Machine Learning

This field of AI has remained a difficult area of research with little general success. This is a serious shortcoming with regard to the fact that the ability to learn is one of the most important components of intelligent behaviour. Expert systems themselves, although solving complex problems, do not remember the solution paths and given a repeated problem, merely perform the same set of computations over and over again.

Nevertheless, a few programs have been written that illustrate the potential for machine learning. One such program, the 'Automated Mathematician', was designed to discover mathematical laws (Lenat-27,28). This program developed new theorems by modifying its current knowledge base and used heuristics to pursue the most promising of a number of alternatives. These early learning programs point towards the existence of a set of general learning principles that will allow the construction of programs with the ability to learn in realistic domains.

2.3 Summary

Past research in AI has revealed the potential in this relatively young field. The primary goal of the study of AI is to find an effective way to understand and apply intelligent problem solving, planning and communication skills to a wider range of practical problems. Despite problems that have yet to be overcome there are several important features which are common to all branches of AI. These include the use of computers to do symbolic reasoning, reliance on heuristic search as an AI problem-solving technique, a desire to solve problems with inexact, missing or poorly defined information and the use of large amounts of domain specific knowledge in solving problems, eg, in expert systems.

CHAPTER 3

RULE-BASED EXPERT SYSTEMS

"Knowledge is Power"

Contents

- 3.1 Introduction**
- 3.2 Overview of 'Knowledge Engineering'**
- 3.3 The Difference Between Expert Systems and Conventional Computer Programmes**
- 3.4 The Structure of Expert Systems**
 - 3.4.1 Knowledge Base
 - 3.4.2 Context
 - 3.4.3 Inference Mechanism
 - 3.4.4 Explanation Facility
 - 3.4.5 Knowledge Acquisition Facility
 - 3.4.6 User Interface
- 3.5 Problem Solving Using Expert Systems**
 - 3.5.1 Types of Knowledge representation
 - 3.5.1.1 Logic-based Representation
 - 3.5.1.2 Rule-based Representation
 - 3.5.1.3 Network-based Representation
 - 3.5.2 Approaches to Problem Solving Strategy
 - 3.5.2.1 Derivation
 - 3.5.2.2 Formulation
 - 3.5.3 Control Strategies
 - 3.5.3.1 Forward Chaining
 - 3.5.3.2 Backward Chaining
 - 3.5.3.3 Mixed Chaining
 - 3.5.3.4 Other Control Strategies
- 3.6 Dealing with Uncertainty in Expert Systems**
- 3.7 Problems Relating to Expert System Technology**
- 3.8 Selecting a Problem for Expert System Development**
- 3.9 Testing and Evaluation of Expert System Performance**
- 3.10 The SYNAPSE Expert System Shell**
 - 3.10.1 Overview
 - 3.10.2 Inference Networks and Rules
 - 3.10.3 Synapse mathematics and Uncertainty

RULE-BASED EXPERT SYSTEMS

3.1 Introduction

An expert system is a knowledge based program that provides expert "quality" solutions to problems in a specific domain. Generally, its knowledge is extracted from human experts in the domain and it attempts to emulate their methodology and performance. As with skilled humans, expert systems tend to be specialists, focusing on a narrow set of problems. Unlike humans current expert systems cannot learn from their own experience, their knowledge must be extracted from humans and encoded in a formal language. This is the major task facing expert system builders. Expert systems neither copy the structure of the human mind nor are mechanisms for general intelligence. They are practical programs that use strategies developed by humans to solve specific classes of problems.

The general characteristics of expert systems are:

- 1) They are open to inspection, both in presenting intermediate steps and answering questions about the solution process. This is desirable in that for any human expert to accept a recommendation from the computer, they must be satisfied that the solution is correct. These explanations help people relate the advice to their existing understanding of the domain and apply it in a more confident and flexible manner.
- 2) They are easily modified, both in adding and deleting skills from the knowledge base. The exploratory nature of AI and expert system programming requires that programs be easily tested and changed. Expert system designers have commented that easy modification of the knowledge base is a major factor in producing a successful program.(McDermott,2)
- 3) They use heuristic problem solving methods to obtain a solution. This use of 'imperfect knowledge', informal 'tricks of the trade' and 'rules of thumb' by expert system designers is often more important than the standard theory presented in textbooks. Sometimes these rules augment theoretical knowledge or are simply shortcuts that seem unrelated to the theory but have been shown to work.

The heuristic nature of expert problem solving knowledge creates problems in evaluation of program performance. As heuristic methods occasionally fail, the question is how often the program must be correct in order to be accepted. The best way to evaluate a program is to compare its results to those obtained by human experts in the same area.

3.2 Overview of 'Knowledge Engineering'

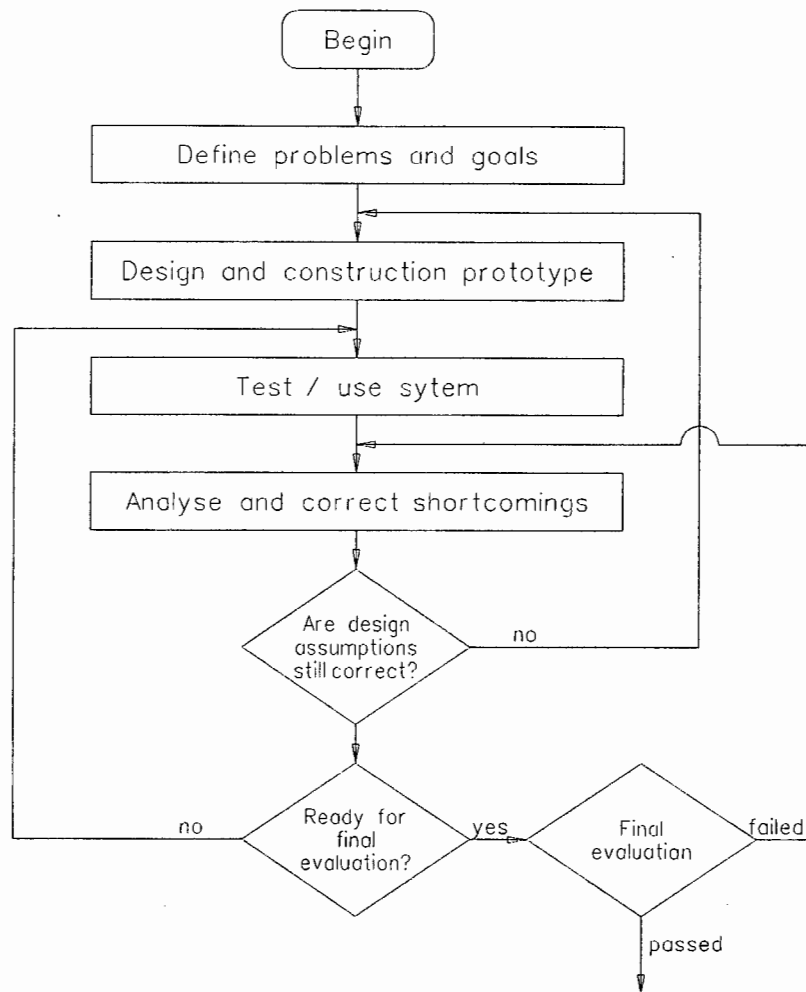
The primary people involved in the development of an expert system are the knowledge engineer, the domain expert and the end user. The knowledge engineer is the AI language and representation expert. His main task is to select the software and hardware tools for the project, extract the necessary knowledge from the domain expert and capture that knowledge in a correct and efficient knowledge base. The knowledge engineer may initially know very little about the application domain.

The domain expert provides the knowledge of the problem. The domain expert is generally somebody who has worked in the domain area and understands its problem solving techniques, including the use of shortcuts, handling imprecise data, evaluating partial solutions, and all the other skills that mark the person as an expert. The end user, as in most applications determines the major design constraints. Several aspects relating to the end user need to be taken into account. These are; the skills and needs of the end user, the level of explanation required, the user interface, whether the user can provide the information required and any environmental restrictions to the program's use.

The development cycle of expert system programming follows a nontraditional method starting by early prototyping and then stages of incremental code revision. The cycle would typically begin by the knowledge engineer obtaining an overview of the problem and going through several problem-solving sessions with the domain expert. Once the knowledge engineer has selected a shell, a prototype is then built. This prototype is then tested together with the domain expert by giving it problems to solve. The knowledge base is then corrected in order to solve any shortcomings and further knowledge is then added. In this way the knowledge base in a sense 'grows' as the systems are built by progressive approximations, with the mistakes leading to corrections and/or further additions to the knowledge base. A typical development cycle is shown in Figure 3.01

A major feature of expert systems is that the system need never be considered as finished. A knowledge base will always have limitations for which additions can be added later. A typical example of this is the XCON expert system which had 500 rules and could configure a VAX 780. By three years later it had 4000 rules and could configure the entire company's product line of computers. It was also noted that up to 50% of the rules were changed each year to keep up with changes to the product line. (Soloway et al.,3).

One of the most important aspects of the evolution of expert systems has been the drastic reduction in average development time for expert system applications. This has been brought about by the emergence of expert system shells.



Exploratory development cycle for rule-based expert systems

Figure 3.01

A major problem that has developed in building of expert system applications is what is known as the 'engineering bottleneck'.

This bottleneck is caused by the fact that building an expert system requires a substantial investment in time and effort on the part of the domain expert and knowledge engineer. As both individuals are generally highly paid individuals, the corresponding loss of productivity together with the logistics of getting these two people together and the complexity of the program, contributes to the expense of expert system development. The obvious solution is to automate the process as much as possible. This has been achieved to a certain degree by shells, however the programmer is still required to understand the methodologies of knowledge representation and search. A further step would be to introduce intelligent knowledge base editors, eliminating the need for knowledge engineers. The domain expert could thus develop the knowledge base entirely independently, allowing the software to handle aspects of knowledge representation and organisation. A prototype editor called Teirasias (Luger,1), has already been developed at Stanford University.

3.3 The Difference Between Expert Systems and Conventional Computer Programmes

The most commonly asked question is what makes an expert system different from a conventional program. The major differences are summed up in the following table :

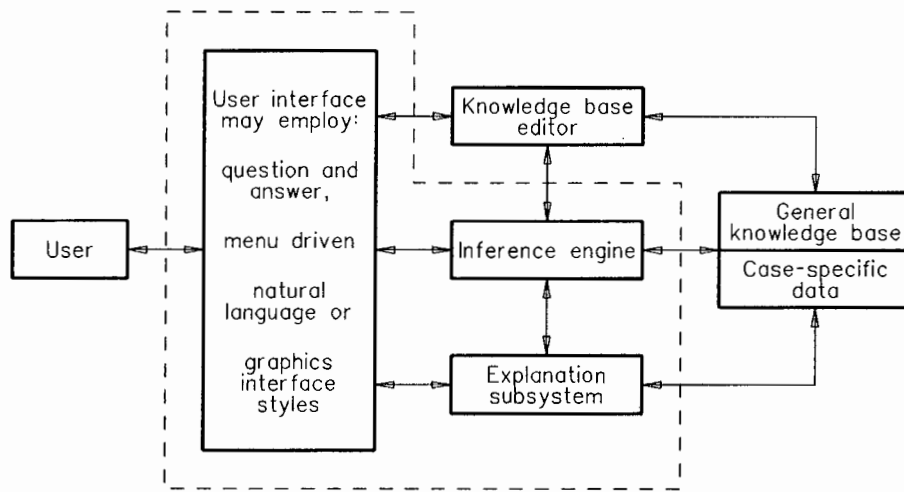
<u>Conventional Programs</u>	<u>Expert Systems</u>
1) Representation and use of data	Representation and use of knowledge
2) Knowledge and control are integrated.	Knowledge and control separated.
3) Algorithmic (repetitive) process.	Heuristic (inferential) process.
4) Effective manipulation of large data bases	Effective manipulation of large knowledge bases.
5) Programmer must ensure uniqueness and completeness	Knowledge engineer inevitably relaxes uniqueness and completeness restraint.
6) Midrun explanation impossible.	Midrun explanation desirable and achievable.
7) Orientated towards numerical processing	Orientated towards symbolic processing.

3.4 The Structure of Expert Systems

The structure of expert systems cannot be clearly defined owing to the many tools and languages in which they are developed. The basic structures of most expert systems can however be generalised into several categories. These are Knowledge Base, Context, Interference mechanism, Explanation facility and Knowledge acquisition and user interface. The relationships between these categories are shown in figure 3.02.

3.4.1 Knowledge Base

The knowledge base is the component of an expert system which contains the facts and heuristics associated with the domain for which that particular expert system has been developed.



Structure of an Expert System

Figure 3.02

The facts are usually represented as declarative knowledge and heuristics take the form of rules. As an example, a structural design knowledge base would contain facts that can be grouped according to the physical objects used to describe a design. An example of an object would be a beam. This beam could be represented in the knowledge base as a structural component with attributes including width, span, depth and loads. The rules or heuristics relating to the structural design of that beam would for example be: if beam span is $< 10\text{m}$ and material is concrete and is simply supported then span / effective depth ratio is equal to 16 (Clause 4.3.6.2.1. SABS 0100,4). It is important to note that this heuristic contains information about structural design with no explicit instructions as to how or when the knowledge should be used.

Other important aspects of the knowledge base is that it should be easily modifiable. This modification is important in most engineering domains since knowledge is continually changing and expanding. More advanced expert system environments provide higher level representation schemes than procedural code, such as rules or frames in order to achieve this goal.

3.4.2 Context

The context is the component of the expert system that contains the information about the problem currently being solved. It initially contains information that defines the parameters of the problem and continually expands, storing the results of the reasoning process as the expert system solves the problem. Once completed the context would then contain all the intermediate results of the problem solving process as well as the final solution.

3.4.3 Inference Mechanism

The inference mechanism is the component of the expert system that contains the control information. Essentially the Inference mechanism uses the knowledge base to modify and expand the context. There are different levels at which the inference mechanism controls the reasoning process. Typically the more specific the inference mechanism, the less control information there is in the knowledge base. Depending on how the particular problem has been structured, the inference mechanism can reason through the problem either forwards or backwards or both. This is known as forwards, backwards or mixed chaining and is dealt with in more detail in section 3.5.3.

3.4.4 Explanation Facility

This facility in expert systems differs from a mere trace of execution to the ability to respond to questions about the reasoning process used to develop a solution. This is not a concept unique to expert systems as most well written computer programs contain this feature. An expert system however, provides more than just a passive trace in that it responds to questions about specific aspects of the problem solution. For example a user may ask a structural design expert system 'How do you know the column is strong enough?' and the expert system will provide the decisions it made in order to reach that conclusion.

3.4.5 Knowledge Acquisition Facility

This facility of an expert system is that component that enables the entering of knowledge into the knowledge base. In its most basic form the facility acts as an editor and the knowledge is entered in a format that the expert system can use directly. On a more sophisticated level the knowledge acquisitions facility interacts with the 'IF' of the expert system and can actively help the domain expert or knowledge engineer in developing the knowledge base. An example of this is Teirasias (Luger,1).

The range of possibilities of a knowledge acquisition facility are thus shown in the type of editor used for creating or modifying the knowledge base. At the one end of the range a conventional screen editor is used to create or modify a file of rules. At the other, the editor itself is an expert system, knowledgeable itself about problem solving using the inference mechanism provided.

3.4.6 User Interface

The user interface of an expert system goes beyond traditional user interfaces. It is typically highly interactive, usually with 'help' facility and contains an explanation facility for illustrating or depicting the inference or reasoning process used.

3.5 Problem Solving Using Expert Systems

In general, problem solving involves the search for a solution or set of solutions. The search starts at an initial state of known facts and conditions and ends at a goal state. The solution path consists of all states that lead from the initial state to the goal state. The expert system's search for a solution is called the problem solving strategy. There are two principal methods of problem solving, the weak methods and the strong methods.

Weak methods are problem solving strategies that are domain independent and can lead to combinational explosions due to 'lack of focus'. Expert systems are considered strong problem solvers since the solution strategy uses domain knowledge.

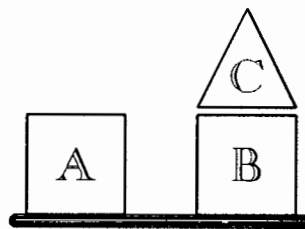
3.5.1. Types of Knowledge Representation

The way in which knowledge is represented within an expert system is of primary importance. Knowledge representation involves trying to find the best way to represent knowledge as data structures so that it can be conveniently and efficiently accessed for a problem solving task. The representation of uncertainty is an advanced aspect of knowledge representation that is dealt with in section 3.6. In addition to the representation of static domain knowledge in a knowledge base it is equally important to represent knowledge about problem solving strategies or control strategies dealt with in section 3.5.3. Although these three aspects of knowledge representation, uncertainty and knowledge representation and problem solving strategies are dealt with separately, the relationship in which these aspects are combined together in a knowledge base remain tightly bound. The main types of knowledge representation are logic-based representation, rules and network-based representation.

3.5.1.1 Logic-Based Representation.

A logic based representation is one in which knowledge about the world is represented as logical statements typically in the form of first order - predicate logic or a variation of it. Luger (1) describes this logic as a means of capturing and reasoning about the qualitative aspects of a problem. A simple example illustrates the concept further: figure 3.03. One way to represent the blocks would be via a Cartesian - co-ordinate system using x,y for vertices of blocks. Although this describes the block's world it fails to record the properties of each block and the relationships between blocks. Using predicate logic, the block world would be described in the following way:

clear (c)	on (c,b)
clear (a)	cube (b)
ontable (a)	cube (a)
ontable (b)	pyramid (c)



A blocks world

Figure 3.03

The first word is a predicate denoting some property or relationship among its arguments and the arguments are symbols denoting the block in the domain. Predicate logic provides AI programmes with a well defined language for describing and reasoning about qualitative aspects of a system. An example of such a language is PROLOG. Usually this form of representation is coupled with an inference procedure based on theorem proving. The main disadvantages are the difficulty of dealing with imprecision and uncertainty in reasoning.

3.5.1.2. Rule-Based Representation

The main concept of rule-based representation is that the domain knowledge is represented in modular rules in the general form: IF <SITUATION OR CONDITION> THEN <ACTION>.

The condition portion is referred to as the Left Hand Side and the action pattern is called the Right Hand Side. The control aspects are separated in a module called an executive or inference engine.

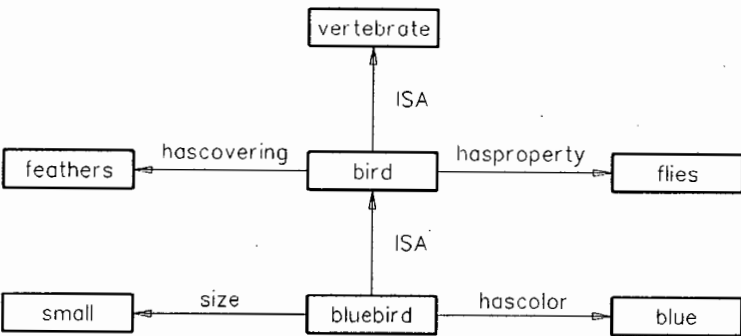
Examples of successful systems are MYCIN, DENDRAL and XCON. The advantages of Rule-based systems is that of homogeneity and simplicity permitting easy modification while the major disadvantage is the inability to describe general relations between pieces of knowledge.

3.5.1.3. Network-Based Representation

Semantic networks explicitly represent the connectivity and hierarchy between pieces of knowledge and also simplify certain relationships such as inheritance. Inference networks use semantic networks to organise rules. Typically in these networks, nodes represent information in the knowledge base and links represent connections or relations between the pieces of information.

Many AI problems use a large amount of highly structured and interrelated knowledge. For example it is not sufficient to describe a car by listing its components, a valid description must also describe the ways in which these parts are assembled and the interactions between them.

For example, in figure 3.04 a bluebird can be described as a small blue coloured bird or a small blue coloured flying feathered vertebrate. In figure 3.04 certain links such as ISA links indicate class memberships and properties attached to this class to be inherited by all members of the class.



Semantic network description of a bluebird

Figure 3.04

3.5.2. Approaches.

There are two fundamental approaches to problem solving currently used in expert systems; the derivation and the formulation approach. The derivation approach starts at a known state and uses deductive reasoning to arrive at a known solution. A formulation approach uses information about a state to generate more information to form higher level solutions. The nature of problem solving is bounded by these two approaches and expert system applications use either one or the other.

3.5.2.1. Derivation

The derivation approach essentially selects a solution to the problem at hand from a list of pre-defined solutions in the knowledge base of the expert system. This implies that all the solutions to the class of problem the expert system will have to solve are known and can be justified by a certain set of circumstances. The implementation of this approach is done by creating an inference network representing the relationships between the pre-defined solutions and the input data specifying a given problem (discussed in section 3.10.2). Examples of engineering problems that use the derivation approach are :

- Diagnosis, for example mechanical failure of a system, where the pre-defined solutions are the possible known malfunctions of the system being diagnosed.
- Classification, such as interpreting field data, where the pre-defined solutions are the set of applicable classifications.

The control strategies that are used in the implementation of the derivation approach are forward chaining, backward chaining and mixed chaining. These are discussed in sections 3.5.3.1. to 3.5.3.3.

3.5.2.2. Formulation.

The formulation approach creates a solution from the eligible solution components stored in the knowledge base. This approach is used when all the possible solutions cannot be counted or stored in the knowledge base. The implementation of this approach involves identification of the parts of the solution and a strategy and heuristics for combining these parts. The parts of the solution are generally arranged into categories representing a hierarchy of decisions or subsystems. Engineering examples of formulation approach are :

- Design, such as preliminary structural design, where subsystems are integrated together.
- Planning, such as construction sequence, where activities are selected and ordered.

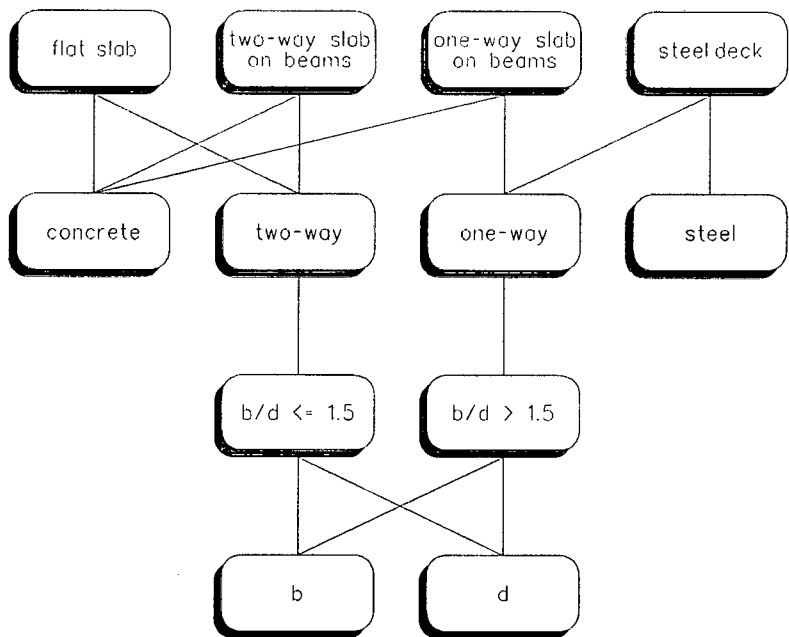
The control strategies appropriate for a formulation approach are problem reduction, plan-generate-test and agenda control. These are discussed briefly in section 3.5.3.4.

3.5.3 Control Strategies

There are many different types of problem solving strategies that are used by expert systems. Most expert systems provide general strategies that are guided by the knowledge in the knowledge base.

3.5.3.1. Forward Chaining

Forward chaining works from an initial state of known facts to a conclusion or goal state. This strategy is also called bottom up, data-driven or antecedent-driven and works best where there are many hypotheses or solutions and little input data. The main disadvantage is that it is sometimes wasteful to input all possible facts for all conditions as in many circumstances these are not all known.



Floor System Design Inference Network

Figure 3.05

In figure 3.05 a forward chaining strategy will calculate the values of b,d for either concrete or steel. This will be used to determine which of the supporting systems is best.

3.5.3.2 Backward Chaining

A backward-chaining strategy starts by assuming a hypothesis or goal state and reasons back to known data or facts to probe or disprove the assumed hypothesis. This strategy is also known as top-down, goal-driven or hypothesis drive strategies. There may be more than one hypothesis in the knowledge base depending on how it is set up. For the example in figure 3.05 a backward chaining strategy would assume a support system and determine whether it is appropriate in a given situation given the input values such as b,d in that situation.

3.5.3.3. Mixed Chaining

This strategy is used when a system combines forward and backward chaining. The system would typically start with an initial state of known facts and assign a probability to the goal states. The system would then try to support the goal state with the highest probability by setting up subgoals and requesting additional facts if required. The advantage of this strategy is that only the available data regarding the problem at hand is supplied to the system and if the initial goal is disproved the next assumption is made according to the current facts. In figure 3.05 this strategy would be implemented as follows: the subset facts b,d could be provided. The expert system would then identify the most likely solutions using the relationship between b and d and pursue only these, requesting additional information regarding material as necessary.

3.5.3.4. Other Control Strategies

For completeness it is necessary to mention that other strategies exist. These include problem reduction, plan-generate-test and agenda control. The problem reduction strategy involves factoring problems into smaller problems. The problems being represented as an AND-OR-TREE similar to Boolean algebra. The plan-generate-test strategy generates all possible solutions from components in the knowledge base and tests each solution until it finds one that satisfies the goal specification. The agenda control strategy involves assigning a priority rating to each task in the agenda. The task with the highest priority is performed first and a list of justifications and a method of determining the priority rating is associated with each task.

3.6. Dealing With Uncertainty in Expert Systems.

As stated before the representation of uncertainty in knowledge bases is closely bound to the representation of knowledge itself. Knowledge is seldom ever certain and often expert systems must attempt to draw correct conclusions from uncertain facts or evidence using unsound inference rules. This is an integrated aspect of everyday human life. For example doctors treat patients correctly by responding to ambiguous symptoms. The reasons for this ambiguity can be better understood using an example of an expert system for a car that will not start.

if:

the engine does not turn over, and
the lights do not come on

then:

the problem is with the battery or cables.

The rule is heuristic in nature and seems to represent a logical implication but it is not necessarily correct. There may be a problem with the starter motor, coil, distribution, ignition switch, etc. It is however interesting to note that the converse of this rule is correct. This car rule is an example of abductive reasoning which means the conclusion is not necessarily true for every interpretation in which the premises are true. In the car example, failure of the battery or cables is not always associated with the failure of the car's lights and starter but it almost always is. For this reason a measure of confidence is attached to the conclusions.

There are several schools of thought as to the method to be used to accommodate uncertainty in the reasoning process. The classic approach to quantify uncertainty is the Bayesian probability theory. This theory, assuming the pure random distribution of events, provides a means of calculating more complex probabilities based on previously known results, for example how playing cards may be distributed to players.

Without discussing further how this is actually implemented it is sufficient to state that the Bayesian approach offers the benefit of a well founded and statistically correct handling of uncertainty. It is however felt that humans do not use this approach in successfully solving problems. To solve this problem a heuristic approach to the management of uncertainty was developed. This included Certainty theory, possibility theory based on fuzzy sets and the Dempster-Shufer theory of evidence. Detailed descriptions of these theories can be found in Luger(1). Each expert system language or environment handles this problem of uncertainty with different theories and methods. The specific way in which the chosen expert system used in this thesis handles uncertainty is dealt with in section 3.10.3.

3.7. Problems Relating to Expert System Technology

Despite the success and promise which expert systems have shown in various fields, the technology itself contains many deficiencies. These include the following:

- 1) Difficulty in capturing 'deep' knowledge of the domain. For example MYCIN, an expert system developed at Stanford University for medical diagnosis does not contain any knowledge of human physiology. It does not know what blood does or the function of the spinal cord.
- 2) Lack of robustness and flexibility. If a human is confronted with a problem that he has not seen before, he/she resorts to solving it by exploring first principles or similar problems in other related fields. An expert system does not have this ability and does not 'degrade gracefully' when confronted with situations that do not fit the knowledge in the knowledge base.
- 3) Expert systems lack the ability to provide deep explanations. Due to the fact that expert systems lack 'deep' knowledge their explanations are restricted to a description of the steps taken to reach the solution. They cannot explain why a certain approach to solving the problem was taken.
- 4) An expert system cannot learn. This limitation is serious in view of the fact that it is one of the most important aspects of intelligence. Even though an expert system performs extensive computations to solve a problem, given the problem a second time it will not remember the solution and will merely perform the same sequence of computations all over again.
- 5) The larger and more complex an expert system becomes, the more difficult it is to modify the knowledge base. This could be addressed by intelligent editors modifying the code interactively with knowledge engineers.

3.8 Selecting a Problem for Expert System Development

The selection of problems that are capable of being solved using expert systems is of paramount importance. As stated before expert systems tend to require vast amounts of time and resources and thus attempts to use expert systems to solve problems that are too complex, poorly understood or unsuited to expert system technology will result in costly and embarrassing failures. The following guidelines should be used in determining whether problems are successfully solvable using expert systems :

- 1) The need for solution justifies the cost and effort of building an expert system. This simply refers to whether the return on investment is sufficient to justify the development cost.
- 2) Human expertise is not available in all situations where it is needed. For example the lack of medical expertise in remote rural areas could justify medical diagnostic expert systems being used. This could potentially solve the problem of the reluctance of professional people to visit these areas.
- 3) The problem may be solved using symbolic reasoning techniques. Expert systems are generally restricted to problems that humans can solve through symbolic reasoning.
- 4) The problem is well structured and does not require common sense reasoning. Highly technical fields are generally well studied and have well defined domains. The amount of knowledge required to solve such problems is small in comparison to the amount of knowledge used by human beings in common sense reasoning.
- 5) The problem may not be solved using traditional computing methods. As expert systems rely on heuristic problem solving they are unlikely to outperform conventional algorithmic solutions. The use of expert systems to solve traditional computational problems is merely 'reinventing the wheel'.
- 6) Co-operative and articulate experts exist. Expert systems are generally developed from experience and judgment of human experts in the domain. These people must be willing and able to share their knowledge and believe that the project is both practical and beneficial.
- 7) The problem is of proper size and scope. For example a program that attempts to capture all the expertise of a medical doctor is not feasible. The dividing of a large problem into smaller, independent problems may be a better approach.

Expert systems have been developed to solve a wide range of problem types. These are best categorised into the following applications :

- 1) Interpretation - forming high level conclusions or descriptions from collections of raw data.
- 2) Prediction - projecting probable consequences of given situations.
- 3) Diagnosis - determining the cause of malfunctions in complex situations based on observable symptoms.

- 4) Design - determining a configuration of system components that meet certain performance goals while satisfying a set of constraints.
- 5) Planning - devising a sequence of actions that will achieve a set of goals given certain starting conditions.
- 6) Monitoring - comparing the observed behaviour of a system to its expected behaviour.
- 7) Debugging and repair - prescribing and implementing remedies for manufacture.
- 8) Instruction - detecting and correcting deficiencies in student's understanding of a subject domain.
- 9) Control - governing the behaviour of a complex environment.

3.9. Testing and Evaluation of Expert System Performance.

One of the most difficult questions that builders of expert systems have to address is 'at what stage of development does an expert system's advice reach expert quality?'. In light of the fact that it is unlikely that a knowledge base can ever be proved to be 100% correct. This problem becomes all the more important when lives are at stake and the legal consequences of incorrect decisions are calculated. Many expert systems are currently evaluated using the Turing Test.

The Turing test measures the performance of an allegedly intelligent machine against that of a human expert in that domain, arguably the best and only standard for intelligent behaviour. Turing (5) presents his test in the following way: the test (also called the imitation game) places a machine and human counterpart in rooms apart from a second human called the interrogator. The interrogator communicates with both of them through a device such as a terminal but does not know the identity of either. The interrogator is asked to distinguish the computer from the human expert solely on the basis of their answers to questions. If the interrogator is unable to reliably distinguish the machine from the human expert the machine may be assumed to be intelligent in that field/domain.

The important features of this test are:

- o It gives an objective notion of intelligence.
- o It prevents people from being sidetracked by currently unanswerable questions, ie whether or not the machine is conscious of its actions.

- It eliminates any bias in favour of living organism by forcing the interrogator to focus solely on the context of the answers to the questions.

This methodology has become an essential tool in the verification and testing of modern expert systems.

This test was used by the developers of MYCIN, a medical diagnostic expert system developed at Stanford Medical School. In the test selected, case histories of meningitis were re-diagnosed using MYCIN and eight practitioners from the Medical School including one research fellow in infectious diseases. The results of the test were evaluated and points awarded according to the correctness of the recommended therapy. In the final result MYCIN was awarded 55 out of a possible 80 points and the top scoring doctor was awarded 54 out of 80. This showed that MYCIN performed at least as well as the Stanford experts. By evaluating other aspects such as the doses of drugs prescribed, MYCIN prescribed fewer drugs than any of the human experts. It must however be noted that there was a large range of disagreement between the Stanford experts regarding therapies to be used, thus highlighting the extent to which human expertise is still largely heuristic in nature.

The encouraging result of the MYCIN Turing Test did not mean however that MYCIN was ready to take on patients. MYCIN is still very much hindered by problems similar to those described in section 3.7 such as time of responses, lack of deeper understanding, lack of flexibility and the inability to resort to first principles or common sense reasoning.

3.10 The SYNAPSE Expert System Shell

3.10.1. Overview.

The SYNAPSE expert system shell was selected for use in this thesis for several reasons. These include relative clarity of knowledge base layout, graphical capability, the ability to solve various types of problems and the fact that it was relatively cheap, having been developed locally, and the writer's familiarity with it.

SYNAPSE is a general purpose expert system shell which is designed to be used as a stand-alone and integrated decision system. SYNAPSE decision making model represents the decision making process as a network of basic elements called an inference network. These elements include goals (decisions), subgoals (supporting arguments) and questions (details). The network is expressed in SYNAPSE language as a number of rules and each network or group of rules is called a rulebase.

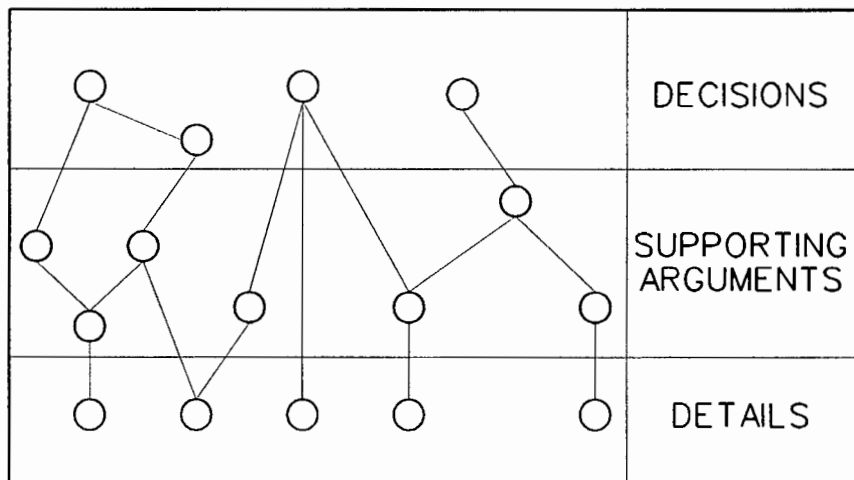
An important ability of SYNAPSE is that it can be integrated with other application programmes.

3.10.2. Inference Networks and Rules.

The SYNAPSE inference networks are divided into various levels. These levels can be loosely described as

- o Decisions
- o Supporting arguments
- o Details

There may be any number of decisions, supporting arguments and details. Each one of these is called a node. Each node can be linked to any other node to form a network. A generalised example of this network is shown in figure 3.06.

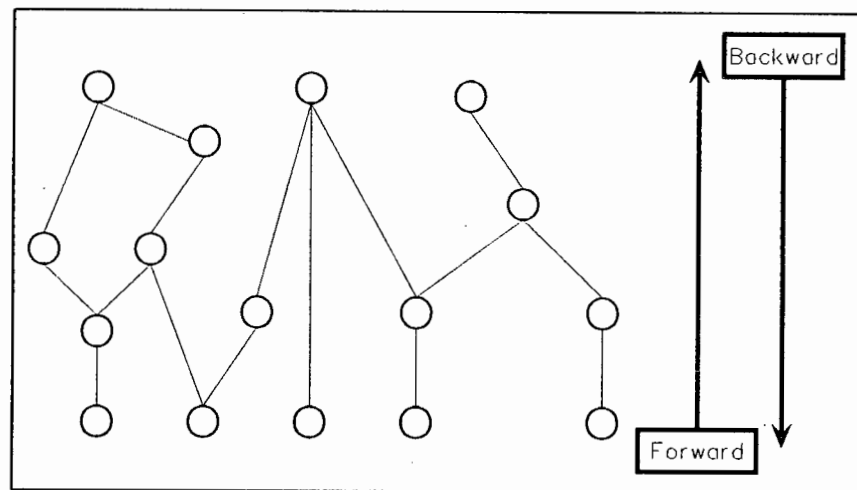


Typical Structure of a SYNAPSE Inference Network

Figure 3.06

The knowledge engineer typically enters the network into a file using a text editor, in the form of SYNAPSE rules. Once entered, the rulebase is then compiled by the SYNAPSE compiler and is ready to run. SYNAPSE uses a mixed chaining control strategy. Therefore unless otherwise instructed, SYNAPSE controls the order in which the goals are analysed and the order in which the questions are asked. The actual control strategy used by SYNAPSE is in two phases. The first is a forward chaining strategy in which it attempts to find the next goal most likely to be established.

The second phase is the backward chaining strategy in which the system attempts to trace the information that will lead to the establishment of the goal. See figure 3.07.



Forward and Backward Chaining Strategies

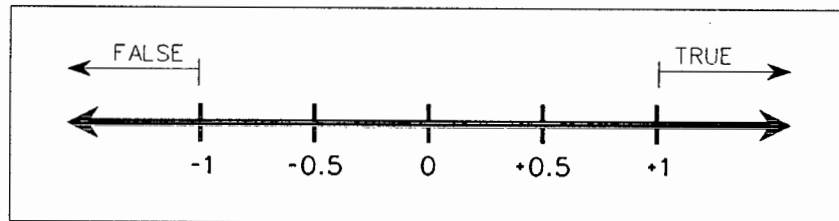
Figure 3.07

In the network each decision, supporting argument or question is contained as a node. The relationships between the nodes (elements) are defined as arcs of the network.

3.10.3. SYNAPSE Mathematics and Uncertain Reasoning.

SYNAPSE contains its own mathematical simulation of uncertain reasoning as described in Appendix 1. As discussed in this section, worldly events and statements can seldom take on the values of true or false. A means of describing and modelling these type of events is thus required. SYNAPSE Mathematics works in the following way.

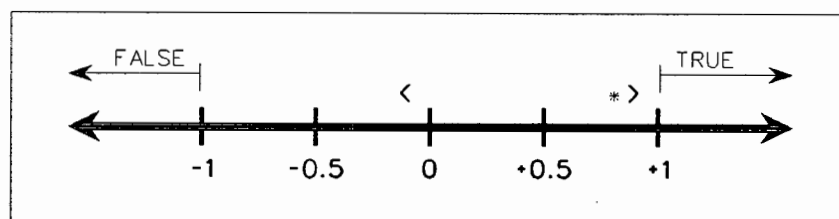
SYNAPSE starts with an assumption that any number less than -1 is false and any number larger than +1 is true. Any question, subgoal or goal rule has a value indicating its position on this number line, see figure 3.08.



TRUE and FALSE positions

Figure 3.08

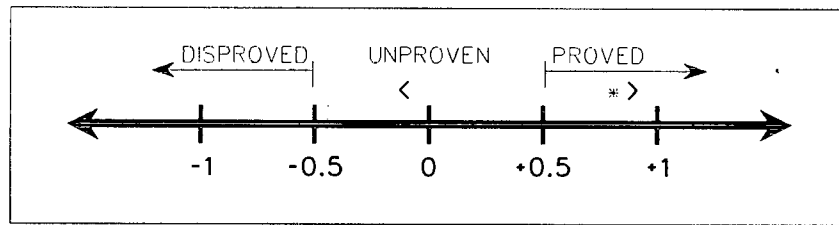
Any element can be inspected to see whether it is true or false. Any number larger than +1 is cut back to +1 and any number below -1 is cut back to -1. In fact, each element (rule) has three values, the worst case value, the expected value and the best case value. The worst case value is that value which the element would have if all unknown but related issues had the worst possible outcome. The best case value is that value which the element would have if all unknown but related issues had the best possible outcomes. The expected value is the value the element would have if all unknown but related issues had neutral outcomes. The symbols representing each value are $<$, $*$, $>$ respectively. These values are known as real value logic (RVL) values and are used as shown in figure 3.09 (For the theory behind the derivation of RVL see Appendix 1). The process by which each element attains these values will be dealt with shortly.



The number line showing an RVL range

Figure 3.09

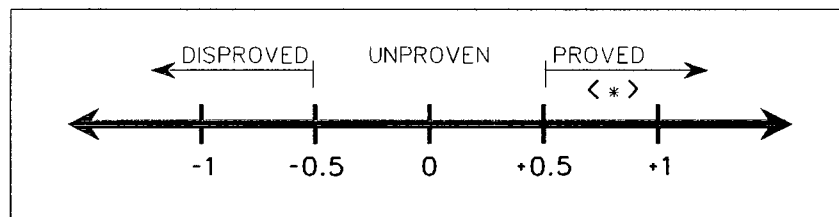
SYNAPSE furthermore defines 'as good as false' or disproved as the range between -1 and -0.5 and 'as good as true' or proved as the range between +0.5 and +1. The range of values between -0.5 and +0.5 is defined as unproved. This is clearly shown in figure 3.10.



The number line showing 'as good as' values

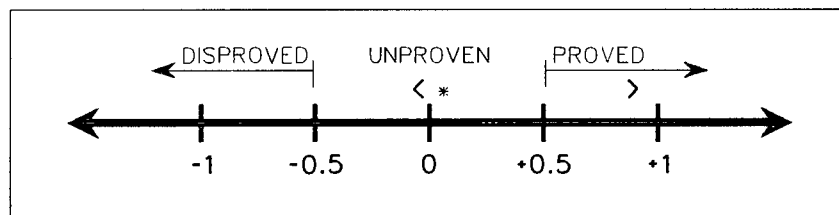
Figure 3.10

The next two concepts are resolved and unresolved. An element is said to be resolved when all three of its RVL values fit into one of the ranges, disproved, unproved or proved. This is because all RVL values will always be in this range and there is no longer any need to investigate further. If the RVL ranges straddles the disproved, unproved and proved range it is said to be unresolved, which means that the issues relating to that element need to be investigated further. See figure 3.11 and figure 3.12.



Node resolved as PROVED

Figure 3.11



Node resolved as UNRESOLVED

Figure 3.12

The three truth values are passed from detail to supporting arguments and decisions through the use of effects which are tied to the arcs. For example when a question in SYNAPSE is answered in a positive sense (ie truth value > 0) the system processes the value based on the true effect defined for that arc.

Conversely when a question is answered in a negative sense (truth value < 0) the system processes the value based on the false effect.

When combining many details or pieces of evidence into a supporting argument or decision the truth value of each component is added. When considering an uncertain response as a piece of evidence it must first be decided whether a response involves a true effect or false effect and the relevant effect is then multiplied by the absolute value of the truth value.

The formula used is depicted as follows:

$$RVL\ V|E = RVL\ V + (| Truth\ e | \times Effect\ v|e)$$

where:

Truth e	The truth value of e. This is either the response to a question or the truth value calculated by SYNAPSE from other issues that make up e.
Effect v e	The TRUE EFFECT or FALSE EFFECT of the arc that links e to v. If Truth e is greater than or equal to zero, then we use the TRUE EFFECT. If Truth e is less than zero we use the FALSE EFFECT.
Truth e	Indicates that we use the absolute value of Truth e. The sign and direction of the calculation is derived from the sign of the Effect v e.

This calculation is made when updating the expected value of the node. This is followed by the range calculation which calculates the worst case by using all the negative effects of unanswered questions and conversely for the best case.

For an example of these ideas a brief hypothetical example is presented below. Unnecessary detail has been stripped from the rule for the purposes of this example.

GOAL A

< = B	-0.65	+ 0.38
< = C	+ 0.45	-0.43
< = D	-0,52	+ 1.27

QUESTION B

QUESTION D

QUESTION C

In this example it can be seen that in the arc that links B to A the FALSE EFFECT is -0.65. This means that if B is false than A will be -0.65 (as good as false). Simalily for the arc that links D to A +1.27 is the TRUE EFFECT so that if D is true then A will be +1.27 (true). It is also possible for there to be a positive FALSE EFFECT or a negative TRUE EFFECT as shown in the arc that links C to A. Before going any further the system can calculate the lower, expected and upper RVL values. These values are -1.6, 0 and +1.27. These numbers are calculated on the assumption that the rule has all the negative effects assigned to the arcs for the worst case, all zeros applied to all the arcs for the expected case and all positive effects assigned to the arcs for the best case.

Assuming further that an answer of 0.0 is inputted for question C (ie C is 'maybe') then the lower, expected and upper RVL values are now calulated as -1.17, -0.65 and +0.62. This is because the truth value of B will now remain -1.0 and the truth value for C will remain 0.0.

Finally answering 0.52 for question D (in the area between 'maybe' and 'yes') the three values of A will now converge to 0.01, 0.01 and 0.01 (ie $[1.0 \times -0.65] + [0] + [0.52 \times 1.27] = 0.01$).

One of the fundamental principals of SYNAPSE mathematics is that before any values are passed up the inference network for further calculations they are trimmed back to fit in the range -1.0 to +1.0. For example if the lower, expected and upper range was -0.34, 0.13 and 1.87 it would be passed to the next level of the inference network as -0.34, 0.13 and 1.0.

ENVIRONMENTAL IMPACT ANALYSIS AND RULEBASE DEVELOPMENT

*"A live mind can see a window onto
a world of possibilities"*

Douglas R. Hofstadter (US Writer)

Contents

- 4.1 Introduction**
- 4.2 The Principals Behind IEM**
- 4.3 The Integrated Environmental Management Procedure**
 - 4.3.1 Stage 1: Plan and Access
 - 4.3.2 Stage 2: Decision
 - 4.3.3 Stage 3: Implementation
- 4.4 The Role of the Engineer in IEM**
- 4.5 Impact Assessment Methodology**
- 4.6 The Proposed Methodology**
- 4.7 Rulebase Design Criteria**
- 4.8 The Knowledge Acquisition Procedure**
- 4.9 Environmental Characteristics in Project Evaluation**
 - 4.9.1 Physical
 - 4.9.2 Ecological
 - 4.9.3 Potential Land Use/Landscape Character
 - 4.9.4 Cultural Resources
 - 4.9.5 Infrastructure
 - 4.9.6 Social and Community Services
 - 4.9.7 Pollution
 - 4.9.8 Risk and Hazard
 - 4.9.9 Health and Safety
 - 4.9.10 Cumulative and Synergistic Effects
- 4.10 Project Evaluation**

4.1 Introduction

The concept of integrated environmental management (IEM) is designed to ensure that the environmental consequences of a proposed development or project are properly understood and catered for while the project is still in the planning phase. The term 'environment' is not limited to the 'natural environment' but has a more comprehensive meaning encompassing the social or cultural environment, the economic environment and a host of other areas of concern. Thus the environment includes not only the biological or physical aspects of human, animal and plant life, but also the socio-economic and cultural conditions that influence the life of a community.

The purpose of IEM is to identify and alleviate any negative influences and to enhance positive aspects of the proposed development. Environmental impacts of many past developments have left a trail of havoc through many ecosystems, especially in the third world. In many of these countries burgeoning fiscal deficits, corrupt administrators, poverty, illiteracy and wars are among the factors affecting the prevailing attitude of 'survival comes first and the environment will be dealt with later'. Nevertheless, engineers are being held increasingly accountable not only for short term impacts or benefits of developments but also for those which may accrue for generations to come. It is thus important for engineers to be fully informed concerning the requirements, methods and guidelines in this increasingly important field. The role of the engineer is critical in combining information from various disciplines into a 'master plan' that will best satisfy all criteria and will ensure the client that the project is completed on time and on budget. It is with this role in mind that the use of an expert system based computer program to assist in the environmental impact assessment of proposed projects was born. This utility is envisaged to be used as a tool to aid EIA and is not intended to replace a detailed assessment or any of the stages in the IEM procedure. The idea is to develop a cheap and accurate tool for identifying proposals that would have significant negative impact on the environment before any further work on them is undertaken. This would result in significant savings in time and money.

4.2 The principles behind IEM

The basic principles behind integrated environmental management are summarised from the Integrated Environment Management Procedure, Guideline Document 1. (6) as follows:

There should be

- Informed decision making
- Accountability for information on which decisions are taken
- The term 'environmental' encompassing a wide variety of aspects such as physical, biological, social, economic, cultural, historical and political.
- An open, participatory approach in the planning of proposals
- Consultation with interested and affected parties
- Consideration of alternative options
- An attempt to alleviate negative impacts and enhance positive aspects of the proposals
- An attempt to ensure that 'social costs' of the development be outweighed by the social benefits
- Democratic regard for individual rights and obligations
- Compliance with these principles during all stages of planning, implementation and decommissioning of the project
- The opportunity for public and specialist input in the decision making process

4.3 The Integrated Environmental Management Procedure

The IEM procedure is summarised from the Integrated Environmental Management Guideline Document 1 (6). The IEM flow diagram is presented in figure 4.01 and a brief description of each stage is presented. The IEM procedure can be divided into three stages:

- Stage 1 : Plan and Assess Proposal
- Stage 2 : Decision
- Stage 3 : Implementation

4.3.1 Stage 1: Plan and Access

Develop Proposal

The most important concept of IEM is that it should be incorporated into the planning process of a new development rather than be addressed once the planning has been completed. It is recommended that certain steps be implemented in the planning process to streamline the decision making process. These steps include the establishing of policy, legal and administrative requirements. These establish the framework within which the new development can be planned.

The IEM Procedure

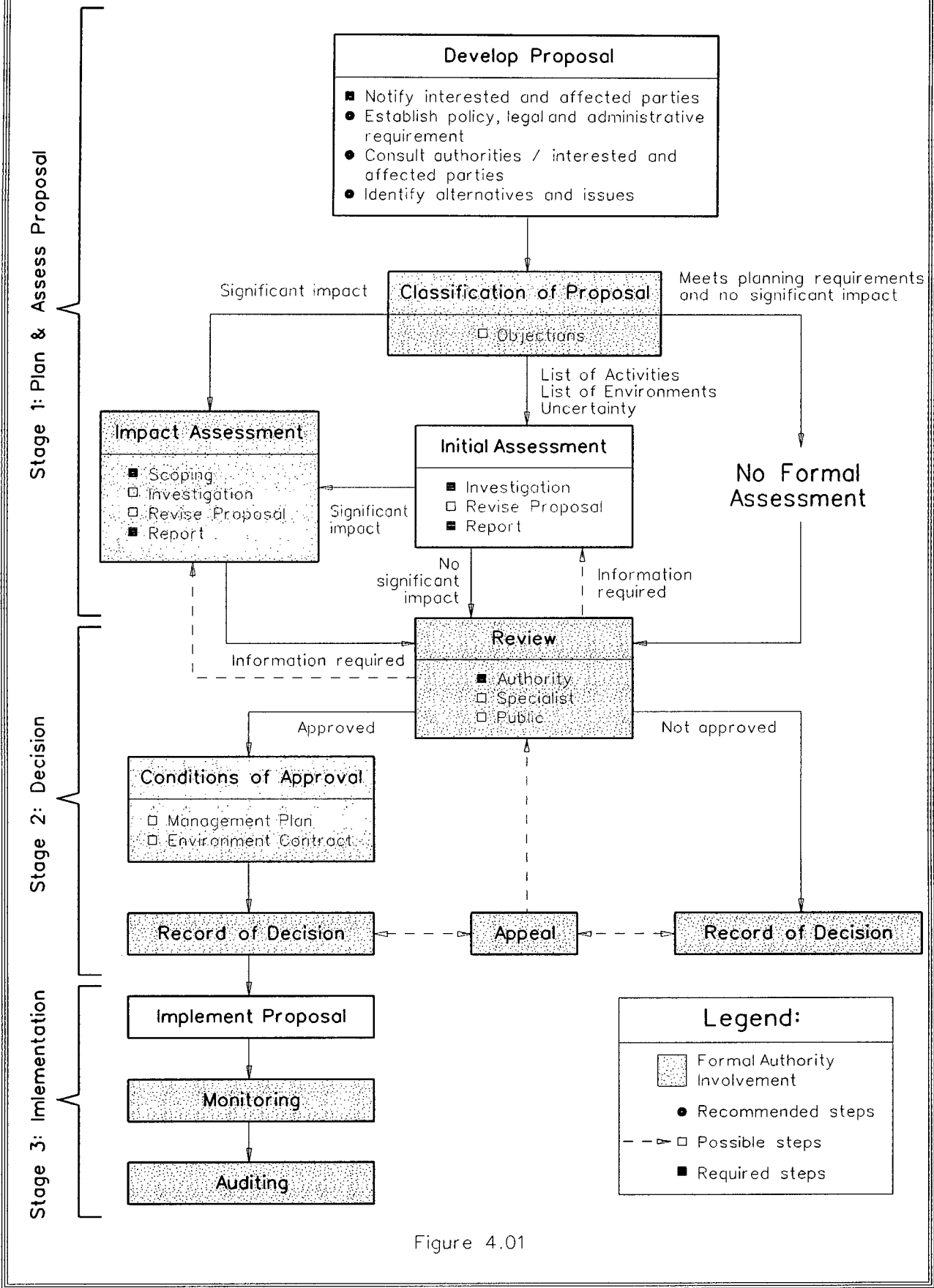


Figure 4.01

One of the most important aspects of IEM is to document the List of Activities, List of Environments, Summary List of Environmental Characteristics and the identification of authorities and interested and affected parties. Negotiations with these groups should provide various alternatives and understanding of the issues and constraints that should be considered in the planning of the development. It is believed that by undertaking these steps at the start of the planning process the planning and assessment stages will be combined and will thus assist the progress of the plans and will result in informed decision-making.

Classification of Proposal

This stage determines which route the proposal will follow, Impact assessment, Initial Assessment, or the No Formal Assessment route. The decision of which route the proposal will take is made by the planner with his/her consultants and in consultation with relevant authorities. The three routes are summarised as follows:

- **Impact Assessment.** This route is chosen when it is clear that the proposal will result in significant environmental impacts.
- **Initial Assessment.** This route is chosen when the proposal is included in the List of Activities, in the List of Environments or when it is unclear whether the proposal will result in significant impact.
- **No Formal Assessment.** This route is chosen if the proposal meets planning requirements and it is clear that the proposal will not result in significant environmental impact.

Impact Assessment

An Impact Assessment must be conducted should it be clear in the Development Proposal stage or the Initial Assessment that there will be significant impact, should the development take place. There are three main stages to the assessment:

- **Scoping.** In this stage the limits, alternatives and issues, the procedure to be followed and the report requirements for the investigation are determined in consultation with relevant parties.
- **Investigation.** The Investigation is conducted with the guidelines set up in the scoping stage. Information is collected regarding positive and negative aspects of the proposal including feasible alternatives.

- Report. This should be based on the Guideline Document 3 (7) together with any requirements determined in the scoping stage.

The final report should be reviewed by the authorities and interested and affected parties. Should the report be inadequate, contain inaccuracies or insufficient information, the report must be returned for further investigation.

Initial Assessment

Should the proposal be included in the List of Activities or in the List of Environments, but no significant environmental impacts have been identified during the Develop Proposal stage then an Initial Assessment must be completed. This investigation may also be necessary should any questions arise based on the Checklist of Environmental Characteristics as to whether the proposal will result in significant impact.

The purpose of the investigation is to collect just sufficient information to decide whether or not significant impacts will occur. The work should be undertaken by individuals who have the necessary expertise. An Initial Assessment report must be produced and should contain the findings as to the likelihood of the impacts. Should it be found that significant impacts are likely to occur the proposal will automatically go to Impact Assessment and the report will form the background information for the scoping stage of the Impact Assessment. Should the findings be that no significant impact will result the consultants must address the questions in the Summary List of Environmental Characteristics and combine these answers together with the report which will be forwarded to the Review.

No Formal Assessment

If the proposal meets the planning requirements and the answers to the Summary List of Environmental Characteristics indicate that no significant impacts are likely to occur the proposal should be submitted for Review. The proposal can be returned to Initial Assessment should further information be required.

4.3.2 Stage 2: Decision

When the reviewing authority is satisfied that sufficient information has been provided, that there has been sufficient consultation with interested parties and that all requirements have been complied with, a decision regarding the acceptability of the proposal should be taken. Should any

of these not be satisfactorily complied with, the proposal can be sent back to Initial Assessment or Impact Assessment. Conditions of approval are drawn up should the proposal be approved. These conditions are set out with regard to various planning, policy, legal and administrative requirements. As part of the Conditions of Approval a Management Plan or an Environmental Contract may be required.

Record of Decision

In order to encourage accountability the Record of Decision should document and explain how all decisions were made leading up to the approval or rejection of the proposal. The Record of Decision should also contain the Conditions of Approval.

Appeal

An opportunity for appeal should be provided by the decision making authority.

4.3.3 Stage 3: Implementation

Implementation of proposal

Should the proposal be approved a Management Plan or Environmental Contract may be required as stipulated in the Conditions of Approval.

Monitoring

Monitoring ensures that the requirements as laid out in the Conditions of Approval and Management Plan and Environmental Contract are adhered to. The monitoring program should include guidelines as to the parties responsible for carrying out the work, financing it, as well as verifications of impact predictions and adherence to approved plans.

Audits

In this aspect assessments are made periodically of the environmental impacts as the project progresses. In this way, feedback is obtained to review the accuracy of the investigations made as well as the decisions and Conditions of Approval.

4.4 The Role of the Engineer in IEM

It is a widely accepted point of view that the process of planning, designing, and building a project and securing the necessary approval has become a very complex process. Projects can no longer be assessed within a narrow range of benefits and costs but must now be reviewed in light of the full range of environmental impacts that they may have. Therefore from an engineer's point of view the achievement of a more socially acceptable and desirable end product involves extra time and cost together with a great deal of uncertainty.

The extra time involved to include an environmental study for a major project may be very great. This time is spent in the assembly of information on potential environmental impact, studies for interaction between the proposed project and the environment, meetings with interested parties and public hearings as well as the final assessment and implementation. The effect of this lengthy process on project schedules is vitally important to both the owner and the engineer. Clearly there would be significant savings in time, could the engineer obtain results of these assessments early on in the planning process.

Ultimately more time translates into higher costs. Apart from the direct cost of the impact assessment there could be costs due to changes in design, delays, inflation and loss of revenue as well as extra professional fees. Nevertheless it would be incorrect to give the impression that significant extra costs are always attributable to IEM procedures. For the majority of small to medium sized projects the additional costs would be relatively small.

The problem of uncertainty arises from the fact that no matter how well the IEM procedure has been undertaken and various measures incorporated into designs to mitigate negative impacts, the reviewing authority may still reject a proposal. This highlights the importance of accuracy of IEM studies together with the possibility of obtaining an approval in principle at an early stage in the planning process. A further problem presented is the question as to what extent an engineer is qualified to carry out the environmental assessment and this appears to be a subject in which there are many differences in opinion. Whether they are suitably qualified or not, engineers will, by the very nature of their function, be involved in these issues.

IEM is an essential tool in project planning and design and will go a long way to avoiding some of the mistakes that have been made in the past and to help build better projects. However, as far as engineers are concerned, they have an obligation to their client to examine the process critically and to make it work smoothly and effectively. To this end tools to enable fast and accurate environmental impact assessment will be of great assistance.

4.5 Impact Assessment Methodology

A methodology is defined in this context as a technique for conducting an environmental investigation typically in the Initial Assessment or Impact Assessment stages. Several different methodologies have been used. These will be discussed in the following categories :

- Basic Methods
- Extended Component Interaction matrix
- System Diagrams
- Quantitative Methods
- Ecosystem Simulation Modelling

a) Basic Methods

Examples of basic methods include guidebooks, multi-disciplinary teams, overlays, and the Leopold matrix. Guide books include various reports and articles describing general methods and procedures to be followed in impact assessment. There is however a general weakness in the accuracy of these methods. The multi-disciplinary team will, without the use of further methodologies, produce a report in terms of the treatment of different impacts and in decision making. The Leopold matrix method was that method that was usually identified as being environmental assessment methodology. It is simple, economical and comprehensive and usually indicates the magnitude and importance of environmental impacts. It can be used successfully for the evaluation of alternatives, however its weakness is that it can encourage the filling in of blocks with questionable subjective judgment. The use of overlays is also common in environmental planning. The idea involves the superposition of mapped environmental factors so as to arrive at a composite conclusion. This method is utilised by computers to superimpose mapped overlays. The strength of this technique lies in its simplicity and understandability.

b) Extended Component Interaction Matrix

This particular method was developed because it was believed that the Leopold matrix was too simple and could not identify second and third order impacts (changes in a complex environmental chain as a response to an impact on one component in the chain). The method begins by using the standard Leopold matrix. Listing the environmental impacts horizontally and vertically direct dependencies between components are identified. Using matrix multiplication techniques (from network analysis) the number of second and higher order dependencies is determined. Therefore impacts on the dependencies of a component can be traced throughout the environmental system being studied.

c) System Diagrams

In this method direct impacts are linked with higher order or secondary impacts in a visual representation of the relationship as well as the energy flows (or energy pathways) between impacts. The method suffers from several drawbacks. The first is that the method is only limited to ecological impacts and cannot deal with social or aesthetic impacts. Secondly the data required to produce these diagrams is generally too onerous to collect and the time period needed for collection is too great. Thirdly, there is also no guidance given on determining the significance of the various impacts and the monitoring of these impacts is not catered for.

d) Quantitative Methods

There are three major methods in this category : Water Resources Assessment Methodology (WRAM), Decision Analysis and the Sondheim Method. They are all variations of the theme of quantification, scaling and aggregation of impacts as first presented in the Environmental Evaluation System (EES) developed by Whitman et al(17). In EES impacts on a variety of environmental, economic and social components are measured in common units on any particular scale, by the use of value functions. These functions quantify each input on a scale of environmental quality in which 0 represents the worst state and 1 the best. These scaled scores are then multiplied by weights assigned to each component where the weights represent the relative importance of each component. All these weighted component scores are then summated to give an overall score for that scenario. A choice is then made between the alternative scenarios by comparison of the scores for each.

One particular variation tries to use standard weights for each environmental component. Different projects are thereby compared by measuring the amount of change it might produce based on a universal index. The quantitative methods in general have received some severe criticism. Some of these are :

- That the factors which quantify the components environmental quality are based on value judgements and are not socially derived. For example they do not use units such as kilocalories to measure the impact.
- That a great deal of information has been lost in the conversion to numbers.
- By producing scores, these methods effectively remove decision-making from the hands of the decision-makers and this transfer has been termed the 'professional theft of decision making'(Bisset,10)

- The idea of weighting component impacts of a proposal before implementation suffers from a major weakness. This weakness is that there is no guarantee that the selected weights will be representative of the actual magnitude of impacts or views of those communities who experience the impacts once the proposal has been implemented.
- These methods produce average results which can be a grey area, thereby leaving investigators undecided.
- The composition of the weightings biases the outcome of the assessment.
- The results tend to be technocratic resulting in interested parties not understanding them.
- Assumptions such as those using standardised factors may be too oversimplified to be applicable.

Some of these criticisms has been as follows :

Hirst (14) notes: "Despite the academic preoccupation with numerical indices and their sophisticated manipulation, managers and decision-makers have difficulty in relating to them and incorporating their information content into rational action in the development process."

Likewise Beanlands (15) suggests that the ranking and weighting systems complicate the decision making process by inserting a new set of information between the decision maker and the available fact.

Bisset (10) criticises complex and quantified approaches with their "inflexible reductionist and mechanistic approach...(and) rigid logico-mathematical frameworks"

e) Ecosystem Simulation Modelling

In this method a model of the ecosystem(s) that are likely to be affected by the proposal is constructed. The model specifically includes information on ecosystem trends and dynamics so that the proposal can be assessed in the context of the environmental changes that are likely to occur. For any particular proposal a model is developed, investigated and the behaviour determined in the context of a number of intensive workshops. These generally consist of experts in relevant disciplines, modelling experts and decision-makers who will use the information obtained from the model.

This method of EIA is still in its infancy and very few successful models have been developed. There is a danger in developing a model quickly for environmental systems about which little is known and as models can quickly grow to become very unwieldy, models tend to be less comprehensive than other methods.

In summary it would appear that the quantitative methods will have a decreasing role to play in impact analysis. This was supported by the findings of a study done in the United States by researchers who were responsible for developing EES. They stated that EES was not popular amongst decision makers and the public in general. These researchers have now developed a new method which uses qualitative information only. This trend is reflected in the United Kingdom where there has been realisation that impact analysis is a part of a political process. Use is now made of a method described in a manual for the assessment of major industrial developments which contains a clear rejection of summing weighted components and places emphasis on providing information that is in a qualitative and 'broken down' form.

4.6 The Proposed Methodology

The concept of using expert systems in environmental management is not new. Several papers have been presented describing expert system based environmental assessments. Examples of these include:

- An Expert System Approach to Environmental Assessment (Lien, 16).
- Expert Systems : An Approach to Problems in Ecological Management that are Difficult to Quantify (Starfield and Bleloch, 9).
- Applying Expert System Technology to Carrying Capacity Assessment : A Demonstration Prototype (Lien,12)

All these papers use examples that have tended to concentrate on limited and clearly defined domains. Researchers have generally avoided large systems that may risk over-generalisation or over-simplification of problems. This is especially relevant as it has been recognised that although there are many EIA methodologies there are no optimal methodologies that have general applicability and which enable all the tasks associated with EIA to be accomplished in a satisfactory manner (Shopley and Fuggle,13)

Even so researchers have attempted to produce integrated systems of general applicability. The most recent of these has been HyperAIA - an Integrated System For Environmental Impact

Assessment (Antunes and Camara,11). HyperAIA is a computer program framework that uses an integrated methodology whereby the acquisition and integration of expert knowledge is combined with both qualitative and quantitative EI assessment methodologies to evaluate a problem. Antunes and Camara state that "it is not possible to develop a computerised system including all the relevant knowledge for all the possible types of EIA which may occur, and thus an empty shell architecture was chosen for the program development". By empty shell architecture is meant that at the beginning of an application the shell can be empty of information, but when the investigation is underway, relevant information is fed in from various participants in the EIA. Once loaded with this information HyperAIA is then used as a decision support system and assists in the selection of the most desirable alternatives. It does, however use some built-in knowledge which is represented as lists of environmental variables and simulation models applicable to other EIA problems. It is important to note that HyperAIA attempts to analyse the problem accurately by using simulation models for the forecasting of environmental effects. These models include simple mathematical models, dynamic models and verbal models.

In this thesis the writer proposes that there is a case for a computer based system of general applicability that can function as a primary assessment tool in order to perform a quick assessment of a proposal. It is envisaged that this system be used early on in the conceptual stages of a project to direct planners to the alternatives with the least environmental impact. It is not intended to replace a full EIA study but function as a pointer to direct planners to feasible alternatives before time or money can be spent on full EIA studies of unfeasible alternatives.

In order to build a computer based environmental analysis, a means by which these characteristics or factors could be evaluated using the computer had to be found. For this reason it was decided to use an expert system shell and build up a set of rulebases around the characteristics or factors that were to be evaluated. The nature of the characteristic assessment is qualitative and relies on heuristic problem solving reasoning (as described in chapter 3) as opposed to quantitative analysis. Furthermore the environmental characteristics contain large amounts of domain specific knowledge. These factors point towards an expert system style approach rather than that of traditional computing methods that use algorithmic methods.

There are nevertheless two major problems associated with this approach. The first is that not every development proposal requires every environmental characteristic or factor to be included. For example a proposed development in an agricultural area does not require the Potential Land Use factor for industrial, commercial or residential areas to be evaluated. Neither do inland developments generally require a marine impact assessment to be carried out. Every development proposal requires a unique set of environmental characteristics or factors depending on where it is located and what its future function will be. For this reason a single expert system rulebase could

not be used but rather a whole series of smaller rulebases that can be selected according to site requirements to make up an overall impact assessment.

The second major problem is that for every development proposal, the importance of every environmental characteristic or factor chosen will vary, again according to its future function and location. This suggests that a weighting be applied to each environmental characteristic in accordance with the analyst's experience and discretion. A method is thus presented whereby for any development proposal a unique set of expert system rulebases can be selected. Each rulebase would relate to a different environmental characteristic or factor and would have a weighting allocated to it where the sum of the weightings used in each development proposal will total 100. The result of the assessment of each factor is multiplied by its weighting and all the results are added together to give an overall assessment of the proposal. This method uses both qualitative (in the rulebases) and quantitative (in the weightings) techniques. The rulebases are qualitative assessments because they use heuristic problem solving reasoning that has been captured in the rulebase from a domain expert.

The use of multiple rulebases has several potential advantages. The first is that each rulebase can be created or administrated by an expert who is specially qualified in that domain. For example the rulebase that assesses marine impact could be maintained by a marine biologist or scientist. The rulebase pertaining to the water infrastructure services could be maintained by a civil engineer. The Potential Land Use and Landscape Character rulebase could be maintained by a town planner. When the overall development proposal is evaluated it will use the expertise of all these specialists in their various fields. The second major advantage is that when the rulebases are broken down into subtopics they are easier to maintain, edit and incorporate new domain specific information.

This method, although versatile, immediately presents a complex problem. This problem is that of building a framework by which this method of analysis can be carried out in the simplest manner possible so that it can be used by any analyst regardless of his/her level of computer literacy. For this reason the computer utility was created to provide a simple user friendly interface to facilitate development proposal assessment using multiple rulebases. This utility is described in chapter 5.

4.7 Rulebase Design Criteria

In order for the rulebases to maintain a usefulness over time they had to be coded in such a way as to be simple, easy to maintain, but maintain a high degree of accuracy. The three design criteria are thus simplicity, maintainability and accuracy.

The method of analysis as proposed in section 4.6 goes a long way to realising these goals. The analysis uses a selected number of small individual rulebases, each coded to analyse a specific environmental characteristic. This in itself resulted in a very simple process of building, testing and debugging each small rulebase by itself to ensure that the rulebase performance provided accurate assessments of impact for various scenarios under which they were tested. This concept of "divide and conquer" ensures that the rulebases remain relatively simple and can be easily maintained, edited, updated and tested at future dates in order to keep the system abreast of new methodologies, ideas or legislation.

One of the most significant advantages of this method is that the rulebases can be used in a stand-alone analysis. That is they do not have to be incorporated in a batchrun with a selected number of other rulebases but can be run individually. An operator may consult with the expert system using that particular rulebase alone. A further advantage of this method is that new rulebases can easily be added without affecting the operation of any of the others thereby further enhancing the maintainability of the system and the rulebases as a whole.

4.8 Knowledge Acquisition Procedure

Knowledge acquisition has been recognised as one of the most important and problematic stages of developing expert systems. In this stage the knowledge engineer gives attention to the procedures used to gather knowledge and the methods that will be used to represent or model that knowledge. At least three different approaches have been identified in which knowledge can be acquired. These are :

- o interviewing experts;
- o learning from documentation and explanations; and
- o learning from example

In all practicality no single approach is necessarily better than any others and in most cases knowledge is acquired using a combination of two or more of these approaches. However in all of these approaches the knowledge acquisition methodology must attempt to achieve the following goals :

- a) investigate the analytical characteristics of the expert source.
- b) investigate the judgmental behaviour displayed by the expertise source.
- c) categorise the expertise and decide how to model it.

It was decided that due to the diversity and generality of the system that the categorising and the structure of the rulebases should follow the guidelines as laid out in the Checklist of Environmental Characteristics, Guideline Document 5 (8). This document categorises the major environment characteristics and linkages that should be considered by environmentalists or planners. Each characteristic is divided into subtopics and specific aspects of each subtopic are listed. The rulebases are coded along the lines of the categories of the environmental characteristics. The relative importance of each selected rulebase used in the computer utility requires weightings to be attached according to the relative significance as seen for each individual proposal being evaluated. These weightings are inputted by the user at runtime.

The document does not, for obvious reasons, attempt to attach importance factors or weightings to each individual component of the characteristics as coded in the rulebases. For this, expertise can only come from individual experts who are encouraged to continually modify the rulebases to reflect their experience and expertise in the topic. It is envisaged that owing to the wide range of environmental characteristics covered by the rulebases that the expertise could conceivably come from a number of differing professions such as engineering, biologists, zoologists, townplanners and environmentalists. It is expected that the rulebases will be edited and improved upon almost indefinitely and thus at no stage is the system considered to be 'finished'.

4.9 Environmental Characteristics in Project Evaluation

The ecological characteristics that have been used for the design and implementation of the rulebases have been obtained from the Checklist of Environmental Characteristics, Guideline Document 5 (8). The characteristics together with a description of the rulebases associated with them are documented below.

4.9.1 Physical Characteristics of the Site and its Surroundings

The categories on which the physical characteristics of the site are assessed are land, climate, freshwater systems and marine and estuarine systems. As no proposal will have all four categories present, three separate rulebases have been prepared using the selected combined categories. These are :

Categories	Rulebase Name
o Land, Climate	LandClim.rb
o Land, Climate, Freshwater Systems	LanClimW.rb
o Land, Climate, Marine and Estuarine Systems	LanClimM.rb

When this characteristic is selected only one of these rulebases should be selected based on which categories are relevant to the proposal.

4.9.2 Ecological Characteristics of the Site and its Surroundings

The categories on which the ecological characteristics of the site are assessed are vegetation, animals and natural and semi-natural communities. All proposals will have at least some Impact on all these categories therefore only one rulebase was prepared that contained all three.

Categories	Rulebase Name
o Vegetation, Animals, Communities	Eco.rb

4.9.3 Current and Potential Land Use and Landscape Character

The categories on which this characteristic of the site are assessed are General considerations for all development proposals, Urban open space and protected and recreational areas, residential areas, commercial areas, industrial areas and agricultural and silvicultural areas. As no proposal will have all these categories present, five separate rulebases have been prepared using the selected combined categories. These are :

Categories	Rulebase Name
o General Considerations, Urban Open Space	CharOpen.rb
o General Considerations, Residential Areas	CharResd.rb
o General Considerations, Commercial Areas	CharComm.rb
o General Considerations, Industrial Areas	CharInds.rb
o General Considerations, Agricultural Areas	CharAgri.rb

When this characteristic is selected only one of these rulebases should be selected based on which categories are relevant to the proposal.

4.9.4 Cultural Resources

There are no individual categories on which the cultural resources characteristics of the site are assessed. For this reason there is only one rulebase that was required.

Categories	Rulebase Name
o All aspects of Cultural Resources	Cultural.rb

4.9.5 Infrastructure Services

The categories on which the infrastructure service characteristics of the site are assessed are energy supply, water, waste management, transportation networks, education, housing, telecommunication and financial implications to the region. As no proposal will have all four categories present and most would have them in almost any combination it was decided to develop a separate rulebase for each category.. These are :

Categories	Rulebase Name
o Energy Supply	Energy.rb
o Water	WaterInf.rb
o Waste Management	WasteInf.rb
o Transport Networks	Transprt.rb

Rulebases have not been developed for all the categories. It was decided at this stage only to develop rulebases for the most critical aspects. The others could be added at a later date.

4.9.6 Social and Community Services and Facilities

The categories on which the social and community service facilities characteristics of the site are assessed are health service facilities, emergency services and recreational facilities.

It is believed that all these categories would have a role to play in a proposed development and thus only one rulebase was implemented.

Categories	Rulebase Name
o Health Service, Emergency Service and Recreational Facilities	Social.rb

4.9.7 The Nature and Level of Present and Future Environmental Pollution

The categories on which the nature and level of present and future environmental pollution characteristics of the site are assessed are air, water, noise, visual and solid and liquid waste. As no proposal will have all four categories present and most would have them in almost any combination it was decided to develop a separate rulebase for each category. These are :

Categories	Rulebase Name
o Air Pollution	AirPol.rb
o Water Pollution	WaterPol.rb
o Noise,Vibration and Lighting	NoisePol.rb

Rulebases have not been developed for all the categories. It was decided at this stage only to develop rulebases for the most critical aspects. The others could be added at a later date.

4.9.8 Risk and Hazard

There are no individual categories on which the risk and hazard characteristics of the site are assessed. For this reason there is only one rulebase that was required.

Categories	Rulebase Name
o Risk and Hazard	Risk.rb

4.9.9 Health and Safety

There are no individual categories on which the health and safety characteristics of the site are assessed. For this reason there is only one rulebase that was required.

Categories	Rulebase Name
o Health and Safety	Health.rb

4.9.10 Cumulative and Synergistic Effects

There are no individual categories on which the cumulative and synergistic effects characteristics of the site are assessed. For this reason there is only one rulebase that was required.

Categories	Rulebase Name
o Cumulative and Synergistic Effects	Synergis.rb

4.10 Project Evaluation

The way in which the system combines these multiple rulebase options to give an overall assessment of a proposal is designed with the aid of a hypothetical example and reference to figure 4.02.

For a given hypothetical proposal a set of four rulebases has been chosen for the analysis. When these rulebases are chosen they are assembled together in what is termed a batchrun. Together with a list of rulebases to be evaluated in the batchrun a set of weightings are required which describe the relative importance of each of the environmental characteristics the rulebases represent. These weightings are inputted by the user as the batchrun is being set up.

When instructed the system executes the batchrun. As the batchrun is executed each consecutive rulebase is run prompting the user for answers to the questions as required. At the end of each rulebase consultation the expert system provides a final assessment and allows the user to browse through the reasoning process used by the expert system to arrive at its

assessment. Once the user has selected to exit from that consultation the results are stored in a results file and the system goes on to run the next rulebase. This process continues until all the consultations have been run and the program returns to the main menu.

From here the user can choose to analyse the results whereby all the result data from each rulebase consultation is read in from the results file and a final overall assessment is calculated using the corresponding weighted factors for each rulebase. A final result is thus presented to the user for the proposal. In order to evaluate alternative proposals the user can choose to re-run the batchrun using data from alternative proposals and thereby compare the overall result or he/she can study the results from each rulebase individually. The exact manner in which the software operates this process is described in section 5.9.

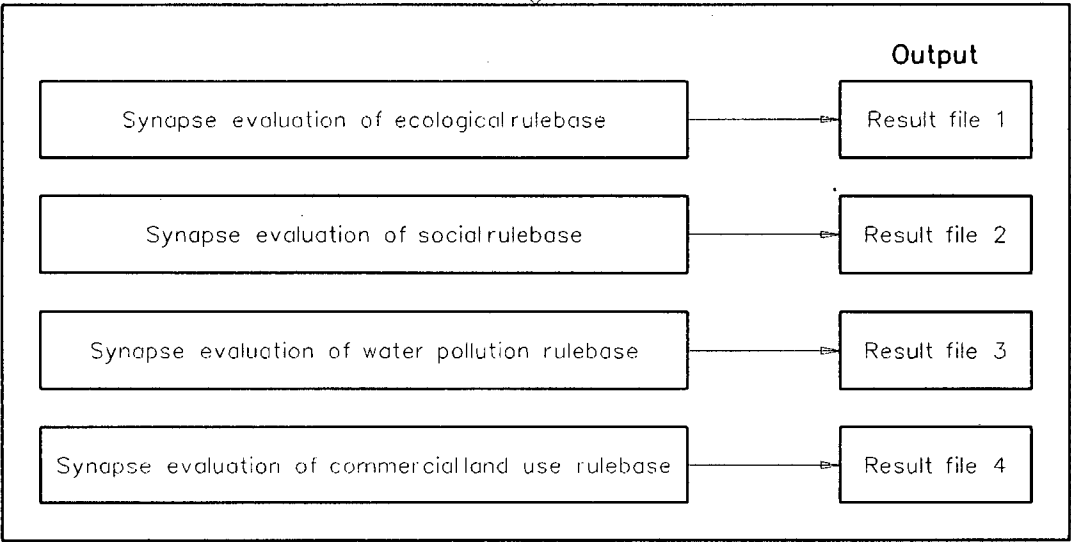
The user has many other alternatives available to him/her. An expert system rulebase can be re-run individually, new batchruns can be set up or the weightings of the batchruns can be adjusted. The composition of the batchruns can also be altered with the capabilities for rulebases to be added or deleted.

Hypothetical EIA Study

Batchrun preparation

Chosen components of the study	
<u>Rulebases</u>	<u>Weightings/ Importance Factors</u>
Ecological	35%
Social	15%
Water Pollution	23%
Commercialland use	27%
	<u>100%</u>

Evaluation of batchrun



Result Analysis

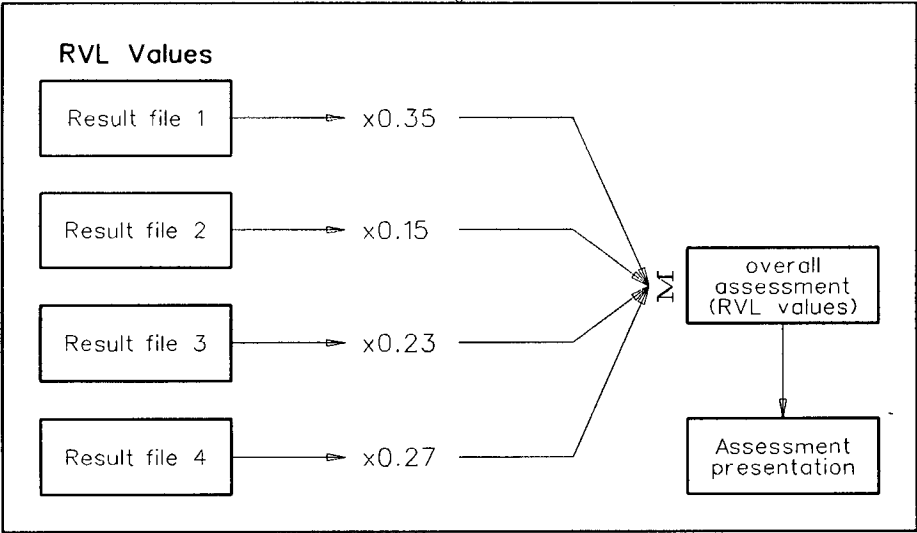


Figure 4.02

CHAPTER 5

OBJECT-ORIENTATED PROGRAMMING AND THE PROGRAM STRUCTURE

*"To understand everything
makes one very indulgent"*

Madame de Stael (French Writer)

Contents

- 5.1 Introduction**
- 5.2 Object-Orientated Programming**
 - 5.2.1 Encapsulation
 - 5.2.2 Inheritance
 - 5.2.3 Polymorphism
- 5.3 Turbo Pascal and Turbo Vision**
- 5.4 Basic Principals of the Program Operation**
- 5.5 Design Criteria**
- 5.6 Program Design and Structure**
 - 5.6.1 Overall Operational Review
 - 5.6.2 Detailed Operational Breakdown
 - 5.6.2.1 Object Hierarchy
 - 5.6.2.2 Program Flow Charts
- 5.7 Data Storage Structures**
 - 5.7.1 Collections
 - 5.7.2 Streams
 - 5.7.3 Resource Files
- 5.8 Rulebase and Program Interfaces**
- 5.9 Databases**
- 5.10 Manipulation of Result Data**

OBJECT-ORIENTATED PROGRAMMING AND THE PROGRAM STRUCTURE

5.1 Introduction

The introduction of object orientated programming has revolutionised the way computer programs are written and has been equated in importance to the introduction of structured programming 20 years ago. While much debate still continues over the merits of object-orientated programming more and more programmers are switching to this new standard.

Programs are now written by teams of programmers and the programs themselves have grown larger to take advantage of increased computing power on PC's. The debugging and maintaining of these programs has thus grown exponentially more difficult. The most obvious advantage of object-orientated programming is that it addresses these problems inherent in debugging and maintaining large and complex programs.

The most important concepts in object-orientated programming are encapsulation, inheritance and polymorphisim. With these three concepts programmers are given a vocabulary for describing relationships and processes in abstract terms - the way people think. The computer language Turbo Pascal version 7 which was used, has all these object-orientated features and capabilities. Building on these concepts the object-orientated application framework called Turbo Vision was developed. The basic idea was to provide a basic platform on which application programs could be written with out "re-inventing the wheel" each time a new program was started. It was together with Turbo Vision libraries that the computer program utility was developed.

The actual structure and working of the program utility is complex but the most important feature is the modularity and extensability of all the program modules which provide for future modification. In this chapter these object-orientated concepts are discussed together with the design criteria, the structure of the program, data storage structures, program and database interfaces and finally the manipulation of result data.

5.2 Object-Orientated Programming

Object-orientated programming (OOP) began in the late 1960's with a language called Simula 67 which was designed for writing simulations. Since then other OOP languages such as Small Talk, LOOPS, Flavors, Object Pascal, C + +, Actor and Turbo Pascal have been developed. This concept of programming has become popular due to the complexity of modern software applications. OOP addresses these problems of managing, maintaining, debugging and expanding

large complex programmes. One of the problems of OOP is that none of the current languages implements OOP concepts in exactly the same way. For example, Turbo Pascal's unique OOP extensions were "borrowed" from both C++ and Object Pascal. The capabilities of OOP can be understood by examining three major concepts. These concepts are encapsulation, inheritance and polymorphism.

5.2.1 Encapsulation

The key feature of OOP is that data and functions that operate on the data are encapsulated into a self-contained object. In OOP the object is thus a data structure that contains variables and related processes called methods. Instead of writing procedures and functions that act upon data, in OOP, objects are created which know how to perform actions on themselves. In Pascal an object is also known as an instance and a class is called an object.

Encapsulation provides numerous advantages. For example data within an object can be declared private and is accessible only within that object. This is useful for protecting data in multi-programmer projects. Furthermore large programs are more readable because all of their related code and data is in one place. This results in object-orientated programs being more modular and easily maintained.

5.2.2 Inheritance

Inheritance refers to the ability of new objects to inherit the characteristics (data and methods) of other objects. OOP provides the ability to create new objects by extending those that already exist. In this way OOP simplifies building new programs from existing program libraries, usually without having to make extensive modifications to programs that already work, this prevents the classical reinventing of the wheel each time a new program is written.

Certain OOP languages support multiple inheritance which allows the creation of a new object with the characteristics of more than one previously defined class. This ability is not supported in Turbo Pascal. Through inheritance a hierarchy of objects can be created to form the backbone of the program. However designing a useful and elegant object hierarchy is difficult and takes considerable planning and skill.

An object that inherits the characteristics of another object is called a descendent object. The object from which the inherited features come is called the ancestor object. The rule is

that descendants can only have one immediate ancestor but may have an unlimited number of descendants.

5.2.3 Polymorphism

The third aspect of OOP is creation of polymorphic objects. These objects can literally assume different forms when the program runs. A polymorphic object can take on the form of itself or any of its descendants. The means by which polymorphic objects are created is by defining virtual methods. The means by which the object decides how it is going to represent itself is called late binding and occurs at runtime as opposed to compile time.

Virtual methods provide Turbo Pascal with a special kind of extensibility that can be powerfully exploited in object libraries. These libraries can be used as is or overridden with customised versions of the methods. This results in code that is highly reusable and again prevents the reinventing of the wheel.

5.3 Turbo Pascal and Turbo Vision

Turbo Vision is a structured high-level language that can be used to write computer programs for any type or size of application. Turbo Pascal was chosen for this thesis because it is one of the world standards in computing languages, offering the latest in programming techniques and aids. These include a state-of-the-art integrated development environment, an object-orientated application framework, extensive debugging facilities and a protected-mode interface allowing access to all the computer's memory (over and above 640K limit - although not used in this program). In addition, Turbo Pascal offers the unique facility called Turbo Vision which was used to develop most of the programs and units for this thesis.

Turbo Vision is an object-orientated application framework for windowing programs. The idea behind it was to save programmers the task of endlessly recreating the basic platforms on which most application programs are developed. This framework includes the following features:

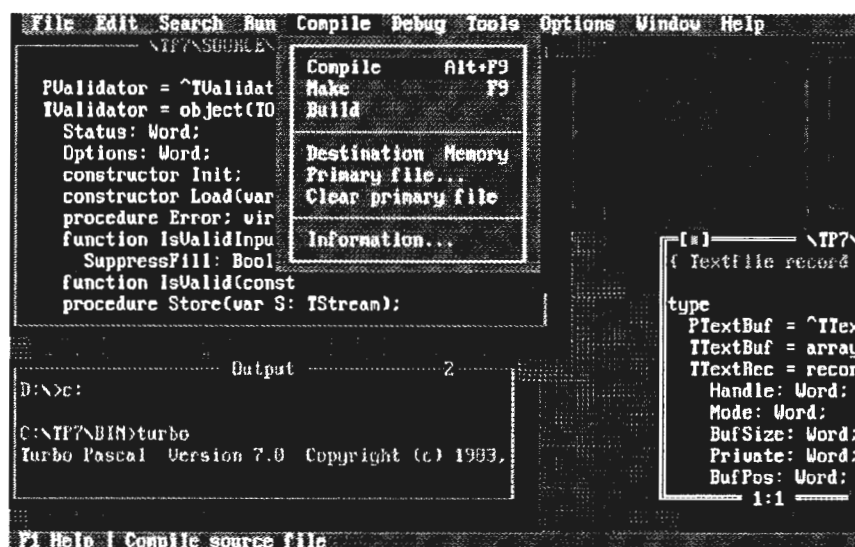
- Multiple resizing, overlapping windows
- Pull-down menus
- Mouse support
- Dialog boxes
- Data validation
- Built in color installation
- Buttons, scroll bars, input boxes, check boxes and radio buttons

- o Standard handling of keystrokes and mouse clicks

Turbo Vision has the appearance of a library containing procedures and functions of objects. However, it is important to note that Turbo Vision is not merely a library but an application framework. The idea is that using Turbo Vision the source code never has to be modified. In order to change code the programmer merely has to extend it. For example by using the concepts of inheritance (as described before) the objects are changed by deriving new objects that inherit the ancestors' characteristics and changing what is needed by overriding the inherited methods with new methods that are written for the new objects.

Turbo Vision is a hierarchy of objects, each descendent inheriting the ancestors' abilities plus whatever is changed or added. The concept is designed to be used in its totality with a single architectural vision behind every object. All Turbo Vision objects work together in many different interlocking ways and cannot be pulled out to be used on their own without major modifications to the code, which would probably involve more work than it would take to write it from scratch. Turbo Vision is event driven. Event driven programming enables the creation of flexible programs that give users control over what part of the program they want to use as opposed to the program execution being 'led' by the program itself. The event driven modes is the same one used by modern graphical environments such as Microsoft Windows.

In conclusion, Turbo Vision was created to provide a proven application framework on which programmers could develop their own applications without the enormous amount of unnecessary repetition involved in developing them from scratch. A typical appearance of this environment is shown in screen capture 5.01.



Screen 1 - A Typical Turbo Vision Environment

5.4. Basic Principles of Program Operation

In section 4.4 the methodology by which EIA could be applied to almost any geographical area was discussed together with a flow chart summarising this proposed solution. (figure 4.02). The purpose of the computer program utility is to provide a vehicle to carry out these multiple assessments in a smooth automated fashion. Most of the data manipulation can be carried out in a 'behind the scenes' approach together with various facilities available to the user. It is important to note that the user has access to varying levels of data manipulation based on his/her own computing skills. The basic principles of how the program will operate can be summarised under the headings as follows:

a) EIA of different sites/projects

- The user would register the name of the new site as a batch file.
- The selection of decision factor rulebases based on relevant site conditions.
- The corresponding selection of weighting or importance values of each decision factor to the chosen site
- The evaluation of the site using the decision factors and weightings chosen to produce a single overall assessment of the site
- The user could iteratively re-assess the site by re-running one or all decision factor rulebases with varying inputs until a satisfactory assessment is obtained. (This would correspond to modifications in the design or concept)

b) Creation/Modification/Viewing of Rulebases

- The user has the facility to create new rulebases or modify existing ones
- The user can carry out these operations to rulebases using either of two approaches
- The user can enter the editing facilities built into the computer program
- or the user can enter the SYNAPSE integrated environment within which the user can modify and compile rulebases
- The compilation of the rulebases can be accomplished by using the program facilities

c) The Linking of Rulebases with Databases

d) The Storage of Data Pertaining to Previous Analyses

all information pertaining to previous assessments is stored including decision factors and weighting. This can be reviewed, modified and reanalysed or deleted at any time

e) Miscellaneous features

Miscellaneous features are often included with programs to provide a richer user environment. These features include an extensive help facility, screen 'saving', screen colour modification, exiting to DOS without quitting the program and changing of video modes. These features listed are all available in this program. The utility also features a separate installation program.

5.5 Design Criteria

During the writing of the programs the following design criteria were implemented:

- User friendliness
- Reliability
- Accuracy
- Good code management
- Provision for code upgradeability and extensibility

Aspects of user friendliness include a clear and simple interface with help facilities, menus, dialog boxes, buttons, pop-up menus, picklists, etc. Other aspects include a user led program control regimen with provisions for exiting at any stage in the program without being locked into finishing the chosen program routines. Provisions for these aspects have mostly been provided for in the rich development environment of Turbo Vision. There is an extensive use of the pull down menu system and the use of window, dialog boxes, information boxes informs the user at all times of the current state of the program.

A comprehensive help facility is built into the program utility. At any phase in the program the user can access prepared help facilities describing inputs required, output, or what the program is doing. An added feature of this is that the help facility provides cross-referencing so that the user can obtain help or information regarding related issues at the same time. In all dialog boxes and menus the program contains a cancel button so that the user can cancel the current operation which he/she is undertaking without proceeding any further. To minimise constant user input, use is made of pop-up menus or picklists for ease of information retrieval. Buttons and mouse support provide the final touches.

The criteria of reliability and accuracy are met largely by careful programming and consistency checks built into the program to ensure that the type and format of data being handled is at all

times what the program is expecting. The point of weakness is at the time when information is being typed in by the user. For example if the computer requires a number to be inputted and instead a character string is inputted, this will cause an internal error at best or data corruption at worst. The method by which this is prevented is through the use of data validation objects. Validation objects are filters that accept data inputted by the user. They can be programmed to allow only what the programmer allows to pass through into the computer. For example characters only, numbers only, or any particular range, pattern or combination thereof.

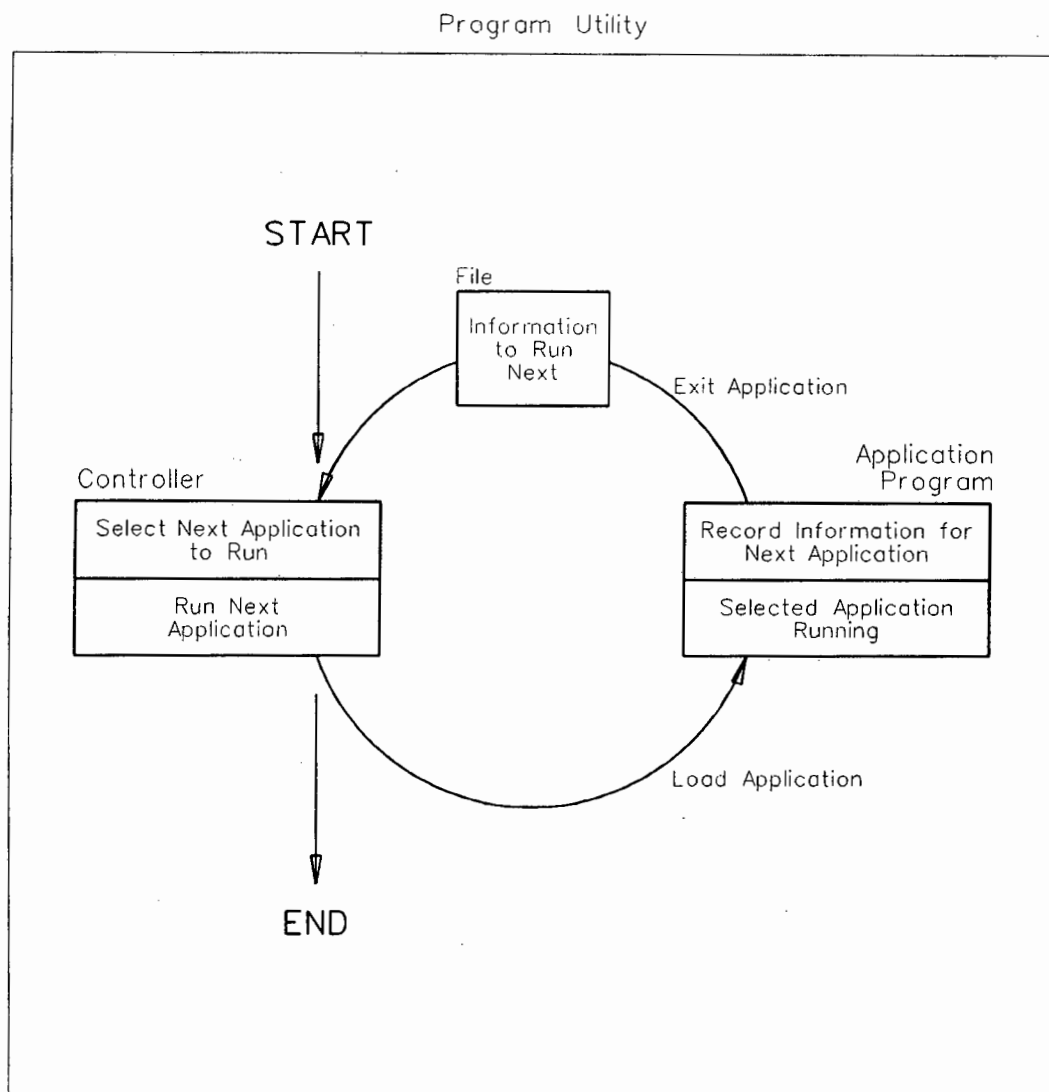
Good code management, upgradeability and extensibility are provided for by the use of object orientated programming. With the use of concepts such as encapsulation, inheritance and polymorphism described previously, these criteria including the management of large complex programs are achieved. Code management is further helped by dividing the code into units. A unit is a library of declarations that can be pulled into the program on demand. By arranging code in separate units containing all declarations that operate together for specific tasks, the management of code is greatly simplified. The design criteria listed above are thus implemented using a variety of interrelated programming features and techniques.

5.6 Programming Design and Structure

5.6.1 Overall Operational Review

Besides the design criteria the single most important factor governing the design of the program utility was the memory requirements of the expert system and the program itself. These programs do not use expanded or extended memory available in most modern computers. Instead they are confined to the 640KB conventional memory area, competing with other terminate and stay resident programs (TSRs) and DOS itself for space. For this reason the computer should be optimally set up when running this utility with most memory resident programs, device drivers and DOS itself loaded into the upper memory area using an extended memory manager. This will free up most of the 640KB conventional memory area for the expert system and utility itself.

Even though use of the memory has been optimised there is not enough conventional memory for the expert system and computer program utility to run concurrently. For this reason it was decided to control the execution of the programs so that they are being executed one at a time. In order to implement this a very small program called the controller resides permanently in memory and controls the sequence of operations. The sequence of operations is described as follows: (see figure 5.01)



Program Operation

Figure 5.01

The controller program is loaded into memory where it resides permanently until the user shuts down the utility. On 'waking up' the controller automatically calls the main program which in turn displays all its initialisation messages. The user continues to use the program utility until he/she wishes to run the expert system. At this point the main program terminates its execution, leaves a message to tell the controller program what to run next and then shuts down, and removes itself from memory. The program control is then passed back to the controller which retrieves the message and then runs the relevant expert system based on the instructions left for it by the main program. The expert system then has access to most of the conventional memory area according to its requirements.

Once the expert system has completed its analysis it terminates, removes itself from memory, but does not pass a message back to the controller program. The controller program is programmed to automatically re-run the main program again. One variation of this is when the user chooses to execute a batch run using several expert system rule bases (figure 5.02). Here the main program pre-prepares a DOS batch file containing all the DOS instructions needed to run the sequence of expert system analyses. Once the main program terminates, program control is passed back to the controller program which now executes the pre-prepared batch file. This batch file runs the expert system several times over using different selected rulebases and producing separate result files for each analysis. These result files are used for further result analysis when re-entering the main program. Once the DOS batch file has completed its sequence of operations program control is handed back to the controller program which now automatically runs the main program.

The operation of the controller program is completely transparent to the user and the changeover from one program to the next is almost unnoticeable. This concept of scheduling operations in order to use the restricted memory available is simple but works very effectively. The operation of the controller program can be likened to a last in first out (LIFO) stack. In this data structure the last instruction inserted on the stack is the first to be used, should no further statements be added control works its way back down the stack of instructions until finally quitting if no further instructions are found.

5.6.2 Detailed Operational Breakdown

5.6.2.1 Object Hierarchy

In section 5.2.2 the concept of inheritance and a hierarchy of objects was introduced. Turbo Vision provides an object hierarchy of standard objects which the user can build on. This hierarchy tree is shown in figure 5.03. Not all these objects shown in the Turbo Vision tree have

Turbo Pascal Object Hierarchy

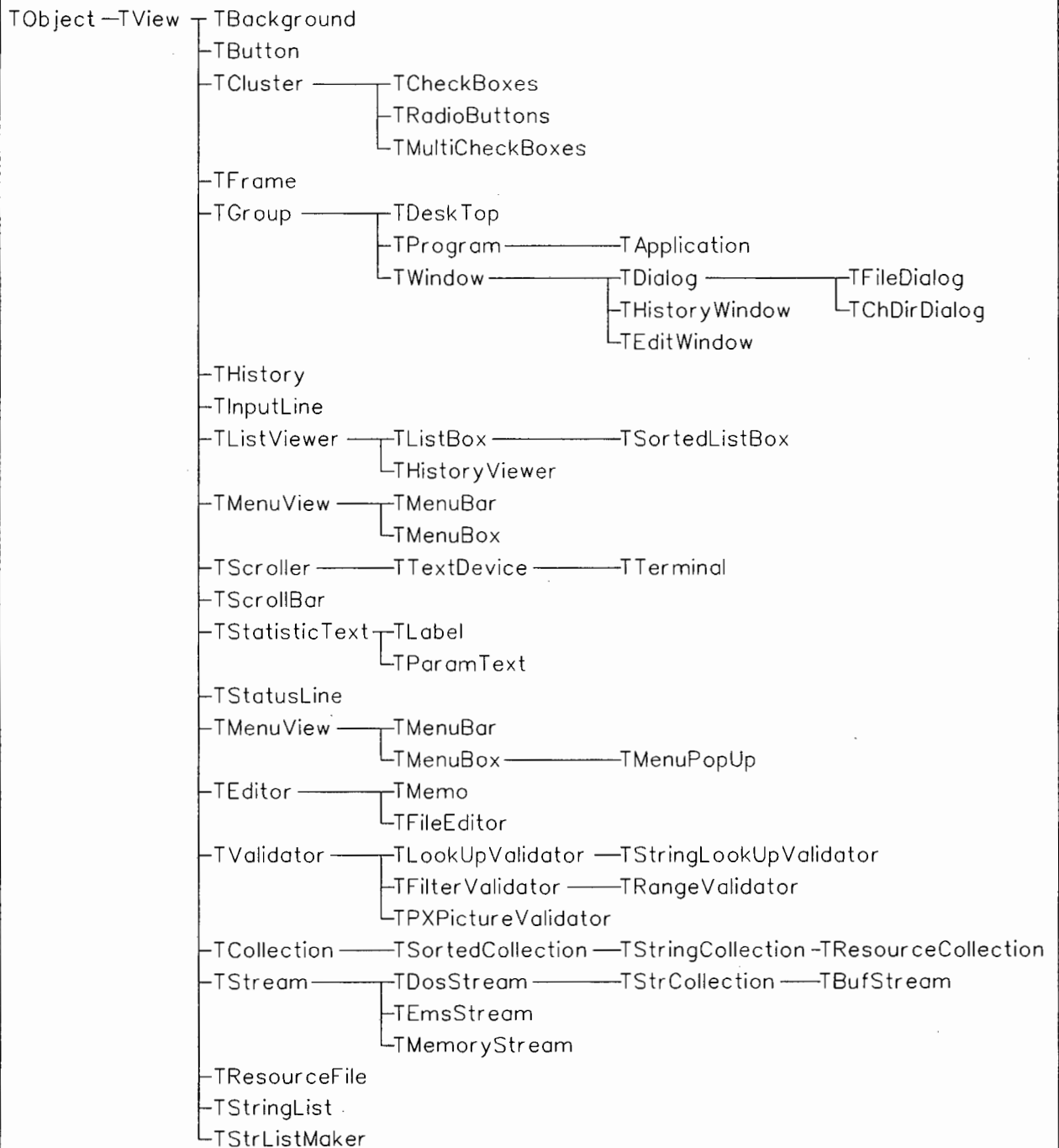


Figure 5.03

been created equal. Through their respective functions they can be divided into four distinct groups:

- Primitive objects
- Views
- Group Views
- Engines

Primitive objects exist primarily to be used by other objects or to act as the starting object of a hierarchy of more complex objects. These are TPoint, TRect and TObject. It is important to note that these objects are not displayable. TObject is the abstract base type with no fields and is the ancestor of all Turbo Vision objects except TPoint and TRect. TObjects descendants fall into one of two families, views and non-views.

The displayable descendants of TObject are known as views and are derived from TView which is an immediate descendent of TObjects. All view objects descended from TView have methods enabling them to be displayed in some way or another. The following standard views are included in Turbo Vision:

Frames	Input lines
Static text	
Buttons	List Viewers
Labels	
Clusters	Scrollers
Statuslines	
Menus	Scroll bars
Histories	Text devices

Group views are views that are derived from TView but are different in that they handle dynamically chained lists of related interacting subviews using a single designated view called the owner of the group. The object TGroup is the abstract ancestor of these groups with the following standard group views descended from it :

Applications	Windows
Desktops	Dialog Boxes

As a group is a view, there can be subviews that are in turn groups owning their own subviews and so on. The state of the chain constantly changes as the user clicks or types during an

application. New groups can also be created and subviews can be added and deleted from any group.

Engines are a special type of object that do not display themselves but are also derived from T Object. These groups are :

- Streams
- Resource files
- String lists
- Collections
- Validations

These objects are essentially concerned with the input, output and storage of data. They thus do not have to be visible in any way but merely interact with other groups and views in order to display or manipulate data. The full object hierarchy used in the program utility is shown in figure 5.04. It can be seen that almost the full Turbo Vision hierarchy tree has been used. The new objects derived from the Turbo Vision standard objects have been demarcated by the dotted line. The actual detailed structure of each new object created for this utility would be far too lengthy to present. Nevertheless it was decided for illustrative purposes to show the reader how a typical new object that is a descendant of a dialog box (TDialog) was derived.

In figure 5.05 one particular branch of the object structure hierarchy is presented, showing how the object TEditDialog2 was derived (as defined in the unit General2.pas). This figure clearly illustrates the topology of TEditDialog2. The left hand side of the figure shows the Turbo Vision standard objects, listing the object variables and methods of each new descendant. Each new object descendant inherits everything the ancestor object had plus whatever methods or variables are newly defined or overwritten. For example TDialog inherits all the characteristics of the TWindow object but overwrites the Init, Load, GetPalette and HandleEvent methods of TWindow and the Valid method of TView that was inherited up to that stage.

The newly defined TEditDialog1 inherits everything from TDialog but overwrites the Init, HandleEvent and Done methods. In turn TEditDialog2 inherits everything from TEditDialog1 except the Init method which it overwrites. A detailed description of each standard Turbo Vision object is contained in the Turbo Vision programming guide and will not be further dealt with here.

5.6.2.2 Program Flow Charts

The operation of the program utility is best depicted in the following series of flow chart representations (see figure 5.06 to figure 5.14, Appendix 6).

Program Object Hierarchy

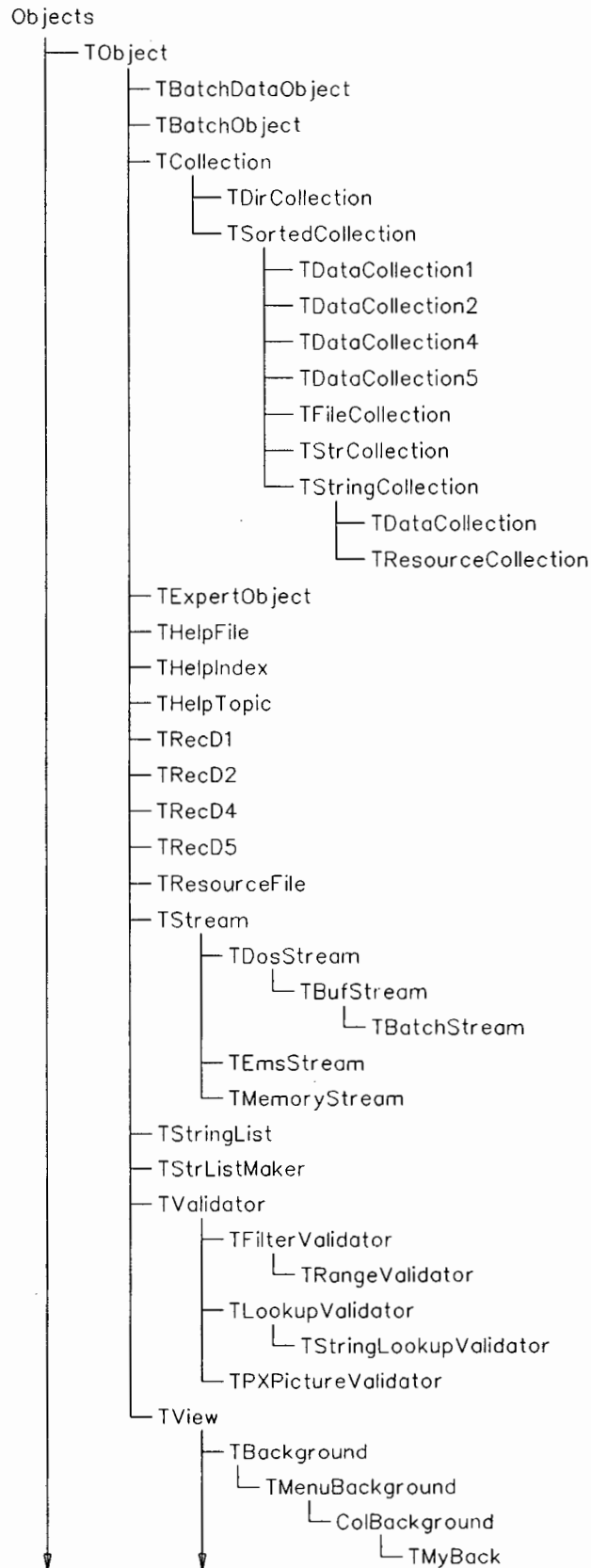


Figure 5.04
(sheet 1 of 3)

Program Object Hierarchy

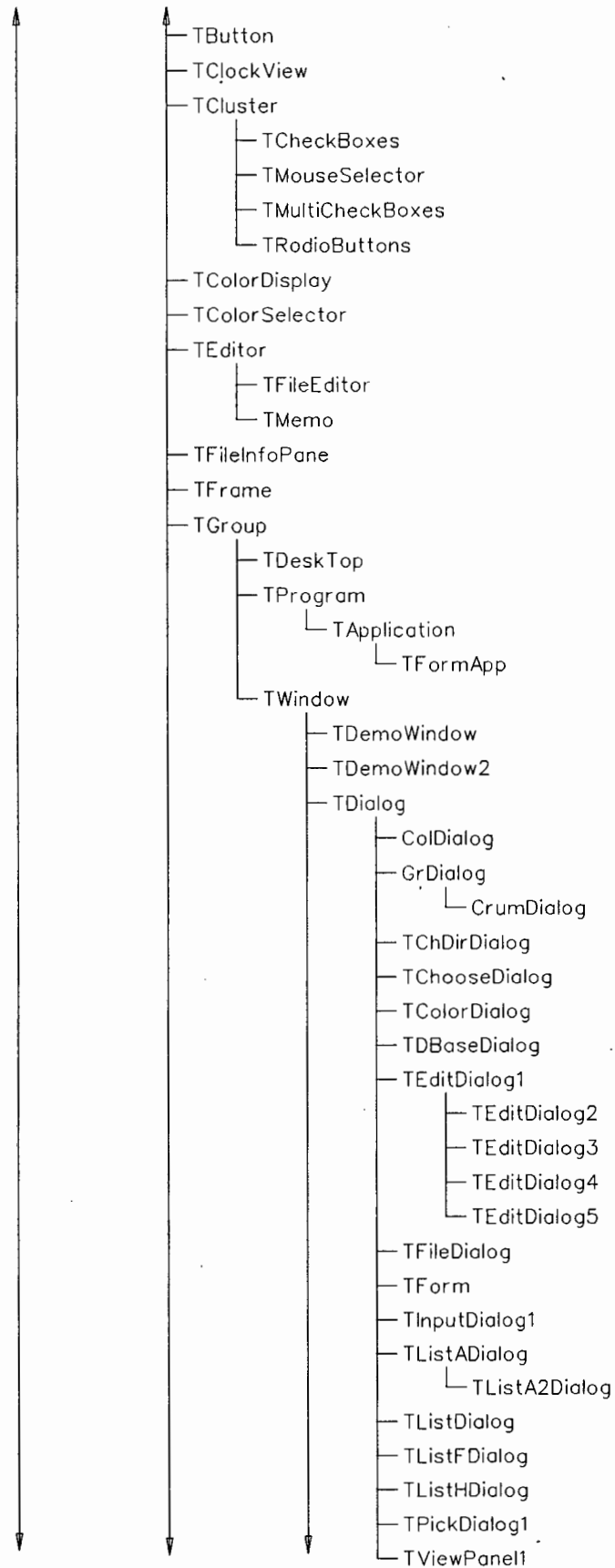


Figure 5.04
(sheet 2 of 3)

Program Object Hierarchy

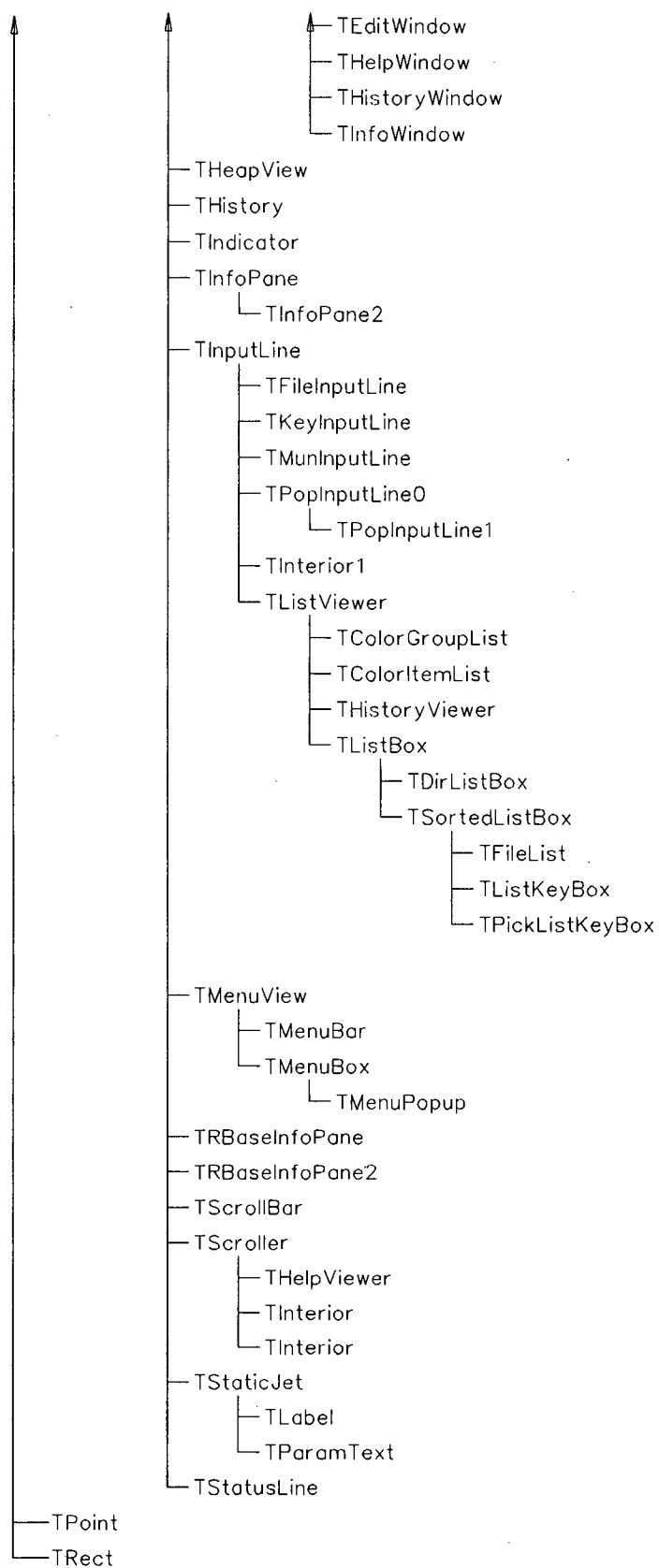


Figure 5.04
(sheet 3 of 3)

TVision Standard Units					Program Units		
Unit: Objects.pas	Unit: View.pas			Unit: Dialog.pas	Unit: General2.pas		
TObject	TView		TGroup	TWindow	TDialog	TEditDialog1	TEditDialog2
~~Init~~ ~~Free~~ ~~Done~~	Cursor DragMode EventMask GrowMode HelpCtx Next	Options Origin Owner Size State	Buffer Current Last Phase	Flags Frame Number Palette Title ZoomRect	~~Init~~ Load GetPalette ~~HandleEvent~~ Valid	~~Init~~ HandleEvent Done	Init
	~~Init~~ ~~Load~~ ~~Done~~ ~~Awaken~~ BlockCursor CalcBounds ~~ChangeBounds~~ ClearEvent CommandEnabled ~~DataSize~~ DisableCommands DragView ~~Draw~~ DrawView EnableCommands ~~EndModal~~ EventAvail ~~Execute~~ Exposed Focus GetBounds GetClipRect GetColour GetCommands ~~GetData~~ GetEvent GetExtent ~~GetHelpCtx~~ ~~GetPalette~~ GetPeerViewPtr GetState GrowTo ~~HandleEvent~~ Hide	HideCursor KeyEvent Locate MakeFirst MakeGlobal MakeLocal MouseEvent MouseInView MoveTo NextView NormalCursor Prev PrevView PutEvent PutInFrontOf PutPeerViewPtr Select SetBounds SetCommands SetCmdState SetCursor ~~SetData~~ ~~SetState~~ Show ShowCursor ~~SizeLimits~~ ~~Store~~ TopView ~~Valid~~ WriteBuf WriteChar WriteLine WriteStr	~~Init~~ ~~Load~~ ~~Done~~ Awaken ChangeBounds DataSize Delete Draw EndModal EventError ExecView Execute First FirstThat FocusNect ForEach GetData GetHelpCtx GetSubViewPtr ~~HandleEvent~~ Insert InsertBefore Lock PutSubViewPtr Redraw SelectNext SetData ~~SetState~~ ~~Store~~ Unlock ~~Valid~~	~~Init~~ ~~Load~~ Done Close ~~GetPalette~~ GetTitle ~~HandleEvent~~ InitFrame SetState SizeLimits StandardScrollBar Store Zoom			

LEGEND

Object Name

Object Variables

Object Methods

Name:

XXX

XXX
XXX

LEGEND

Object Name
Object Variables
Object Methods

Name:
XXX
XXX XXX

Object Structure of TEditDialog2

Figure 5.05

5.7 Data Storage Structures

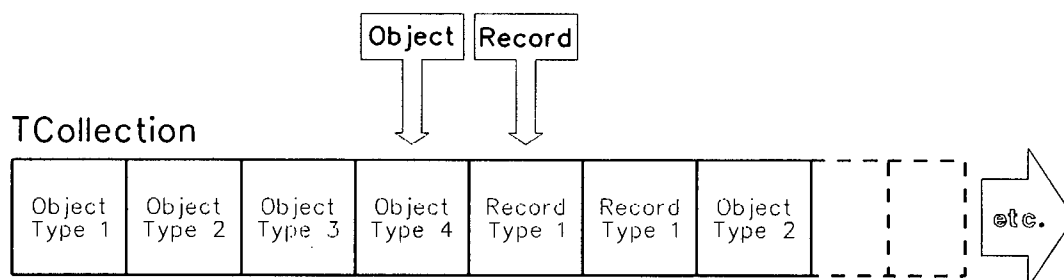
The data storage structures used in the program can be divided into three general groups. These are collections, streams and resource files.

5.7.1 Collections

Collections are objects that store lists of items including arbitrary objects and records of different types and provide many different methods for manipulating them. These collections are all derived from an ancestor type called `TCollection` and have certain features build into them including the capacity to be dynamically sized and polymorphic.

Collections have advantages over traditional Pascal arrays in that the maximum size is not fixed at compile time. An initial size is fixed but can then grow at run time to accommodate the data stored in it. The data stored in these collections is not limited to one particular type. A collection can consist of objects and even non-objects of any size and type. The collection does not know anything about the objects it is handling but merely holds onto them and gives them back when required. There are many descendants of `TCollection` including a `TSortedCollection`, `TStringCollection` and a `TResourceCollection`. The concept of a collection is depicted in figure 5.15

Collections are extensively used in this program. Typically data is read in from storage on the hard disc into a collection. This data is then manipulated using the collection and when it is finished it reads the collection back into the storage area on disc and then disposes of the collection from memory. Examples of data stored on collections can be seen in the list boxes or picklists when using the batchrun facilities in the program. The code for these collections can be seen in units such as `ListFDlg.pas`, `ListHDLg.pas`, `ListADlg.pas` and `ListDDlg.pas`.



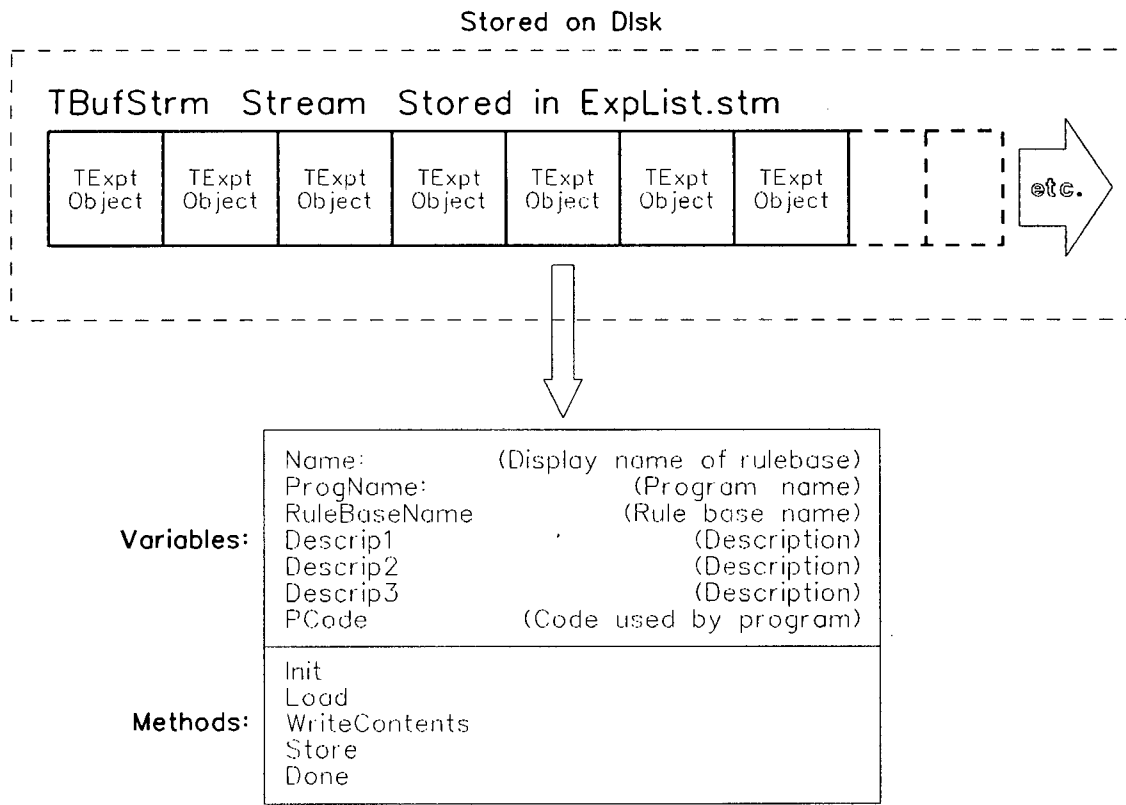
Graphical Depiction of a TCollection Object

Figure 5.15

5.7.2 Streams

The concept of object-orientated programming techniques gives programmers a very powerful way of encapsulating code and data (see section 5.2.1). However, when it comes to storing data the problem is that the data is bound together with the code in objects and would require an unravelling of code and data in order to store the data only. The mechanism by which data is stored in object-orientated programming is called streams. Streams are by definition collections of objects which are being sent somewhere. This could be a file on disc, a serial port or any other destination. Streams are objects that handle input/output on the object level rather than on the data level.

Like collections, streams are polymorphic in that any object type can be stored on a stream as long as it is a descendant of TObject. There are three different types of standard streams available in Turbo Vision which include TDOSStream, TEmsStream and TBufStream. All these objects are descendants of TStream. The program uses only TBufStream. This object implements a buffered version of TDOSStream and is significantly more efficient than TDOSStream when there are large amounts of small data transfers taking place. Only one stand-alone stream is implemented in the program and is used for storing a list of TExpertObject objects. These objects store the information corresponding to the expert system rulebases currently registered with the program. The stream is stored in a file called "ExptList.stm" and information is used from the stream whenever the rulebase data is required. The structure of this stream is depicted in figure 5.16.



Example of Structure

Graphical Depiction of the ExptList Stream

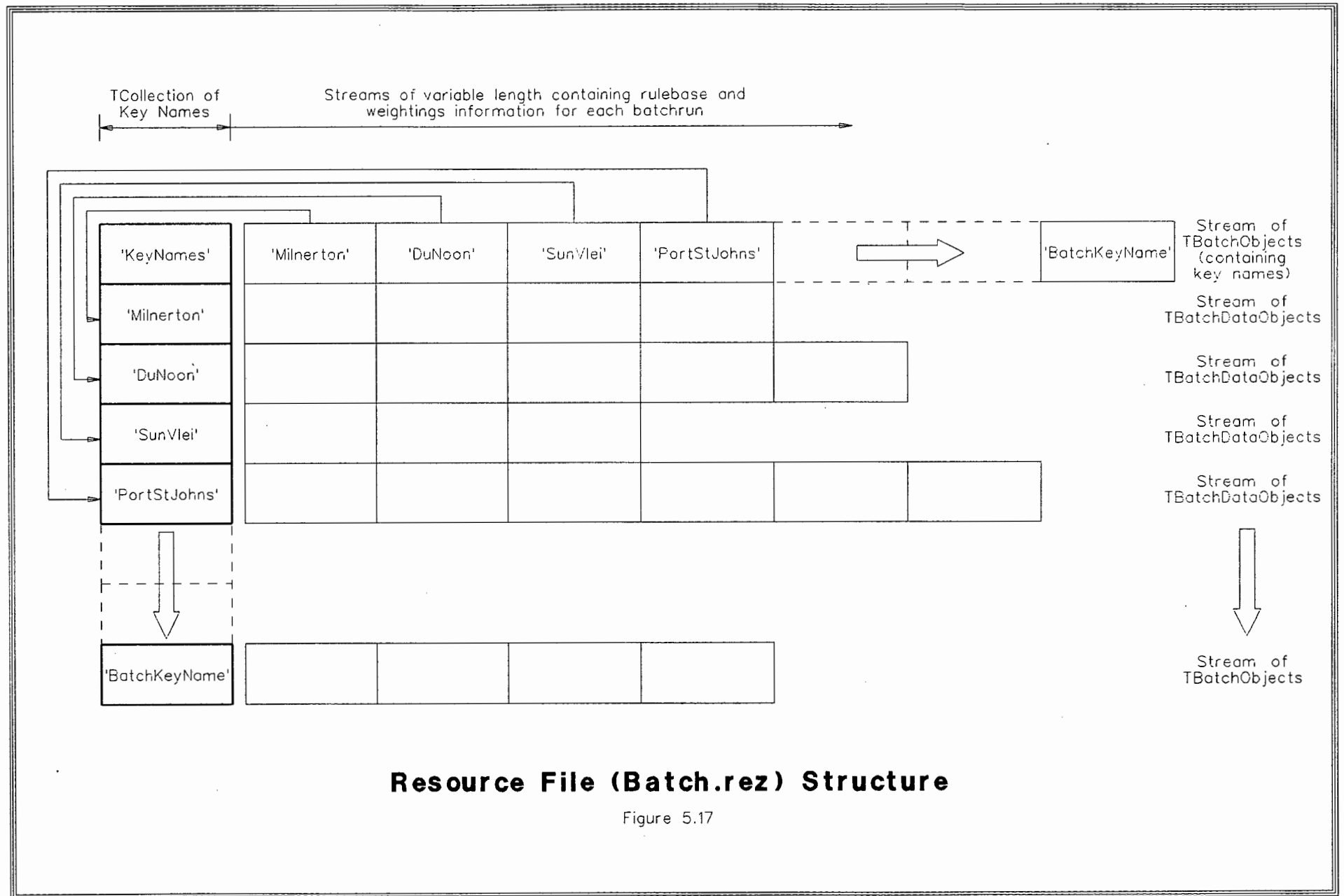
Figure 5.16

5.7.3 Resource File

A resource file is similar to a stream in that it saves objects handed to it but goes one step further in that it can retrieve the objects by name and in so doing acts like a random access stream. The object that implements this is the TResourceFile object and provides all the tools for manipulating the objects. The objects themselves are accessed by keys which are simply unique strings that identify the objects stored.

The program uses a resource file called "Batch.rez" for the implementation and storage of all the batch runs defined in the program. The structure of the resource file is set up to allow simple batch run information retrieval. This is done by using two objects especially created for the job; TBatchObject and TBatchDataObject. The first stream on the resource file is a stream of batch names of type TBatchObject. When the user wishes to use or edit a batch run the first stream object is read in containing the list of batch names. From this list the user selects a batchrun and the program then retrieves the stream object from the resource file using the key contained in the TBatchObject selected. The means by which this is done is that each TBatchObject has a BatchCallName variable which is the key to retrieving that stream.

Each BatchRun stream consists of a list of objects of type TBatchDataObject. Each one of these objects defines a rulebase to be used in the batch run analysis together with the weighting that will be used in the final result analysis (see figure 5.17).



5.8 Rulebase and program interfaces

SYNAPSE uses two main methods of interfacing with other programs. These are by using two types of ASCII files called answer files and result files.

An answer file is an optional file with which a SYNAPSE session can be started and which provides all the answers to the questions contained in the rulebase. This file is a standard ASCII file that has been created using a program or text editor of any sort. If an answer file is supplied, SYNAPSE behaves as though all the answers have been supplied as the session is begun, the user is not prompted for any further information during the analysis. The format for using the answer file parameter is:

```
SYNAPSE -f rulebasename -a filename/d
```

The format of the answer file can be of two types. A plain ASCII file as shown in figure 5.18 (a) and a delimited ASCII file as shown in figure 5.18 (b). The first column of both is the question name and the second column is the answer supplied to that question

```
exp3      1
support2  1
exp5      1
support4  1
prob1     3
prob2     1
prob3     2
prob4     1
econ5     1
econ4     3
econ7     5
econ6     4
econ3     5
econ8     9
exp7      3
exp2      5
exp4      3
support3  3
exp6      1
support5  2
exp1      1
```

Figure 5.18(a) - Answer File - Plain ASCII Format

```
"exp3", 1
"support2", 3
"exp5", 1
"support4", 2
"prob1", 4
"prob2", 3
"prob3", 1
"prob4", 2
"econ5", 3
"econ4", 2
"econ7", 2
"econ6", 1
"econ1", 1
"econ3", 1
"econ2", 2
"econ8", 1
"exp7", 1
"exp2", 2
"exp4", 3
"support3", 1
"exp6", 2
"support5", 1
"exp1", 2
```

Figure 5.18(b) - Answer File - Delimited ASCII Format

APPENDIX 1

SYNAPSE Mathematics

A results file is an ASCII file in which the final results of a SYNAPSE consultation are stored. The format for using the results file is

```
SYNAPSE -f rulebasename -r filename/d
```

This file is similar to that of the answer file in that two formats can be used, that of plain ASCII files or delimited ASCII files. Examples of these are shown in figure 5.19(a) and 5.19(b).

```
problem type      -0.562 -0.562 -0.562
economics         -0.376 0.000 0.376
use and support   0.600 0.900 1.200
expertise         0.856 0.856 0.856
problem type goal -0.562 -0.562 -0.562
economics goal    -0.376 0.000 0.376
use and support goal 0.600 0.900 1.000
expertise goal    0.856 0.856 0.856
score            0.129 0.290 0.417
definitely       -1.000 -1.000 -1.000
```

Figure 5.19(a) - Result File - Plain ASCII Format

```
"problem type", 0.100, 0.100, 0.100
"economics", 1.034, 1.034, 1.034
"use and support", -0.150, 0.150, 0.450
"expertise", 0.963, 0.963, 0.963
"problem type goal", 0.100, 0.100, 0.100
"economics goal", 1.000, 1.000, 1.000
"use and support goal", -0.150, 0.150, 0.450
"expertise goal", 0.963, 0.963, 0.963
"score", 0.500, 0.575, 0.650
"definitely", 1.000, 1.000, 1.000
```

Figure 5.19(b) - Result File - Delimited ASCII Format

The structure of these files is slightly different from that of the answer file. The first column contains the name of the goals or subgoals and the three columns of numbers following correspond to the lower, expected and upper RVL values for that goal or subgoal. The results of the consultation are stored in this file which can then be used at a later stage for further result processing (see section 5.10).

5.9 Databases

The program contains a menu item called "Databases" which is a facility for accessing data bases that are used by the expert system rulebases. SYNAPSE has a very limited capacity to access information stored in databases, however more modern expert system shells use databases extensively. For this reason it was decided to include this facility for completeness using a few example databases listed.

SYNAPSE was shipped with a capacity to access information stored on dBase 3 data files only. This was a database program extensively used until recently but has subsequently been replaced by more sophisticated relational databases and other products. The way in which SYNAPSE makes an enquiry on a database to find out the answer to a question is by using a Numeric rule. A typical example would be :

```
NUMERIC salary.  
TRUE {...}  
FALSE {...}  
HELP {...}  
DBASE synex.dbf synex.ndx {JONES} {SALARY}
```

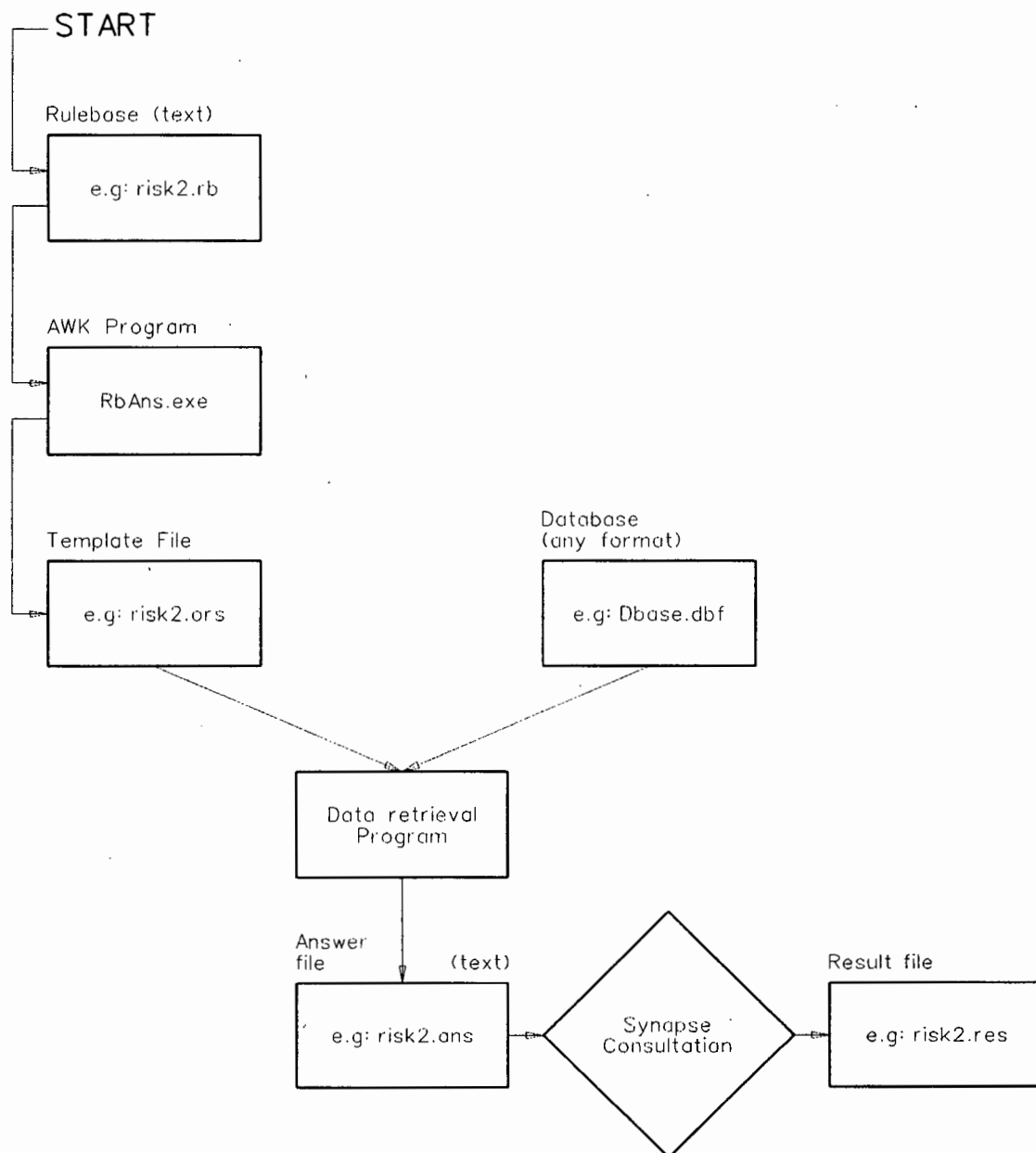
When SYNAPSE wishes to retrieve the numerical value of the question 'salary' it uses the dBase file synex.dbf and the corresponding index file synex.ndx and finds the record with the name 'JONES' and the field with the name 'SALARY' and retrieves the value.

A further extension of this theme would be to integrate SYNAPSE into dBase. Here the entire system can be built into dBase and the 'RUN' or '!' command can be used to run SYNAPSE from within dBase at the appropriate time. This feature is not used for the purposes of the utility developed and any further information can be obtained from the SYNAPSE user manual.

A third possibility exists. This involves using a third party program that retrieves the information for the expert system before the rulebase is run. Using this method a program could specifically be created to extract the data needed from whatever data base is used, placing it in an answer file in a format that can be used by the expert system at run time. An example program was developed for just such a purpose. The program called rbans.exe was developed using the language AWK (a pattern matching language). This program scans a rulebase file and prepares a '.ors' file that specifies all the information that will be required by the rulebase. This '.ors' file is essentially a template that can be used by a second purpose-written program to retrieve exactly the data specified by the '.ors' file from the data base. A flow chart of this operation is shown in figure 5.20 and the program listings for this facility are contained in Appendix 2.

It must be stressed that the interfacing of expert systems with databases is highly dependent on the capabilities built into both the expert system in terms of the various database products and their formats by which to retrieve data, and the databases themselves in terms of the extent to which they make data available to external programs. Users of this facility may not only use their own expert system program/rulebases but the databases that these expert system shells may be interfaced to, allowing as much flexibility as possible.

Although the feature of retrieving data from databases has been included, work was not continued to develop it to its full potential. It was felt that this aspect should be addressed when a more up to date expert system shell was found (see recommendations - chapter 9).



Flow Chart of Rulebase Information Retrieval for any Database Format

Figure 5.20

5.10 Manipulation of Result Data

One of the most important aspects of the processing of the result data produced from SYNAPSE consultations is the naming convention used to identify the result files produced. The DOS naming convention requires an eight character filename followed by a full stop and a three letter file extension, indicating to the user what type of file it is. The most commonly used file extensions used with SYNAPSE are the '.ans' (answer file) and the '.res' (result file), described in section 5.8.

The program utility maintains these file extension conventions but creates its own file name for identification. This name is made up as follows :

The first 4 characters correspond to the first 4 letters in the batchrun name being analysed.

Characters 5 and 6 correspond to the final 2 characters in the batchrun name being analysed.

Characters 7 and 8 correspond to an internal rulebase code of the rulebase used to produce that file.

For example if a hypothetical batchrun name of 'Clifton' was used and the internal rulebase code for the rulebase being run is 12 then the filename of the resultfile produced would be 'clifon12.res'. (ie 'clif' + 'on' + '12' + '.res')

Another important aspect of the processing of result data is the correct information that is to be extracted from the result files. All rulebases have been setup with the GOAL in each being called 'Assessment'. The result files are produced with the lower, expected and upper RVL values recorded in the files in the format described in section 5.8. In the processing of the results files, the system automatically seeks the data relating to the GOAL 'assessment' and extracts the three RVL values which are then stored.

These values are then adjusted according to the weightings recorded in the batchrun stream as described in section 5.7.3 and are then presented to the user in graphical form. Methods by which RVL values are adjusted are controversial. However the following 'points system' method is used. The maximum number of points for each rulebase is worked out by multiplying the number of rulebases by 100 and then by the percentage weighting allocated to that rulebase. In the example below this would be as follows :

$$\text{No. rulebases} \times 100 = 500$$

500 x 12/100 =	60
500 x 24/100 =	120
500 x 23/100 =	115
500 x 10/100 =	50
500 x 31/100 =	<u>155</u>
Total	500

The RVL values retrieved from each result file are then converted to a percentage figure through a simple mapping equation as follows :

(+0.000 + 1) / 2 =	0.50
(-0.120 + 1) / 2 =	0.44
(+0.280 + 1) / 2 =	0.64
(+0.213 + 1) / 2 =	0.61
(+0.077 + 1) / 2 =	0.54

These figures are then multiplied by their respective number of points to produce a result out of 500. This figure is then converted back to a RVL value for presentation in the output.

0.50 x 60 =	30.0	Note : figures have been rounded off.
0.44 x 120 =	50.6	
0.64 x 115 =	72.8	
0.61 x 50 =	32.0	
0.54 x 155 =	<u>83.5</u>	
Total	<u>269</u>	

$$\text{RVL value} = 269 / 500 \times 2^{-1} = +0.075$$

This processing of result data is best illustrated using an example as shown in figure 5.21.

CHAPTER 6

AN EIA WORKED EXAMPLE

*"Any sufficiently advanced technology
is indistinguishable from magic"*

Arthur C Clarke

Contents

- 6.1 Introduction**
- 6.2 Environmental Assessment of Park Rynie Sewage Treatment Works**
- 6.3 Summary of Report Assessment**
- 6.4 Site Assessment Using the Expert System Utility**
- 6.5 Comments on Results of Utilities Assessment**
- 6.6 Comparison of the Computer Analysis With the Consulting Engineer's Analysis.**
 - 6.6.1 Overall Weighted Assessment**
 - 6.6.2 Comparison of Each Characteristic**
 - 6.6.3 Other Aspects of the Program**

AN EIA WORKED EXAMPLE

6.1 Introduction

Although several environmental assessment projects were used for experimentation during the development of the expert system utility, it was decided to present one of these as an example to demonstrate the potential and capabilities of the system. It must be stressed that the results of the evaluation by the consulting engineer had a major influence in the recommended site. The writer believes that definite opinions regarding the usefulness of the system can only emerge after frequent usage and evaluation by specialists in the various environmental characteristics.

The example chosen was based on a report for a sewage works for Park Rynie on the Natal south coast(18). This example was chosen because it involved not only an environmental impact analysis but also the assessment of alternative sites. Furthermore it was conducted using a quantitative method as opposed to a qualitative analysis used for the program utility thus highlighting the differences between the methods.

Sections 2 and 3 summarise the consulting engineer's analysis and its results. Section 4 presents the analysis using the program utility developed in this thesis and conducted by the same consulting engineer who wrote the original report and finally in section 5 the results of the two analyses are compared. It must be stressed that this utility is not intended to perform a complete environmental impact assessment, but acts as a preliminary assessment tool to direct attention to the features of the Environmental Impact Assessment and how they influence each alternative site.

6.2 Environmental Impact Assessment of Park Rynie Sewage Treatment Works.

This report was to identify and evaluate potential sites for the location of the proposed works. This was done in terms of the effects of the treatment works on selected environmental characteristics. A diagram illustrating the region and its land usage is shown in figure 6.01. In order that the capabilities of the program utility may be better appreciated a summary of the basic criteria of service with respect to features which would affect the environment is summarised as follows :

- Engineering

The engineering requirements for a scheme of this nature are :

Conveyance of Sewage - sewage should ideally flow by gravity from the contributors to the works. This avoids pumping which is expensive and requires reliable operation. Pumping should be kept to a minimum of elevation above the pumping station.

Sewage Treatment Works - Once sewage arrives at the works it should be biologically processed to produce effluent of required standards.

- Economics

The works should be situated at a low level to maintain gravity feed and thereby avoid or reduce the costs of pumping because the costs of electricity is high and will increase every year. Any increase in costs will affect the viability of the scheme with respect to income of ratepayers and the additional tariffs that they would be expected to pay to meet the capital and operating costs.

- Principals of Regional Development

Sewage schemes are usually developed in stages according to the limits that can be afforded by the residents and the growth of the population contributing to the effluent flow to the works. Allowance must be made for future development of a regional works so an adequate buffer area around the works must be provided for this purpose.

- Agriculture

Good agricultural land should be maintained for agricultural purposes. Land lower down in the valley is generally more fertile which conflicts with the principles of establishing a treatment works at a low level. There is, however the possibility of using the treated effluent for irrigation although strict health regulations will have to be observed if this is done.

- Social

Most communities object strongly to the presence of sewage treatment works which they believe will affect them in terms of sight, odour and health. It is therefore preferable to establish the works in an area that will not have an effect on existing

communities, usually outside township boundaries. For the above reasons sewage works would prevent growth of townships in their vicinity.

- Health

The effluent must at all times meet standards acceptable to the Department of State Health. In the event of possible malfunctions at the works, adequate measures should be provided to manage effluent until normal operation of the works is restored. Consideration must be given to the effluent passing through a caravan park and entering the sea within the tidal zone because of the serious affect which it could have on filter feeders such as mussels which are eaten by the public.

- Nature Conservation

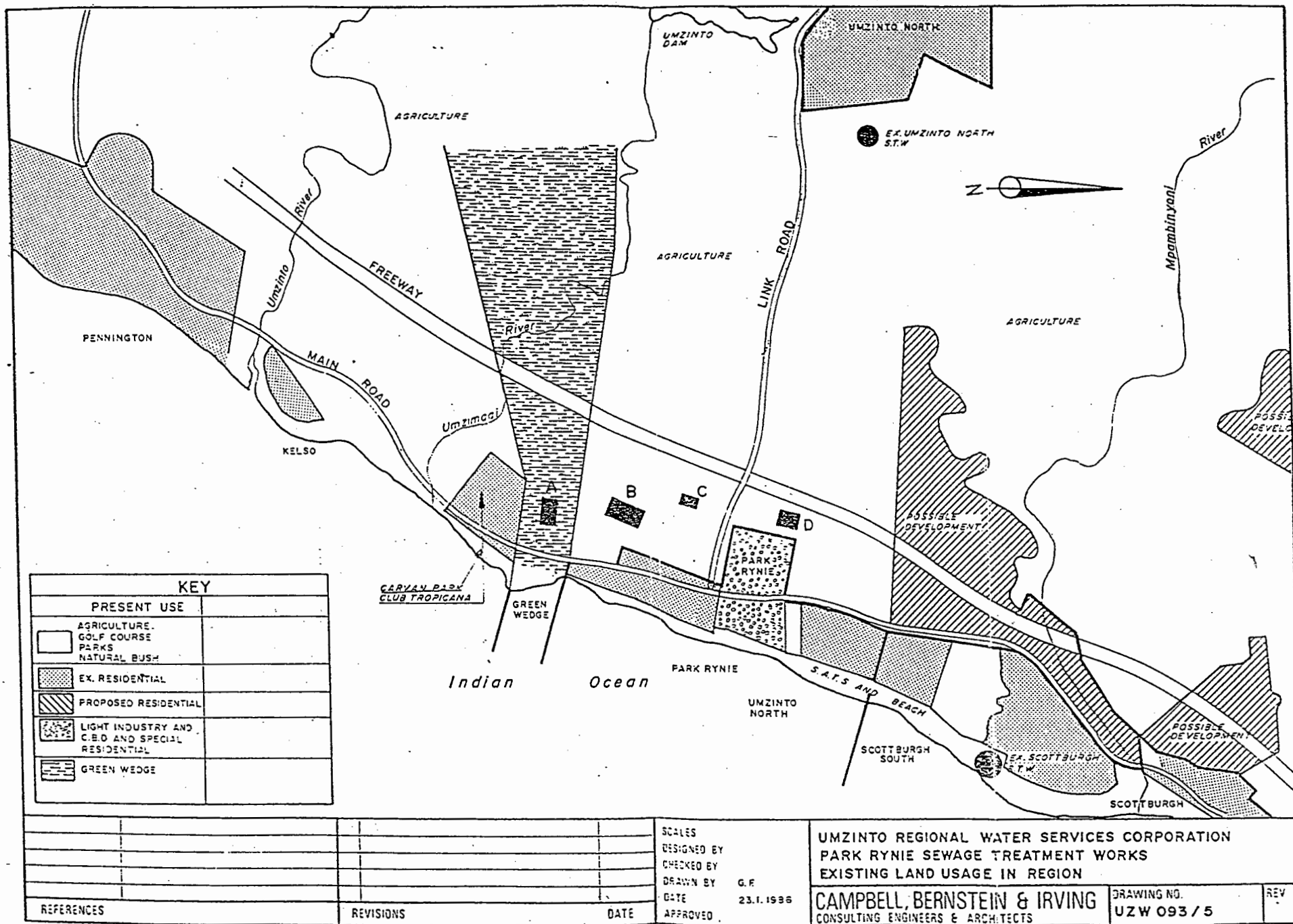
After treatment, effluent is passed through ponds. This attracts bird life, especially water fowl. Concrete tanks are usually recessed into the ground and thus blend into the natural surroundings. Works themselves (not their effluent) generally have a minimal physical impact on surrounding natural communities.

- Recreation

In this characteristic the most beneficial amenity is the beach zone and it is important that the works do not impact negatively on this in any way whatsoever.

Four potential sites were selected by the consultant that would broadly meet the criteria of service discussed above. These were numbered Site A to Site D and are shown in figure 6.02. The report evaluates each site with regard to criteria of service and possible impacts on the environment.

It should be noted that the E.I.A. procedure does not include Engineering and economical criteria in the comparison of the alternative sites although this was included by the consulting engineer because the works was intended to serve a low income population.



6.3 Summary of Report Evaluation

The method used in the report is a quantitative form whereby each environmental characteristic was given a value for each site by the consulting engineer relative to the weighted value of the particular characteristic. These are shown on page 6-08. These values were then summated for each site and compared. The site with the highest figure, according to this system was the most favourable.

It is of particular interest to note that the system described briefly in the previous paragraph is described in section 4.5(d). The criticisms submitted in section 4.5(d) on quantitative methodologies is discussed later in this chapter.

The assessments on Figure 6.03 had been presented after approval by the town and regional planning authorities and provincial engineers.

UMZINTO REGIONAL WATER SERVICES CORPORATION
PARK RYNIE SEWAGE TREATMENT WORKS
ENVIRONMENTAL IMPACT ASSESSMENT

EVALUATION OF SITES - ANNEXURE "A"

ALLOCATION OF VALUES TO SITES

SITE		A	B	B1	C	C1	D	REMARKS
FEATURE	VALUE							
ENGINEERING - TREATMENT	15	15	15	15	9	9	15	Treatment to standards
CONVEYANCE	10	10	8	8	4	4	Nil	Gravity or pumping
ECONOMICS	15	15	11	11	7	7	2	Importance of cost to the Community, i.e. Income of Ratepayers
REGIONAL PLANNING	15	5	10	10	10	10	7	Effect on Planning of Region, i.e. Value of land usage.
AGRICULTURE	5	Nil	1	1	5	5	3	Benefit to agricultural usage - crop productivity
SOCIAL	10	6	6	6	8	8	4	Comfort to local residents
HEALTH	10	8	1	8	1	8	1	Threat to health relative to breakdown of process.
NATURE CONSERVATION	10	8	8	8	4	4	8	Adjacent areas and indigenous bush.
RECREATION	10	8	6	8	6	8	4	Disturbance of recreation amenities, eg. beach.
TOTAL	100	75	66	75	54	63	44	

NOTES: 1. For Sites B1 and C1 the treated effluent from these sites will be discharged into the stream below Site A.

2. The Regional Planning feature for Site A has a very low allocation to include the rejection of this site by the Environmental Planning Division of the NPA.

6.4 Site Assessment Using the Expert System Utility.

In order to conduct the comparison as effectively as possible help from the consulting engineer who originally compiled the report was enlisted. It was decided that an analysis would be done of Site A and Site C as these had the most widely varying characteristics. Both batchruns were assembled using identical weightings and environmental characteristics as shown in figures 6.04 and 6.05. The relevant environmental characteristics include characteristics additional to the consulting engineer's report and were identified as follows:

- Ecological characteristics of the site and its surroundings
- Health and safety
- Current and potential land use and landscape character for agricultural areas.
- Physical characteristics of the site in relation to land, climate, marine and estuarine systems.
- All aspects of social and community services.
- Nature and level of present and future water pollution.

The weightings for each were decided by the consulting engineer as shown in figures 6.04 and 6.05. It must be noted that although figures 6.01 and 6.02 show site A in a green wedge this was in fact an existing agricultural area and the zoning for a green wedge was later removed.

These weightings have not been scrutinised by any other authority such as town and regional planning and relevant specialists in each of the environmental characteristics. The batchruns for site's A and C were run with the consulting engineer providing the answers to the expert system questions. The results of these runs are shown in figures 6.06 and 6.07. It is noted that the characteristics for Engineering, Conveyance and Economics were not included in the comparison of the two systems of analysis.



Figure 6.04 - Batchrun for Site A

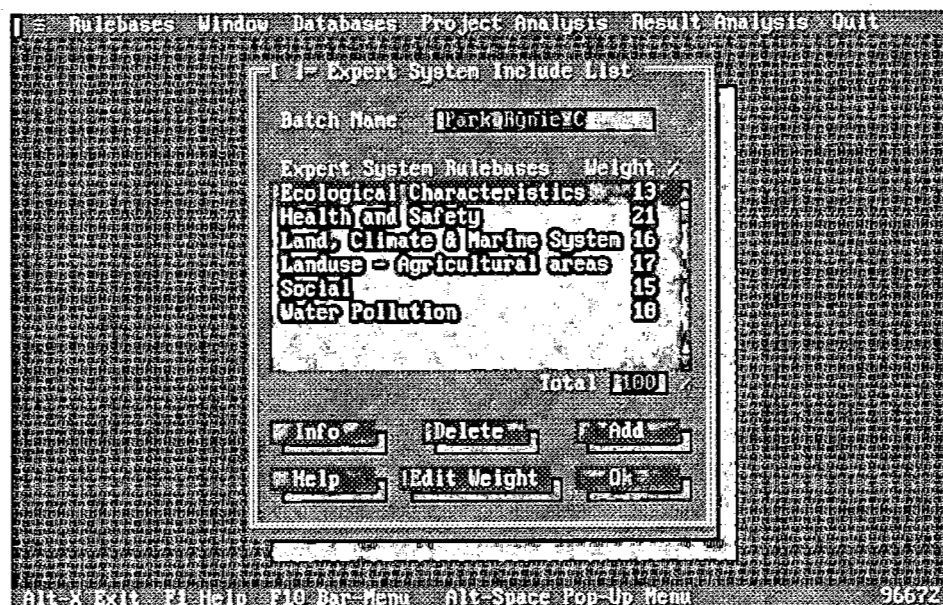


Figure 6.05 - Batchrun for Site C

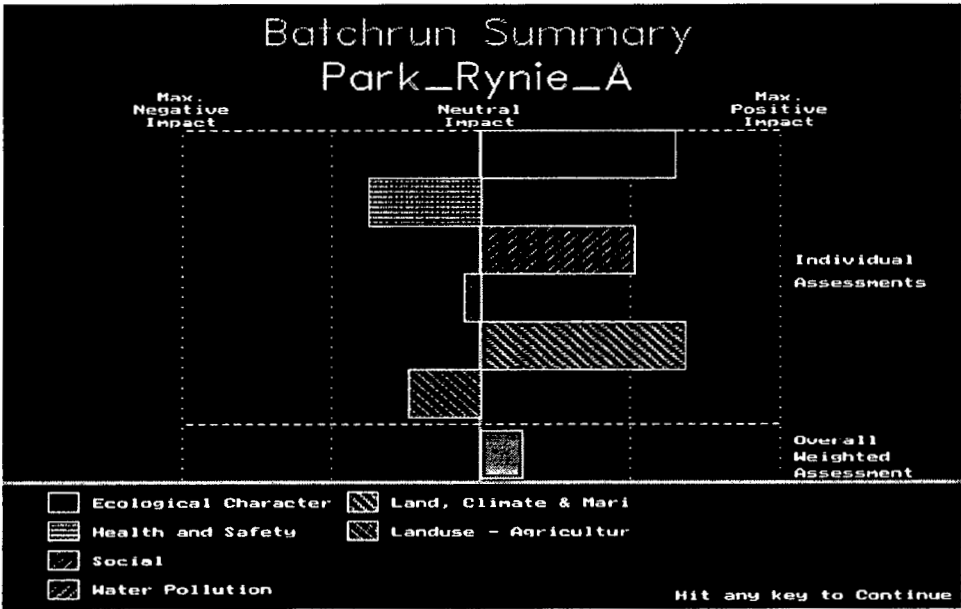


Figure 6.06 - Analysis of Site A

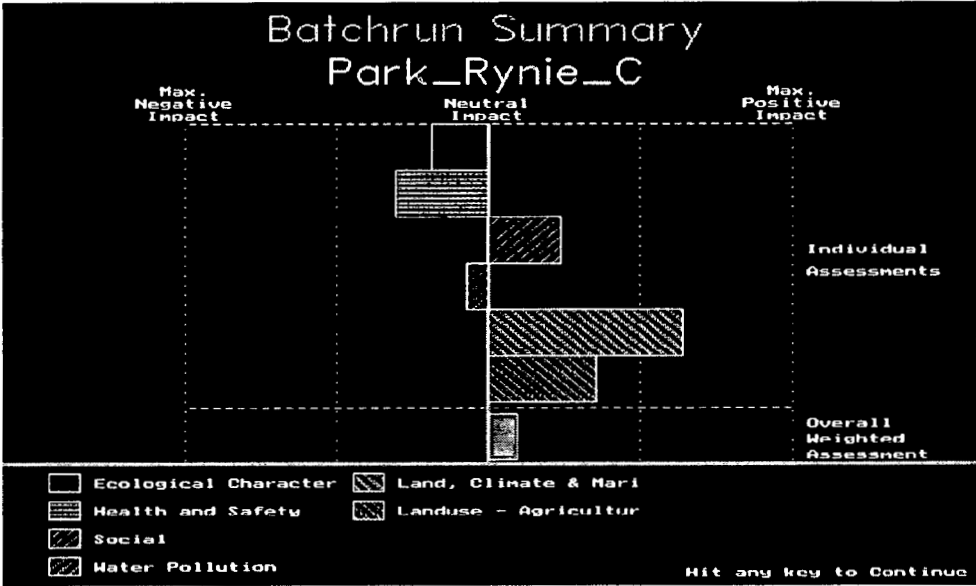


Figure 6.07 - Analysis of Site C

6.5 Comments on Results of Expert System Utility Assessment

The results of the assessment as shown in figures 6.06 and 6.07. Commentary on these results is summarised as follows :

Site A The three negative environmental impacts for this site were health, water pollution and agricultural land use. The site would discharge into the river that flows through reedbeds and through recreational facilities. This could cause potential public health problems as well as possible water pollution problems. In addition the site is situated on fertile agricultural land resulting in the permanent loss of this productive land.

However, positive impacts were also apparent. Of note was the positive impact of the ecological characteristic. This was in line with the fact that sewage treatment works attract considerable bird and fish life to the maturation ponds on the site thus re-establishing favourable habitats destroyed by agricultural activities. The social impacts are also favourable in that the works is situated far from residential areas.

Site C Site C as opposed to site A is situated in indigenous bush. Consequently the analysis of this site showed that the ecological characteristic was negatively impacted as a result of loss of virgin vegetation and natural habitat for wildlife.

The health and water pollution characteristics were very similar to site A as the effluent was also discharged into the waterway which flowed through the caravan park and recreational facilities. The agricultural land use characteristic impacted favourably as there was no affect or loss of agricultural land. The result of the social characteristic was the same as site A as both were situated far for residential areas.

The expert system analysis differed from the report in that three factors of engineering, conveyance and economics were not included. It was assumed for basic comparison with respect to the environmental analysis that both sites could operate at the correct engineering standards and that financial implications were not allowed to influence the assessment. Obviously these other factors must be considered in projects before any final decisions are made on implementation especially where payment for the project was by low income residents.

It was evident from the start of testing the system that the results required careful interpretation. Although the results are depicted graphically in a type of 'quantitative' form they

need to be interpreted using a qualitative reasoning process. This process is aided by the use of the expert system which displays its reasoning process, and so helps to identify those causes that have the greatest impact on the overall project.

6.6 Comparison of the Computer Analysis With the Consulting Engineer's Analysis

6.6.1 Overall Weighted Assessment

In order to compare the results of the two systems it is necessary to adjust the values of Figure 6.03 on page 6.08 of the quantitative analysis. The values for Engineering, Conveyance and Economics were deducted from total values of site A and site C to present the following :

Total for site A 35 of 60 = 58 %

Total for site C 34 of 60 = 56 %

ie. site A is slightly better than site C.

The remaining characteristic viz. Regional Planning, Agriculture, Social, Health, Nature Conservation and Recreation are not the same as the characteristics evaluated by the computer analysis viz. Ecological Characteristics, Health and Safety, Land Climate and Marine System, Landuse- agricultural areas, Social and Water Pollution.

Comparison of the Overall Weighted Assessments for the Computer Analysis shows this value to be slightly greater for site A (+ 14%) compared to that of site C(+9%) ie they are both slightly favourable with site A better than site C.

In section 4.5(d) of this thesis various criticisms are presented. One aspect is clearly displayed by using the programme. Referring to Figures 6.06 and 6.07 it is readily seen that the numerical values used in the Quantative system of analysis do not clearly show or depict the negative values which form the basis of illustrating the computer program method. This is important in assessing each characteristic. Positive and negative values have greater meaning in interpretation of any "quality" system.

Care should be taken when considering the "Overall Weighted Assessment" as this does not take into consideration the importance of a single characteristic which could be of such importance that its assessment could make it unacceptable to the community .

6.6.2 Comparison of Each Characteristic

Ecological Character

Computer program (Figs 6.06 and 6.07)

Site A is presented to be good for this character whereas site C is seen to have a slight negative influence on ecology.

Quantitative Method (Fig 6.03)

Site A is shown to be good (8 out of 10) for ecology ie. it should enhance this characteristic but site C is presented as being below "average" (4 out of 10).

Comparison

Both systems indicate very similar values for ecology.

Health and Safety

Computer Program (Figs 6.06 and 6.07)

Site A will have a fairly bad influence on health and Site C will have a slightly bad influence on health.

Quantitative Method (Fig 6.03)

Site A is valued at 8 out of 10 ie a good effect on health whereas site C is rated as having a bad effect on health.

Comparison

There is no correlation between these two methods of analysis for site A. The reason for this is that after the presentation of the E.I.A. report it was considered a "future" pipeline would be required from both of these works to the sea to ensure good health conditions through the beach and caravan site (site A) and through the two caravan sites for site C The criteria presented for the Computer Program was the same for both sites viz. a pipeline into the sea which is a possible threat to health in the early stage when the effluent flows

over land. In the latter part of the operation of the works there will be a slight threat to health as a result of possible accumulation of bacteria by "filter feeders" such as mussels.

Social

Computer Program (Figs 6.06 and 6.07)

Site A displays a fairly good effect on social criteria and Site C presents a slightly good effect on social criteria.

Quantitative Method (Fig 6.03)

Site A indicates the effect on social criteria will be slight (6 out of 10) and site C shows it will be good (8 out of 10).

Comparison

The correlation is fairly good except for Site C for the Quantitative Method. It is possible that this discrepancy could be caused by the Consulting Engineer because a period of some eight years has lapsed since he last worked on the project (this was considered to be the reason by him). The degree of error here could also be considered to be within the limit of human error.

Water Pollution

Computer Program

Both site A and site C indicate very slight pollution which is the condition which is reasonable to expect in waterways below a sewage treatment works.

This characteristic was not considered in the analysis by the Quantitative Method.

Land, Climate and Marine

Computer Program

Site A and Site C show a fairly good effect on these characteristics which could be expected to be the result of the introduction of a sewage treatment works replacing the existing system of septic tanks and soakage systems for disposal of sewage.

This characteristic was not considered in the analysis by the Quantative Method.

Landuse - Agriculture

Computer Program

Site A will have a fairly bad effect on agricultural land whereas site C will be fairly good effect on agricultural land.

Quantitative Method

Site A will have a very bad effect (0 out of 5) on agricultural land and site C will have a very good effect (5 out of 5) on this characteristic.

Comparison

The above results correspond well.

6.6.3 Other Aspects of the Program

The evaluator has to play a totally impartial and unbiased role in the operation of the program for the results to be reliable. There is a need for identification of the relevant environmental characteristics and their appropriate weightings by specialists. Furthermore, rulebases can be further subdivided so as to concentrate on smaller domains and so provide enhanced accuracy.

In order to attain a greater degree of reliability it would be possible to repeat "runs" by evaluators. An important aspect of Environmental Impact Assessment is participation by the public or residents in an affected area. The use of a computer programme presents the possibility of obtaining more reliable opinions in a confidential manner from the public for the alternatives being considered.

In this regard such a survey could be used to present reliable weighting factors which could be representative of the opinion of the communities who experience the impacts. This could be more acceptable to the community than the selection of weightings by the operator/analyst or even a specialist, provided that a truly representative sample of the public is utilised in the survey. It is essential that the results of the analyses from the use of the programme are examined to confirm their reliability.

If the latter proves to be an important consideration then the programme as developed in this thesis could also be used in the final analysis of the Environmental Impact Assessment.

CHAPTER 7

CONCLUSIONS

*"Some of his decisions were accurate. A
stopped watch is right twice a day."*

Anonymous

Contents

- 7.1 Literature Survey of Expert System Technology
- 7.2 Environmental Impact Assessment and the Rulebase Development
- 7.3 Development of the Program Utility
- 7.4 Operational Review

CONCLUSIONS

7.1 Literature Survey of Expert System Technology

In the initial literature survey carried out for this thesis the single most important aspect that was evident to the writer was the disappointing lack of understanding of expert system technology, especially its capabilities and limits. This was further evident in discussion with interested parties where there is much disillusionment as a result of expert systems being used for inappropriate applications. For this reason a lot of time was spent in this thesis discussing the architecture and most importantly the problem types best suited to expert systems. It was noted that expert systems were not intended to solve conventional algorithmic problems but are confined to problems that humans can solve through symbolic reasoning.

An important problem that was identified was that of the "engineering bottle-neck". This is a term used to describe the huge impeding factor caused as a result of the substantial time, effort and cost associated with developing an expert system application by skilled individuals. This bottleneck is a major hindrance to the widespread development of expert system applications.

Another important aspect was that of testing and evaluation of expert system performance. An important question to ask is 'at what stage of development does expert system advice reach expert quality?'. The Turing Test was one method presented, however it is a difficult problem to address in view of the fact that a knowledge base is unlikely ever to be proved 100% correct.

In the final part of this section a description of the SYNAPSE expert system shell is presented. This shell was chosen as it was developed locally, was relatively cheap and had graphical capabilities and could be used for various problem types. SYNAPSE has an important mathematical tool for representing uncertain reasoning called real value logic (RVL). This was used extensively for the program utility. There are several negative aspects of synapse. The most important of these is that it is restricted to using the 640K conventional memory of a computer thereby limiting the number of applications that could be loaded into the computer at any one time. Finally towards the end of the development work of this thesis it was learned that the company that marketed SYNAPSE had closed. This will affect future development work as technical support will no longer be obtainable and a new shell will probably have to be sought.

7.2 Environmental Impact Assessment and the Rulebase Development

The writer found in the literature survey that expert systems in EIA applications have been used previously in research work. These applications were however very disappointing in that they tended to be approached from a point of view of investigating their potential and relevance in this

operations. This technique presents a smooth flow of program operation to the user. A very important feature of rulebases is that they have the ability to retrieve information directly from databases. Although this feature was not used in this thesis, the writer believes that it could well be required at a future date. For this reason the program utility provides a menu which would be used for various database management tasks such as viewing and updating databases. A major limitation of SYNAPSE is that its rulebases can only access dBase3 files directly. For this reason the writer presents another technique whereby rulebases can retrieve information automatically from various differing database files using a third party program.

7.4 Operational Review

After using this system to evaluate several case studies it was concluded that the methodology used was indeed viable considering the task for which this system was developed. There were however several aspects that were identified as being of concern. These were :

- o The accuracy of the rulebases.
- o Whether the rulebase categories were sufficiently divided up so as not to be of too general a nature.
- o Results are vulnerable in that the weightings can be tampered with to produce desired results.
- o Subjective nature of many of the rulebase questions and the user's perception of probable impacts could further bias the results.
- o Who would maintain and refine the rulebases and teach users how to operate the system.
- o Experience and skill of the operator/analyst will determine the effectiveness of the system.

Some of these problems are not new to EIA. It is unlikely that two experts will ever have the same opinion on a matter given the diversity of human nature. There is however a case to be made for consistency in evaluation and towards this end several proposals have been made (see Recommendations).

Finally, no matter what the methodology employed for environmental impact assessment, the validity of the results will depend on the quality of the information being used. The maxim, *garbage in, garbage out* applies only too well to the computer related forms of environmental assessment.

CHAPTER 8

RECOMMENDATIONS

*"Victory belongs to the most
persevering"*

Napoleon (French emperor)

Contents

- 8.1 SYNAPSE
- 8.2 Rulebase Maintenance
- 8.3 Weightings and EIA Preparation
- 8.4 Rulebase Use
- 8.5 Security
- 8.6 Database Interfaces
- 8.7 Program Application

RECOMMENDATIONS

As a result of the development of this system and subsequent testing of it the following recommendations are made :

8.1 SYNAPSE

An expert system shell should eventually be sought to replace SYNAPSE which has been discontinued.

8.2 Rulebase Maintenance

A team of experts and interested parties should be assembled to continually maintain, edit, update and fine-tune the expert system rulebases according to each member's speciality.

8.3 Weightings and EIA Preparation

A set of generalised guidelines should be prepared to help users in their selection of weightings and environmental characteristics with regard to categories of locations and project types that will be encountered.

8.4 Rulebase Use

As the system is by intention generalised in nature, a short guide document should be developed to enable users to use the widely differing rulebases as correctly as possible and answer subjective questions as accurately as possible.

8.5 Security

As no security features in terms of preventing intentional or unintentional rulebase corruption have been built into the system, a full backup of all rulebases and system files should be kept at all times and the system should be monitored to check for tampering.

CHAPTER 9

REFERENCES

REFERENCES

- 1) G.F. Luger and W.A. Siblefield, Artificial Intelligence and the Design of Expert Systems. (Redwood City; Benjamin/Cummings Publications, 1989)
- 2) J McDermott, R1, The Formative Years.(AI Magazine,1981,July).
- 3) E Soloway, E Bachant and K Jensen, Assessing the Maintainability of XCON-in-RIME : Coping with the problems of a very large rulebase. (California, Los Altos, Morgan Kaufmann,1987).
- 4) SABS 0100 - Code of Practice for the Structural Use of Concrete, Part 1 : Design. (The Council of the South African Bureau of Standards ,1980)
- 5) A Turing, Computing Machinery and Intelligence.(Mind 59:433-460).
- 6) Integrated Environmental Management Procedure, Integrated Environmental Management Guideline Series, Guideline Document 1. (S Africa, Dept. of Environmental Affairs, 1992)
- 7) Guidelines for Report Requirements, Integrated Environmental Management Guideline Series, Guideline Document 3. (S Africa, Dept. of Environmental Affairs, 1992)
- 8) Checklist of Environmental Characteristics, Integrated Environmental Management Guideline Series, Guideline Document 5. (S Africa, Dept. of Environmental Affairs, 1992)
- 9) A.M. Starfield and A.L.Bleloch, Expert Systems: An Approach to Problems in Ecological Management that are Difficult to Quantify.(J.Env.Management(1983).16, pgs 261-268)
- 10) R. Bisset, Methods for Environmental Impact Analysis : Recent Trends and Future Prospects. (J.Env.Management(1980).11, pgs 27-43)
- 11) M.P.Antunes and A.Camara, HyperAIA - An Integrated System for Environmental Impact Assessment. (J.Env.Management(1992).35, pgs 93-111)
- 12) J.K.Lein, Applying Expert Systems Technology to Carrying Capacity Assessment : a Demonstration Prototype. (J.Env.Management(1993).37, pgs 63-84)

- 13) J.B.Shopley and R.F Fuggle, A Comprehensive Review of Current Environmental Impact Assessment Methods and Techniques. (J.Env.Management(1984).18, pgs 25-47)
- 14) S.M.Hirst, Applied Ecology and the Real World.I. Institutional Factors and Impact Assessment. (J.Env.Management(1984).18, pgs 189-202)
- 15) G.E.Beanlands, Do EIA Methods Have a Future?. Symposium on Environmental Impact Assessment : Current Status and Future Prospects. (Crete, Organised by PADC, EIA and Planning Unit, University of Aberdeen, Aberdeen).
- 16) J K Lein, An Expert System Approach to Environmental Impact Assessment. (J. Env. Studies (1989).33, pgs 13-27).
- 17) D Whitman, N McGinnis, J Fahringer and J Baker, Design of an Environmental Evaluation System. (Columbus, Ohio : Battelle Columbus Labs.).
- 18) The Environmental Impact Assessment for Umzino Regional Water Services Corporation for a Sewage Treatment Works at Park Rynie. (Campbell, Bernstein and Irving, Durban, Jan 1986).
- 19) Morrison,P. and Morrison,E., Charles Babbage and his calculating Machines (eds 1961.,New York)
- 20) Redeeming Charles Babbage's Mechanical Computer (Scientific American, Feb 1993, pgs 62-67)
- 21) G.Boole, An Investigation into the Laws of Thought (1854, London : Walton & Maberly)
- 22) G.Frege, Die Grundlagen der Arithmtic (Breslau, 1884)
- 23) A.Tarski, Logic, Semantics, Metamathematics (1956, London : Oxford press)
- 24) E.Shapiro, Concurrent PROLOG : Collected Papers, Volumes 1 and 2. (Cambridge MA, 1987)
- 25) Lindsay et al, Applications of Artificial Intelligence for Organic Chemistry : The DENDRAL Project (New York :MacGraw-Hill)

- 26) Winograd,T. and Flores,F, Understanding Computers and Cognition (Norwood, NJ: Ablex)
- 27) Lenat,D.B., On automated scientific theory formulation: A case study using the AM program (Machine Intelligence, 1982)
- 28) Lenat,D.B., AM: An artificial intelligence approach to discovery in mathematics as heuristic search (1982)

CHAPTER 10

BIBLIOGRAPHY

BIBLIOGRAPHY

Expert Systems

Expert Systems for Civil Engineers - Technology and Application. Edited by Mary Lou Maher (New York; The American Society of Civil Engineers, 1987)

E. Payne and R. McArthur, Developing Expert Systems, A Knowledge Engineers Handbook for Rules and Objects. (USA, Wiley, 1990)

Synapse Expert Systems - User Manual Version 2 (Johannesburg, Hitep, 1988)

Synapse Expert Systems - User Manual Version 3 (Johannesburg, Hitep, 1989)

Turbo Pascal/Programming

T. Swan, Mastering Turbo Pascal 5.5. Third Edition. (Carmel; Hayden Books, 1990)

S. Canter, Applications Development.(PC Magazine, July 1992)

S.K. O'Brian, Turbo Pascal 5.5, The Complete Reference. (Berkley, CA, Osborne McGraw-Hill, 1989)

N.J.Rubenking, Turbo Pascal 6, Techniques and Utilities. (Emeryville, CA, Ziff-Davis Press, 1991)

Borland, Turbo Pascal 7, Users Guide. (Ireland, Borland, 1992)

Borland, Turbo Pascal 7, Programmers Reference. (Ireland, Borland, 1992)

Borland, Turbo Pascal 7, Language Guide. (Ireland, Borland, 1992)

Borland, Turbo Vision, Programmers Guide. (Ireland, Borland, 1992)

Environmental Management

Guidelines for Scoping, Integrated Environmental Management Guideline Series, Guideline Document 2. (S Africa, Dept. of Environmental Affairs, 1992)

Guidelines for Report Requirements, Integrated Environmental Management Guideline Series, Guideline Document 3. (S Africa, Dept. of Environmental Affairs, 1992)

Guidelines for Review, Integrated Environmental Management Guideline Series, Guideline Document 4. (S Africa, Dept. of Environmental Affairs, 1992)

Glossary of Terms used in Integrated Environmental Management, Integrated Environmental Management Guideline Series, Guideline Document 6. (S Africa, Dept. of Environmental Affairs, 1992)

Environmental Awareness for Engineers, Course notes for a Seminar (S Africa, Environmental Evaluation Unit, UCT, 1993)

R.C.Hill, Environmental Engineering, EIA Theory and Practice. (Paper Presented for Seminar on Environmental Engineering, SAICE, 1992)

SAICE, Division of Highway and Traffic Engineering, Environmental Impact Studies, Sixth Quinquennial Convention, Course Notes (S Africa, Prepared by De Leuw Cather)

N.Lee, The Future Development of Environmental Impact Assessment. (J.Env.Management(1982).14, pgs 71-90)

APPENDIX 1

SYNAPSE mathematics is very simple to apply and use. The basic structure of the mathematics means that you need have no other mathematical skill beyond addition and multiplication. This appendix has been added to give a derivation of the SYNAPSE RVL formula from the point of view of Bayesian statistics.

Let

H be an hypothesis,

E be evidence,

P(H) be the probability that the hypothesis is TRUE
(sometimes known as the PRIOR probability),

P(H|E) be the probability of the hypothesis being TRUE, given that the evidence is TRUE,

~ be the symbol for NOT. For example, P(H|~E) is the probability that the hypothesis is TRUE, given that the evidence is FALSE.

Now the Bayes' formula for updating a prior probability given some evidence is

$$\begin{aligned}P(H|E) &= \frac{P(H)P(E|H)}{P(E)} \\&= \frac{P(H)P(E|H)}{P(H)P(E|H) + P(\sim H)P(E|\sim H)}\end{aligned}$$

Multiplying this formula by 1 in the form of

$$\frac{P(E|\sim H)P(\sim H)}{P(E|\sim H)P(\sim H)}$$

gives

$$P(H|E) = \frac{P(H)P(E|H)}{P(E|\sim H)P(\sim H)P(H)} \frac{P(E|\sim H)P(\sim H)}{P(E|H) + P(\sim H)P(E|\sim H)}$$

This is of the form

$$\frac{Z}{1 + Z}$$

where

$$Z = \frac{P(E|H)}{P(E|\sim H)} \frac{P(H)}{P(\sim H)}$$

Z is also known as the ODDS of the hypothesis given the evidence. The first factor in the equation for Z is the logical sufficiency (LS). This factor is a measure of how much the presence of the evidence E strengthens the hypothesis H.

So the equation for Z can be written as

$$o_{H|E} = LS \cdot o_H \quad (1)$$

This formula is based on the assumption that the evidence is TRUE. If this is not the case, the following formula is derived in a similar fashion

$$o_{H|\sim E} = LN \cdot o_H \quad (2)$$

where LN is the logical necessity. This factor is the measure of how much the absence of the evidence E makes the hypothesis H more likely.

It would be convenient to have a scaling factor to simulate the uncertainty in the evidence. This is necessary because the evidence itself need not be TRUE or FALSE, but may take on a value somewhere between.

In order to provide a smooth transition between TRUE evidence and FALSE evidence it is useful to perform a logarithmic transformation. So (1) and (2) become

$$\log(o_{H|E}) = \log(LS) + \log(o_H) \quad \text{if } E \text{ TRUE}$$

$$\log(o_{H|\sim E}) = \log(LN) + \log(o_H) \quad \text{if } E \text{ FALSE}$$

It is now possible to introduce a scaling factor. When $P(E) = 0,5$, this value should be 0. When $P(E) = 1,0$, the value is 1, and when $P(E) = 0$, the value should be -1.0. This will allow the formula to be modified in the following way to achieve a smooth transition from E being TRUE to E being FALSE. Suppose we introduce a value R that has these properties and acts in a linear fashion.

$$\log(o_H|E) = \log(LS).R + \log(o_H) \quad \text{if } R > 0$$

$$\log(o_H|E) = \log(LN).|R| + \log(o_H) \quad \text{if } R < 0$$

$$\log(o_H|E) = \log(o_H) \quad \text{if } R = 0$$

There are no real theoretical reasons for giving preference in the choice of the function that maps $P(E)$ into R. SYNAPSE uses a linear mapping from $P(E)$ in the interval $[0,1]$ to R in the interval $[-1,1]$.

This then yields the RVL formula given in Chapter 3.10.3.

$$RVL_{H|E} = RVL_H + (|Truth_E| \times Effect_{H|E})$$

While this derivation shows the close ties between RVL and this formulation of Bayes' Theorem, it is essential to ensure that all the Bayes' assumptions are met before the inverse of moving from RVL back to Bayes is applied. RVL should be seen as a method of modelling uncertainty in its own right.

APPENDIX 2

AWK Program for the Interfacing of Rulebases and Databases

APPENDIX 2

In this appendix the method of extracting information from databases of any type is presented using a short example. The flow chart showing how this method works is shown in figure B.01.

The information required in Risk2.rb is extracted from it using AWK programme RbAns.awk and is written into a template file Risk2.ors. The program Retrieve.pas uses the template file to find out what information to extract from the database file KnowAll.dbf. The program goes on to extract this information and write it together with the SYNAPSE Question rule that wants it, in an answer file, Risk2.ans, ready to be used in the next SYNAPSE run.

There can be variations of this method. One variation is to enable the rulebase to request information from multiple databases. An example of this is shown in Risk3.rb and Risk3.ors. In the Question rule of Risk3.rb an additional parameter is provided whereby a specific database is given for the retrieval program to look for the information requested. The particular format of the database is identified by the extension to the filename. In this particular example a ".dbf" extension refers to a dBase3 format and the ".wdf" a Microsoft Works database format. The retrieval program is designed to handle each database format according to its own particular architecture.

rem
rem
rem
rem
rem

Example DOS batchfile for implementing information retrieval from database files

RbAns.rb Risk2.rb

Retrieve.exe Risk2.ors Dbase.dbf

Synapse run -fRisk2 -aRisk2.ans -rRisk2.res

```
FNR == 1 \
{
  aname = tolower(FILENAME)
  sub(/.rb$/, ".ors", aname)
}

{
  printf("%4d\r", FNR) > "con"
}

match($0,/^NUMBER +/) \
{
  number = substr($0,RLENGTH+1)
  sub(/ +$/, "", number)
  qtype = "n"
  printf("%4d number question\r", FNR) > "con"
  next
}

match($0,/^SCALE +/) \
{
  number = substr($0,RLENGTH+1)
  sub(/ +$/, "", number)
  qtype = "s"
  printf("%4d scale question\r", FNR) > "con"
  next
}

match($0,/( *DBASE +.+)/) \
{
  $0 = substr($0, RSTART+1, RLENGTH-2)
  file = $2
  $1 = ""
  $2 = ""
  sub(/^ */, "")
  sub(/ *$/, "")
  element = $0
  if (db3list[file " " element])
    print "\n*** element " element " in file " file " *** already defined" >
"con"
  db3list[file " " element]=number
  db3listtype[file " " element]=qtype
  printf("%4d database lookup\r", FNR) > "con"
  next
}

END \
{
  for (file in db3list)
    printf("%s\n", file) > aname
}
```



```

program retrieve;

uses crt,objects,addru,db3read;

type string30=string[30];

var infile1,infile2,outfile:text;
    ch:char;
    nameslist : addresslist;
    namesfile : addressfile;
    dbase_obj : db3_std;

procedure message (s:string);
begin
    writeln;
    write('---('memavail, ')--- press <enter> to ',s);
    readln;
end;

procedure openfiles;
var filename : string[12];
begin
    if paramcount < 3 then begin
        writeln('Usage');
        writeln(' process2 infile1 infile2 outfile');
        exit;
    end;
    filename:=paramstr(1);
    assign(infile1,filename);
    writeln('input file 1 = ',filename);
    filename:=paramstr(2);
    assign(infile2,filename);
    writeln('input file 2 = ',filename);
    filename:=paramstr(3);
    assign(outfile,filename);
    writeln('output file = ',filename);
    rewrite(outfile);
    filename:='test1.dbf';
    dbase_obj.init(filename,1024);
end;

procedure noblanks(var s:string30);
var j:integer;
begin
    j:=1;
    while (j<=length(s)) do
        if s[j]=' ' then delete(s,j,1) else inc(j);
    end;
end;

procedure processfiles;
var instring,status,final:string30;
    found:boolean;
    ans:integer;
begin
    reset(infile1);
    while not eof (infile1) do
        begin
            readln(infile1,instring);
            noblanks(instring);
            {$V+}

```

```

            nameslist.searchlist(instring,status,found);
            {$V+}
            if found=true then
                final:=concat(instring,',',status)
            else final:=concat(instring,',not found');
            if found=true then ans:=1 else ans:=2;
            writeln(instring,' ',ans,' ',final);
            writeln(outfile,final);
        end;
    end;
end;

procedure readin_database;
var line_string1,line_string2 : string30;

begin
    while dbase_obj.readnext <>-1 do
        begin
            line_string1:=dbase_obj.getfield('NAME');
            line_string2:=dbase_obj.getfield('VALUE');
            noblanks(line_string1);
            noblanks(line_string2);
            nameslist.append(new(addrobjptr,init(line_string1,line_string2,3443,31,1)));
        end;
    end;
end;

begin
    clrscr;
    openfiles;
    nameslist.clear;
    writeln('reading in database');
    readin_database;
    message('display list in memory');
    nameslist.displaylist;
    writeln;
    message('process files');
    processfiles;
    message('store list on disk');
    namesfile.init('testdata.odb',screate,1024);
    nameslist.store(namesfile);
    namesfile.done;
    message('delete list in memory');
    nameslist.delete;
    message('read list from disk');
    namesfile.init('testdata.odb',sopen,1024);
    nameslist.load(namesfile);
    namesfile.done;
    message('display list in memory');
    nameslist.displaylist;
    close(infile1);
    { close(infile2); }
    close(outfile);
    nameslist.delete;
    repeat until keypressed;
    while keypressed do ch:=readkey;
    if ch=#27 then exit;end.
end.

```

```
unit db3read;
interface
uses
    genfunc;

type
    db3_header_ptr = ^db3_header;

    field10 = array [1..10] of char;

    db3_header = record
        version      : byte;
        date_updated : array[1..3] of byte;
        no_record    : longint;
        header_length, record_length : word;
        x            : array[1..20] of char;
    end;

    db3_field_arr_ptr = ^db3_field_arr;

    db3_field = record
        field : field10;
        x1    : char;
        typ   : char;
        address : longint;
        len   : byte;
        decs  : byte;
        x2    : array[1..14] of char;
    end;

    db3_field_arr = array[1..200] of db3_field;

    charptr = ^char;

    db3_access = object
        dbf      : file;
        header   : db3_header_ptr;
        fieldsize : word;
        fields   : db3_field_arr_ptr;
        no_fields : integer;
        offset   : longint;
        constructor init(dbase : string);
        destructor  done;
    end;

    db3_std = object(db3_access)
        no_in_buf : integer;
        cur_in_buf : integer;
        buf_size  : word;
        blkBuf    : charptr;
        recbuf    : charptr;
        constructor init(dbase : string; buf_size : word);
        destructor  done;
        function    readnext : longint; virtual;
        function    getfield(name : string) : string;
        procedure   putfield(name, s : string);
        procedure   writecurr;
    end;
```

DESCRIPTION (

Risk and Hazard

This expert system rulebase is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any risk or hazard that would result from the development of the project in question.

)

GOAL Assessment

TRUE (The ecological characteristics of the site is favourable)

FALSE (The ecological characteristics of the site is unfavourable)

HELP ()

<= identity of hazard	PRIORITY 100	-0.200	0.200
<= probability of occurrence	PRIORITY 90	-0.200	0.200
<= extent of effect	PRIORITY 80	-0.200	0.200
<= workers safety	PRIORITY 70	-0.200	0.200
<= risk for other orgs	PRIORITY 60	-0.200	0.200

SCALE identity_of_hazard

TRUE (The level and identity of hazard to the public rate is high)

FALSE (The level and identity of hazard to the public rate is low)

HELP (a)

QUESTION (DBASE identity_of_hazard)

LABELS (Low) (Average) (High)

SCALE probability_of_occurrence

TRUE (The probability of occurrence is high)

FALSE (The probability of occurrence is low)

HELP (a)

QUESTION (DBASE probability_of_occurrence)

LABELS (Low) (Average) (High)

SCALE extent_of_effect

TRUE (The extent of the effect is panoramic)

FALSE (The extent of the effect is local)

HELP (a)

QUESTION (DBASE extent of effect)

LABELS (Local) (Regional) (Panoramic)

SCALE workers_safety

TRUE (The workers degree of risk is high)

FALSE (The workers degree of risk is low)

HELP (a)

QUESTION (DBASE workers safety)

LABELS (Low) (Average) (High)

SCALE risk_for_other_orgs

TRUE (The level of risk and hazard for other living organisms is high)

FALSE (The level of risk and hazard for other living organisms is low)

HELP (a)

QUESTION (DBASE risk_for_other_orgs)

LABELS (Low) (Average) (High)

extent of effect
knowall.dbf identity_of_hazard
probability of occurrence
risk_for_other_orgs
workers_safety

DESCRIPTION (Risk and Hazard

This expert system rulebase is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any risk or hazard that would result from the development of the project in question.

GOAL Assessment

TRUE (The ecological characteristics of the site is favourable)
FALSE (The ecological characteristics of the site is unfavourable)
HELP ()

<= identity of hazard	PRIORITY 100	-0.200	0.200
<= probability of occurrence	PRIORITY 90	-0.200	0.200
<= extent of effect	PRIORITY 80	-0.200	0.200
<= workers safety	PRIORITY 70	-0.200	0.200
<= risk for other orgs	PRIORITY 60	-0.200	0.200

SCALE identity_of_hazard

TRUE (The level and identity of hazard to the public rate is high)
FALSE (The level and identity of hazard to the public rate is low)
HELP (a)
QUESTION (DBASE Hazards.dbf Identity_of_hazard)
LABELS (Low) (Average) (High)

SCALE probability_of_occurrence

TRUE (The probability of occurrence is high)
FALSE (The probability of occurrence is low)
HELP (a)
QUESTION (DBASE Knowall.dbf Probability_of_occurrence)
LABELS (Low) (Average) (High)

SCALE extent_of_effect

TRUE (The extent of the effect is panoramic)
FALSE (The extent of the effect is local)
HELP (a)
QUESTION (DBASE Knowall.dbf Extent_of_effect)
LABELS (Local) (Regional) (Panoramic)

SCALE workers_safety

TRUE (The workers degree of risk is high)
FALSE (The workers degree of risk is low)
HELP (a)
QUESTION (DBASE Safety.wdb Workers_safety)
LABELS (Low) (Average) (High)

SCALE risk_for_other_orgs

TRUE (The level of risk and hazard for other living organisms is high)
FALSE (The level of risk and hazard for other living organisms is low)
HELP (a)
QUESTION (DBASE Safety.wdb Risk_for_other_orgs)
LABELS (Low) (Average) (High)

Hazards.dbf Identity_of_hazard
Knowall.dbf Extent_of_effect
Knowall.dbf Probability_of_occurrence
Safety.wdb Risk_for_other_orgs
Safety.wdb Workers_safety

APPENDIX 3

Expert System Rulebase Listings

SUMMARY OF RULEBASE NAMES

<u>Rulebase Name</u>	<u>Categories and Description</u>
<u>Physical Characteristics of the Site and its Surroundings</u>	
LandClim	Land, Climate
LanClimW	Land, Climate, Freshwater systems
LanClimM	Land, Climate, Marine and Estuarine systems
<u>Ecological Characteristics of the Site and its Surroundings</u>	
Eco.rb	All ecological characteristics
<u>Current and Potential Land Use and Landscape Character</u>	
CharOpen.rb	General considerations and urban open space
CharResd.rb	General considerations and residential areas
CharComm.rb	General considerations and commercial areas
CharInds.rb	General considerations and industrial areas
CharAgri.rb	General considerations and agricultural areas
<u>Cultural Resources</u>	
Cultural.rb	All cultural resources
<u>Infrastructure Services</u>	
Energy.rb	Energy supply
WaterInf.rb	Water
WasteInf.rb	Waste management
Transprt.rb	Transport Networks
<u>Social and Community Services and Facilities</u>	
Social.rb	All aspects of social and community services

**Nature and Level of Present and Future
Environmental Pollution**

Air.rb	Air pollution
WaterPol.rb	Water pollution
NoisePol.rb	Noise pollution

Risk and Hazard

Risk.rb	All aspects of risk and hazard

Health and Safety

Health.rb	All aspects of health and safety

Cumulative and Synergistic Effects

Synergis.rb effects	All aspects regarding cumulative and synergistic

/*
Current and Potential Land use and Landscape Character

This Expert System Rule Base is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any of the following :

- 1) General considerations applicable to all development proposals
- 2) Agricultural and sylvicultural areas

*/
DESCRIPTION (Current and Potential Land use and Landscape Character -
Agricultural and sylvicultural areas)

GOAL Assessment

TRUE (The current and potential landuse character is favourable)
FALSE (The current and potential landuse character is unfavourable)

HELP ()
<= General PRIORITY 100 -0.300 0.300
<= Agricultural PRIORITY 90 -0.700 0.700

/*
***** General *****

SUBGOAL General

TRUE (The vegetation of the land will not be affected)
FALSE (The vegetation of the land will be affected)

HELP (a)
<= compatability -0.091 0.091
<= aesthetic 0.091 -0.091
<= place -0.091 0.091
<= character 0.091 -0.091
<= scale_of_develop 0.091 -0.091
<= materials 0.091 -0.091
<= views 0.091 -0.091
<= reveitilization -0.091 0.091
<= landscape -0.091 0.091
<= buffer 0.091 -0.091
<= claims 0.091 -0.091
<= legal 0.091 -0.091

MENU compatability

TRUE (The compatability of land uses within the area is favourable)
FALSE (The compatability of land uses within the area is unfavourable)

HELP (a)
QUESTION (How does the compatability of the land use rate)
OPTIONS
1 (The land use is favourable)
-1 (The land use is unfavourable)

YESNO aesthetic

TRUE (The aesthetic quality of the landscape will be affected)
FALSE (The aesthetic quality of the landscape will not be affected)
HELP (a)
QUESTION (Will the aesthetic quality of the landscape be affected)

SCALE place

TRUE (The sense of place within the area is highly regarded)
FALSE (The sense of place within the area is poorly regarded)

HELP (a)
QUESTION (How is the sense of place within the area regarded)
LABELS (Poorly) (Ambivalent) (Highly)

MENU character

TRUE (The character of the overall area will be affected)
FALSE (The character of the overall area will not be affected)

HELP (a)
QUESTION (How will the character of the overall area be affected)
OPTIONS
1 (Detrimntally)
0 (Slightly)
-1 (No effect)

SCALE scale of develop

TRUE (The projects compatibility with the scale of developments in the area is poor)
FALSE (The projects compatibility with the scale of developments in the area is high)

HELP (a)
QUESTION (What is the projects compatibility with the scale of developments in the area)
LABELS (High) (Average) (Poor)

SCALE materials

TRUE (The projects compatibility with building materials used in the area is poor)
FALSE (The projects compatibility with building materials used in the area is high)

HELP (a)
QUESTION (What is the projects compatibility with building materials used in the area)
LABELS (High) (Average) (Poor)

YESNO Views

TRUE (Scenic views and features of the areaa will be affected)
FALSE (Scenic views and features of the areaa will not be affected)
HELP (a)
QUESTION (Will the scenic views and features of the area be affected)

YESNO reveitilization

TRUE (The project will encourage the reveitilization of neighbouring run-down areas)
FALSE (The project will discourage the reveitilization of neighbouring run-down areas)

HELP (a)
QUESTION (Will the project encourage the reveitilization of neighbouring run-down areas)

YESNO landscape

TRUE (The projects plans include providing for landscaping/site restoration)
FALSE (The projects plans exclude providing for landscaping/site restoration)
HELP (a)
QUESTION (Does the projects plans provide for landscaping of the site/site restoration)

YESNO buffer

TRUE (The site does require buffer zones to allow for natural processes)
FALSE (The site does not require buffer zones to allow for natural processes)
HELP (a)

QUESTION (Does the site require buffer zones to allow for natural processes)

YESNO claims

TRUE (There are political considerations such as land claims or historical rights)

FALSE (There are no political considerations such as land claims or historical rights)

HELP (a)

QUESTION (Are there political considerations such as land claims or historical rights)

YESNO legal

TRUE (There are legal considerations such as servitudes and rights of way)

FALSE (There are no legal considerations such as servitudes and rights of way)

HELP (a)

QUESTION (Are there legal considerations such as servitudes and rights of way)

 ***** Agricultural or silvicultural areas *****

SUBGOAL agricultural

TRUE (The affect of the site on agricultural areas makes the site suitable for development)

FALSE (The affect of the site on agricultural areas makes the site unsuitable for development)

HELP (a)

<= agri potential	0.180	-0.180
<= agri use	0.450	-0.450
<= agri forests	0.034	-0.034
<= agri greenbelts	-0.034	0.034
<= agri water	-0.034	0.034
<= agri pollution	0.034	-0.034
<= agri disease	0.034	-0.034
<= agri toxins	0.034	-0.034
<= agri erosion	0.034	-0.034
<= agri bush	0.034	-0.034
<= agri methods	0.034	-0.034
<= agri invasive	0.034	-0.034
<= agri housing	-0.034	0.034

YESNO agri potential

TRUE (The development will use high-potential farmland)

FALSE (The development will not use high-potential farmland)

HELP (a)

QUESTION (Will the development use high-potential farmland)

SCALE agri use

TRUE (The projects use of productive agricultural land is very large)

FALSE (The projects use of productive agricultural land is very small)

HELP (a)

QUESTION (To what extent will the project reduce productive agricultural land)

LABELS (None) (Average) (Large Areas)

YESNO agri forests

TRUE (The development will use areas available for commercial forests)

FALSE (The development will not use areas available for commercial forests)

HELP (a)

QUESTION (Will the development use areas available for commercial forests)

YESNO agri greenbelts

TRUE (The development will use be located in a greenbelt or buffer zone)

FALSE (The development will not use be located in a greenbelt or buffer zone)

HELP (a)

QUESTION (Will the development be located in a greenbelt or buffer zone)

YESNO agri water

TRUE (There is water available for the development)

FALSE (There is no water available for the development)

HELP (a)

QUESTION (Is there water available for the development)

YESNO agri pollution

TRUE (The development will be affected by pollution due to pesticides, fertilizers, etc)

FALSE (The development will not be affected by pollution due to pesticides, fertilizers, etc)

HELP (a)

QUESTION (Will the development be affected by pollution due to pesticides, fertilizers, etc)

YESNO agri disease

TRUE (The development will be affected by disease control activities such as crop spraying)

FALSE (The development will not be affected by disease control activities such as crop spraying)

HELP (a)

QUESTION (Will the development be affected by disease control activities such as crop spraying)

YESNO agri toxins

TRUE (The development will increase levels of toxins, dust and bad smells in the air)

FALSE (The development will not increase levels of toxins, dust and bad smells in the air)

HELP (a)

QUESTION (Will the development increase levels of toxins, dust and bad smells in the air)

YESNO agri erosion

TRUE (The development will increase erosion and sedimentation in the rivers)

FALSE (The development will not increase erosion and sedimentation in the rivers)

HELP (a)

QUESTION (Will the development increase erosion and sedimentation in the rivers)

YESNO agri bush

TRUE (The development will increase bush encroachment)

FALSE (The development will not increase bush encroachment)

HELP (a)

QUESTION (Will the development increase bush encroachment)

YESNO agri methods

TRUE (The development will further damage land due to overgrazing or bad farming methods)

FALSE (The development will not further damage land due to overgrazing or bad farming methods)

HELP (a)

QUESTION (Will the development further damage land due to overgrazing or bad farming methods)

ing methods}

YESNO agri invasive

TRUE {The development will be affected by invasive alien plants}

FALSE {The development will not be affected by invasive alien plants}

HELP {a}

QUESTION {Will the development be affected by invasive alien plants}

YESNO agri housing

TRUE {The development will make provision for further housing and education facilities}

FALSE {The development will not make provision for further housing and education facilities}

HELP {a}

QUESTION {Will the development make provision for further housing and education facilities}

/*
Current and Potential Land use and Landscape Character

This Expert System Rule Base is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any of the following :

1) General considerations applicable to all development proposals

4) Commercial areas

*/

DESCRIPTION (Current and Potential Land use and Landscape Character -
Commercial Areas)

GOAL Assessment

TRUE (The current and potential landuse character is favourable)

FALSE (The current and potential landuse character is unfavourable)

HELP ()

<= General PRIORITY 100 -0.300 0.300

<= Commercial PRIORITY 90 -0.700 0.700

/*

***** General *****

*/

SUBGOAL General

TRUE (The vegetation of the land will not be affected)

FALSE (The vegetation of the land will be affected)

HELP (a)

<= compatability -0.091 0.091

<= aesthetic 0.091 -0.091

<= place -0.091 0.091

<= character 0.091 -0.091

<= scale of develop 0.091 -0.091

<= materials 0.091 -0.091

<= views 0.091 -0.091

<= revitalization -0.091 0.091

<= landscape -0.091 0.091

<= buffer 0.091 -0.091

<= claims 0.091 -0.091

<= legal 0.091 -0.091

MENU compatability

TRUE (The compatability of land uses within the area is favourable)

FALSE (The compatability of land uses within the area is unfavourable)

HELP (a)

QUESTION (How does the compatability of the land use rate)

OPTIONS

1 (The land use is favourable)

-1 (The land use is unfavourable)

YESNO aesthetic

TRUE (The aesthetic quality of the landscape will be affected)

FALSE (The aesthetic quality of the landscape will not be affected)

HELP (a)

QUESTION (Will the aesthetic quality of the landscape be affected)

SCALE place

TRUE (The sense of place within the area is highly regarded)

FALSE (The sense of place within the area is poorly regarded)

HELP (a)

QUESTION (How is the sense of place within the area regarded)

LABELS (Poorly) (Ambivalent) (Highly)

MENU character

TRUE (The character of the overall area will be affected)

FALSE (The character of the overall area will not be affected)

HELP (a)

QUESTION (How will the character of the overall area be affected)

OPTIONS

1 (Detrimentially)

0 (Slightly)

-1 (No effect)

SCALE scale of develop

TRUE (The projects compatibility with the scale of developments in the area is
poor)

FALSE (The projects compatibility with the scale of developments in the area is
high)

HELP (a)

QUESTION (What is the projects compatibility with the scale of developments in
the area)

LABELS (High) (Average) (Poor)

SCALE materials

TRUE (The projects compatibility with building materials used in the area
is poor)

FALSE (The projects compatibility with building materials used in the area
is high)

HELP (a)

QUESTION (What is the projects compatibility with building materials used
in the area)

LABELS (High) (Average) (Poor)

YESNO Views

TRUE (Scenic views and features of the areaa will be affected)

FALSE (Scenic views and features of the areaa will not be affected)

HELP (a)

QUESTION (Will the scenic views and features of the area be affected)

YESNO revitalization

TRUE (The project will encourage the revitalization of neighbouring run-down
areas)

FALSE (The project will discourage the revitalization of neighbouring run-down
areas)

HELP (a)

QUESTION (Will the project encourage the revitalization of neighbouring run-down
areas)

YESNO landscape

TRUE (The projects plans include providing for landscaping/site restoration)

FALSE (The projects plans exclude providing for landscaping/site restoration)

HELP (a)

QUESTION (Does the projects plans provide for landscaping of the site/site
restoration)

YESNO buffer

TRUE (The site does require buffer zones to allow for natural processes)

FALSE (The site does not require buffer zones to allow for natural processes)

HELP (a)

QUESTION {Does the site require buffer zones to allow for natural processes}

YESNO claims

TRUE {There are political considerations such as land claims or historical rights}

FALSE {There are no political considerations such as land claims or historical rights}

HELP {a}

QUESTION {Are there political considerations such as land claims or historical rights}

YESNO legal

TRUE {There are legal considerations such as servitudes and rights of way}

FALSE {There are no legal considerations such as servitudes and rights of way}

HELP {a}

QUESTION {Are there legal considerations such as servitudes and rights of way}

/*****
 /***** Commercial *****/
 /*****/

SUBGOAL Commercial

TRUE {The affect of the site on commercial areas makes the site suitable for development }

FALSE {The affect of the site on commercial areas makes the site unsuitable for development }

HELP {a}

<= urban character	0.125	-0.125
<= urban traffic	0.125	-0.125
<= siting	-0.125	0.125
<= parking	-0.125	0.125
<= walkways	-0.125	0.125
<= conflict	0.125	-0.125
<= safety	0.125	-0.125
<= decay	0.125	-0.125

YESNO urban character

TRUE {The site will affect the character of the urban centre}

FALSE {The site will not affect the character of the urban centre}

HELP {a}

QUESTION {Will the site affect the character of the urban centre}

YESNO urban traffic

TRUE {The development will affect the volume of traffic in the urban centre}

FALSE {The development will not affect the volume of traffic in the urban centre}

HELP {a}

QUESTION {Will the development affect the volume of traffic in the urban centre}

YESNO siting

TRUE {The development will be sited appropriately}

FALSE {The development will not be sited appropriately}

HELP {a}

QUESTION {Will the development be sited appropriately}

YESNO parking

TRUE {The urban area will have sufficient parking}

FALSE {The urban area will not have sufficient parking}

HELP {a}

QUESTION {Will the urban area have sufficient parking}

YESNO walkways

TRUE {The urban area will have sufficient pedestrain walkways}

FALSE {The urban area not have sufficient pedestrain walkways}

HELP {a}

QUESTION {Will the urban area have sufficient pedestrain walkways}

YESNO conflict

TRUE {The development will result in conflict between pedestrain and vehicular traffic}

FALSE {The development will not result in conflict between pedestrain and vehicular traffic}

HELP {a}

QUESTION {Will the development result in conflict between pedestrain and vehicular traffic}

YESNO safety

TRUE {The saftey of the urban area will be affected}

FALSE {The saftey of the urban area will not be affected}

HELP {a}

QUESTION {Will the saftey of the urban area be affected}

YESNO decay

TRUE {The development will increase the rate of decay or change the character of the area}

FALSE {The development will not increase the rate of decay or change the character of the area}

HELP {a}

QUESTION {Will the development increase the rate of decay or change the character of the area}

```

/*
Current and Potential Land use and Landscape Character

This Expert System Rule Base is specifically designed to investigate
whether the proposed development could have a significant impact on,
or be constrained by any of the following :

1) General considerations applicable to all development proposals
2) Industrial areas
*/

DESCRIPTION ( Current and Potential Land use and Landscape Character -
Industrial Areas )

GOAL Assessment
TRUE ( The current and potential landuse character is favourable)
FALSE ( The current and potential landuse character is unfavourable )
HELP ( )
<= General      PRIORITY 100  -0.300  0.300
<= Industrial   PRIORITY 90   -0.700  0.700

/*****
/***** General *****/
/*****

SUBGOAL General
TRUE (The vegetation of the land will not be affected )
FALSE ( The vegetation of the land will be affected )
HELP (a)
<= compatability  -0.091  0.091
<= aesthetic      0.091 -0.091
<= place          -0.091  0.091
<= character       0.091 -0.091
<= scale_of_develop 0.091 -0.091
<= materials      0.091 -0.091
<= views          0.091 -0.091
<= revitalization -0.091  0.091
<= landscape      -0.091  0.091
<= buffer         0.091 -0.091
<= claims         0.091 -0.091
<= legal         0.091 -0.091

MENU compatability
TRUE (The compatability of land uses within the area is favourable)
FALSE (The compatability of land uses within the area is unfavourable)
HELP (a)
QUESTION ( How does the compatability of the land use rate)
OPTIONS
1 (The land use is favourable)
-1 (The land use is unfavourable)

YESNO aesthetic
TRUE (The aesthetic quality of the landscape will be affected)
FALSE (The aesthetic quality of the landscape will not be affected)
HELP (a)
QUESTION (Will the aesthetic quality of the landscape be affected)

SCALE place
TRUE (The sense of place within the area is highly regarded)
FALSE (The sense of place within the area is poorly regarded)

```

```

HELP (a)
QUESTION (How is the sense of place within the area regarded)
LABELS (Poorly) (Ambivalent) (Highly)

MENU character
TRUE (The character of the overall area will be affected)
FALSE (The character of the overall area will not be affected)
HELP (a)
QUESTION (How will the character of the overall area be affected)
OPTIONS
1 (Detrimentially)
0 (Slightly)
-1 (No effect)

SCALE scale_of_develop
TRUE (The projects compatibility with the scale of developments in the area is
poor)
FALSE (The projects compatibility with the scale of developments in the area is
high)
HELP (a)
QUESTION (What is the projects compatibility with the scale of developments in
the area)
LABELS (High) (Average) (Poor)

SCALE materials
TRUE (The projects compatibility with building materials used in the area
is poor)
FALSE (The projects compatibility with building materials used in the area
is high)
HELP (a)
QUESTION (What is the projects compatibility with building materials used
in the area)
LABELS (High) (Average) (Poor)

YESNO Views
TRUE (Scenic views and features of the areaa will be affected)
FALSE (Scenic views and features of the areaa will not be affected)
HELP (a)
QUESTION (Will the scenic views and features of the area be affected)

YESNO revitalization
TRUE (The project will encourage the revitalization of neighbouring run-down
areas)
FALSE (The project will discourage the revitalization of neighbouring run-down
areas)
HELP (a)
QUESTION (Will the project encourage the revitalization of neighbouring run-down
areas)

YESNO landscape
TRUE (The projects plans include providing for landscaping/site restoration)
FALSE (The projects plans exclude providing for landscaping/site restoration)
HELP (a)
QUESTION (Does the projects plans provide for landscaping of the site/site
restoration)

YESNO buffer
TRUE (The site does require buffer zones to allow for natural processes)
FALSE (The site does not require buffer zones to allow for natural processes)
HELP (a)

```

QUESTION (Does the site require buffer zones to allow for natural processes)

YESNO claims

TRUE (There are political considerations such as land claims or historical rights)

FALSE (There are no political considerations such as land claims or historical rights)

HELP (a)

QUESTION (Are there political considerations such as land claims or historical rights)

YESNO legal

TRUE (There are legal considerations such as servitudes and rights of way)

FALSE (There are no legal considerations such as servitudes and rights of way)

HELP (a)

QUESTION (Are there legal considerations such as servitudes and rights of way)

/*****
 /***** Industrial *****/
 /*****/

SUBGOAL Industrial

TRUE (The affect of the site on industrial areas makes the site suitable for development)

FALSE (The affect of the site on industrial areas makes the site unsuitable for development)

HELP (a)

<= indus traffic 0.200 -0.200

<= indus parking -0.200 0.200

<= indus runoff 0.200 -0.200

<= indus pollution 0.200 -0.200

<= indus aesthetic 0.200 -0.200

YESNO indus traffic

TRUE (The development will affect the volume of traffic in the industrial centre)

FALSE (The development will not affect the volume of traffic in the industrial centre)

HELP (a)

QUESTION (Will the development affect the volume of traffic in the industrial centre)

YESNO indus parking

TRUE (The industrial area will have sufficient parking)

FALSE (The industrial area will not have sufficient parking)

HELP (a)

QUESTION (Will the industrial area have sufficient parking)

YESNO indus runoff

TRUE (The site will result in polluted street runoff)

FALSE (The site will not result in polluted street runoff)

HELP (a)

QUESTION (Will the site result in polluted street runoff)

YESNO indus pollution

TRUE (The site will affect local pollution levels)

FALSE (The site will not affect local pollution levels)

HELP (a)

QUESTION (Will the site affect local pollution levels)

YESNO indus aesthetic

TRUE (The site will affect the aesthetic quality of the industrial centre)

FALSE (The site will not affect the aesthetic quality of the industrial centre)

HELP (a)

QUESTION (Will the site affect the aesthetic quality of the industrial centre)

/*
Current and Potential Land use and Landscape Character

This Expert System Rule Base is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any of the following :

- 1) General considerations applicable to all development proposals
- 2) Urban open space, protected and recreational areas

*/

DESCRIPTION (Current and Potential Land use and Landscape Character -
Urban open space, protected and recreational areas)

GOAL Assessment

TRUE (The current and potential landuse character is favourable)
FALSE (The current and potential landuse character is unfavourable)
HELP ()

<= General PRIORITY 100 -0.300 0.300
<= Open Space PRIORITY 90 -0.700 0.700

/*****
***** General *****
*****/

SUBGOAL General

TRUE (The vegetation of the land will not be affected)
FALSE (The vegetation of the land will be affected)

HELP (a)
<= compatability -0.091 0.091
<= aesthetic 0.091 -0.091
<= place -0.091 0.091
<= character 0.091 -0.091
<= scale_of_develop 0.091 -0.091
<= materials 0.091 -0.091
<= views 0.091 -0.091
<= revitalization -0.091 0.091
<= landscape -0.091 0.091
<= buffer 0.091 -0.091
<= claims 0.091 -0.091
<= legal 0.091 -0.091

MENU compatability

TRUE (The compatability of land uses within the area is favourable)
FALSE (The compatability of land uses within the area is unfavourable)

HELP (a)
QUESTION (How does the compatability of the land use rate)
OPTIONS

- 1 (The land use is favourable)
- 1 (The land use is unfavourable)

YESNO aesthetic

TRUE (The aesthetic quality of the landscape will be affected)
FALSE (The aesthetic quality of the landscape will not be affected)

HELP (a)
QUESTION (Will the aesthetic quality of the landscape be affected)

SCALE place

TRUE (The sense of place within the area is highly regarded)
FALSE (The sense of place within the area is poorly regarded)

HELP (a)
QUESTION (How is the sense of place within the area regarded)
LABELS (Poorly) (Ambivalent) (Highly)

MENU character

TRUE (The character of the overall area will be affected)
FALSE (The character of the overall area will not be affected)

HELP (a)
QUESTION (How will the character of the overall area be affected)
OPTIONS

- 1 (Detrimentially)
- 0 (Slightly)
- 1 (No effect)

SCALE scale_of_develop

TRUE (The projects compatibility with the scale of developments in the area is poor)

FALSE (The projects compatibility with the scale of developments in the area is high)

HELP (a)
QUESTION (What is the projects compatibility with the scale of developments in the area)

LABELS (High) (Average) (Poor)

SCALE materials

TRUE (The projects compatibility with building materials used in the area is poor)

FALSE (The projects compatibility with building materials used in the area is high)

HELP (a)
QUESTION (What is the projects compatibility with building materials used in the area)

LABELS (High) (Average) (Poor)

YESNO Views

TRUE (Scenic views and features of the areaa will be affected)
FALSE (Scenic views and features of the areaa will not be affected)

HELP (a)
QUESTION (Will the scenic views and features of the area be affected)

YESNO revitalization

TRUE (The project will encourage the revitalization of neighbouring run-down areas)

FALSE (The project will discourage the revitalization of neighbouring run-down areas)

HELP (a)
QUESTION (Will the project encourage the revitalization of neighbouring run-down areas)

YESNO landscape

TRUE (The projects plans include providing for landscaping/site restoration)
FALSE (The projects plans exclude providing for landscaping/site restoration)

HELP (a)
QUESTION (Does the projects plans provide for landscaping of the site/site restoration)

YESNO buffer

TRUE (The site does require buffer zones to allow for natural processes)

FALSE (The site does not require buffer zones to allow for natural processes)
HELP (a)

QUESTION (Does the site require buffer zones to allow for natural processes)

YESNO claims

TRUE (There are political considerations such as land claims or historical rights)

FALSE (There are no political considerations such as land claims or historical rights)

HELP (a)

QUESTION (Are there political considerations such as land claims or historical rights)

YESNO legal

TRUE (There are legal considerations such as servitudes and rights of way)

FALSE (There are no legal considerations such as servitudes and rights of way)

HELP (a)

QUESTION (Are there legal considerations such as servitudes and rights of way)

 ***** Urban Open Space *****

SUBGOAL Open Space

TRUE (The site contains urban open space or recreational areas)

FALSE (The site contains urban open space or recreational areas)

HELP (a)

<= recreational	0.111	-0.111
<= features	0.111	-0.111
<= heritage	0.111	-0.111
<= use	0.111	-0.111
<= pressures	0.111	-0.111
<= enhancement	-0.111	0.111
<= rehabilitation	-0.111	0.111
<= public amenity	-0.111	0.111
<= crime	0.111	-0.111

YESNO recreational

TRUE (The are endangered animals species on the site)

FALSE (There are not endangered animals species on the site)

HELP (a)

QUESTION (Does the site contain endangered animals species)

YESNO features

TRUE (The diversity of the animals species will be affected)

FALSE (The diversity of the animals species will not be affected)

HELP (a)

QUESTION (Will the diversity of the animals be affected)

YESNO heritage

TRUE (The site is a natural heritage site)

FALSE (The site is not a natural heritage site)

HELP (a)

QUESTION (Is the site a natural heritage site)

YESNO use

TRUE (The site will result in a change in land use or intensity of land use)

FALSE (The site will not result in a change in land use or intensity of land use)

HELP (a)

QUESTION (Will the site result in a land use or intensity of land use)

YESNO pressures

TRUE (The site will pressurise recreational facilities or open space systems)

FALSE (The site will not pressurise recreational facilities or open space systems)

HELP (a)

QUESTION (Will the site pressurise recreational facilities or open space systems)

YESNO enhancement

TRUE (The site will enhance or link facilities and open space systems)

FALSE (The site will not enhance or link facilities and open space systems)

HELP (a)

QUESTION (Will the site enhance or link facilities and open space systems)

YESNO rehabilitation

TRUE (The site will rehabilitate existing degraded sites)

FALSE (The site will not rehabilitate existing degraded sites)

HELP (a)

QUESTION (Will the site rehabilitate existing degraded sites)

YESNO public amenity

TRUE (The project will result in an improved public amenity)

FALSE (The project will not result in an improved public amenity)

HELP (a)

QUESTION (Will the project result in an improved public amenity)

YESNO crime

TRUE (The project will have the potential for harbouring vagrants or criminals)

FALSE (The project will not have the potential for harbouring vagrants or criminals)

HELP (a)

QUESTION (Will the project have the potential for harbouring vagrants or criminals)

/*
Current and Potential Land use and Landscape Character

This Expert System Rule Base is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any of the following :

- 1) General considerations applicable to all development proposals
- 3) Residential areas

*/

DESCRIPTION (Current and Potential Land use and Landscape Character - Residential areas)

GOAL Assessment

TRUE (The current and potential landuse character is favourable)
FALSE (The current and potential landuse character is unfavourable)
HELP ()
<= General PRIORITY 100 -0.300 0.300
<= Residential PRIORITY 90 -0.700 0.700

/*
***** General *****

*/

SUBGOAL General

TRUE (The vegetation of the land will not be affected)
FALSE (The vegetation of the land will be affected)
HELP (a)
<= compatability -0.091 0.091
<= aesthetic 0.091 -0.091
<= place -0.091 0.091
<= character 0.091 -0.091
<= scale_of_develop 0.091 -0.091
<= materials 0.091 -0.091
<= views 0.091 -0.091
<= revitalization -0.091 0.091
<= landscape -0.091 0.091
<= buffer 0.091 -0.091
<= claims 0.091 -0.091
<= legal 0.091 -0.091

MENU compatability

TRUE (The compatability of land uses within the area is favourable)
FALSE (The compatability of land uses within the area is unfavourable)
HELP (a)
QUESTION (How does the compatability of the land use rate)
OPTIONS
1 (The land use is favourable)
-1 (The land use is unfavourable)

YESNO aesthetic

TRUE (The aesthetic quality of the landscape will be affected)
FALSE (The aesthetic quality of the landscape will not be affected)
HELP (a)
QUESTION (Will the aesthetic quality of the landscape be affected)

SCALE place

TRUE (The sense of place within the area is highly regarded)
FALSE (The sense of place within the area is poorly regarded)

HELP (a)
QUESTION (How is the sense of place within the area regarded)
LABELS (Poorly) (Ambivalent) (Highly)

MENU character

TRUE (The character of the overall area will be affected)
FALSE (The character of the overall area will not be affected)
HELP (a)
QUESTION (How will the character of the overall area be affected)
OPTIONS
1 (Detrimentially)
0 (Slightly)
-1 (No effect)

SCALE scale_of_develop

TRUE (The projects compatibility with the scale of developments in the area is poor)
FALSE (The projects compatibility with the scale of developments in the area is high)
HELP (a)
QUESTION (What is the projects compatibility with the scale of developments in the area)
LABELS (High) (Average) (Poor)

SCALE materials

TRUE (The projects compatibility with building materials used in the area is poor)
FALSE (The projects compatibility with building materials used in the area is high)
HELP (a)
QUESTION (What is the projects compatibility with building materials used in the area)
LABELS (High) (Average) (Poor)

YESNO Views

TRUE (Scenic views and features of the areaa will be affected)
FALSE (Scenic views and features of the areaa will not be affected)
HELP (a)
QUESTION (Will the scenic views and features of the area be affected)

YESNO revitalization

TRUE (The project will encourage the revitalization of neighbouring run-down areas)
FALSE (The project will discourage the revitalization of neighbouring run-down areas)
HELP (a)
QUESTION (Will the project encourage the revitalization of neighbouring run-down areas)

YESNO landscape

TRUE (The projects plans include providing for landscaping/site restoration)
FALSE (The projects plans exclude providing for landscaping/site restoration)
HELP (a)
QUESTION (Does the projects plans provide for landscaping of the site/site restoration)

YESNO buffer

TRUE (The site does require buffer zones to allow for natural processes)
FALSE (The site does not require buffer zones to allow for natural processes)
HELP (a)

QUESTION (Does the site require buffer zones to allow for natural processes)

YESNO claims

TRUE (There are political considerations such as land claims or historical rights)

FALSE (There are no political considerations such as land claims or historical rights)

HELP (a)

QUESTION (Are there political considerations such as land claims or historical rights)

YESNO legal

TRUE (There are legal considerations such as servitudes and rights of way)

FALSE (There are no legal considerations such as servitudes and rights of way)

HELP (a)

QUESTION (Are there legal considerations such as servitudes and rights of way)

***** Residential *****

SUBGOAL Residential

TRUE (The affect of the site on residential areas makes the site suitable for development)

FALSE (The affect of the site on residential areas makes the site unsuitable for development)

HELP (a)

<= displace	0.077	-0.077
<= lifestyle	0.077	-0.077
<= quality	0.077	-0.077
<= privacy	0.077	-0.077
<= overshadowing	0.077	-0.077
<= res compatability	-0.077	0.077
<= cohesion	0.077	-0.077
<= needs	0.077	-0.077
<= safety aspects	0.077	-0.077
<= infrastructure	-0.077	0.077
<= access	0.077	-0.077
<= traffic	0.077	-0.077
<= tax	0.077	-0.077

YESNO displace

TRUE (The site will require the displacement of people/affect existing housing)

FALSE (The site will not require the displacement of people/affect existing housing)

HELP (a)

QUESTION (Will the site require the displacement of people or affect existing housing)

YESNO lifestyle

TRUE (The site will affect the lifestyle, neighbourhood character or stability)

FALSE (The site will not affect the lifestyle, neighbourhood character or stability)

HELP (a)

QUESTION (Will the site affect the lifestyle, neighbourhood character or stability)

YESNO quality

TRUE (The site will affect the quality of life within the neighbourhood)

FALSE (The site will not affect the quality of life within the neighbourhood)

HELP (a)

QUESTION (Will the site affect the quality of life within the neighbourhood)

YESNO privacy

TRUE (The site will affect views or privacy within the neighbourhood)

FALSE (The site will not affect views or privacy within the neighbourhood)

HELP (a)

QUESTION (Will the site affect views or privacy within the neighbourhood)

YESNO overshadowing

TRUE (The site will cause overshadowing and loss of sunlight hours)

FALSE (The site will not cause overshadowing and loss of sunlight hours)

HELP (a)

QUESTION (Will the site cause overshadowing and loss of sunlight hours)

YESNO res compatability

TRUE (The development is compatable with surrounding residential developments)

FALSE (The development is not compatable with surrounding residential development s)

HELP (a)

QUESTION (Is the development compatable with surrounding residential developments)

YESNO cohesion

TRUE (The development will affect community cohesion)

FALSE (The development will not affect community cohesion)

HELP (a)

QUESTION (Will the development affect community cohesion)

YESNO needs

TRUE (The site does affect the needs of the elderly, handicapped or other special interest groups)

FALSE (The site does not affect the needs of the elderly, handicapped or other special interest groups)

HELP (a)

QUESTION (Does the site affect the needs of the elderly, handicapped or other special interest groups)

YESNO safety aspects

TRUE (The site does affect community safety aspects such as lighting, open areas and policing)

FALSE (The site does not affect community safety aspects such as lighting, open areas and policing)

HELP (a)

QUESTION (Does the site affect community safety aspects such as lighting, open areas and policing)

YESNO infrastructure

TRUE (There is an adequacy of infrastructure to service the area)

FALSE (There is an inadequacy of infrastructure to service the area)

HELP (a)

QUESTION (Is there an adequacy of infrastructure to service the area)

YESNO access

TRUE (The development will affect access and movement patterns in the neighbourh ood)

FALSE (The development will not affect access and movement patterns in the neighb ourhood)

HELP (a)

QUESTION (Will the development affect access and movement patterns in the neighbo

urhood}

YESNO traffic

TRUE {The development will change the volume of through traffic in the neighbourhood}

FALSE {The development will not change the volume of through traffic in the neighbourhood}

HELP {a}

QUESTION {Will the development change the volume of through traffic in the neighbourhood}

YESNO tax

TRUE {The development will affect property values and the local tax base}

FALSE {The development will not affect property values and the local tax base}

HELP {a}

QUESTION {Will the development affect property values and the local tax base}

/*

Cultural Resources of the Site and Its Surroundings

This Expert System Rule Base is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any cultural or historic heritage.

*/

DESCRIPTION (Cultural Resources of the Site and Its Surroundings)

GOAL Assessment

TRUE (The cultural characteristics of the site is favourable)

FALSE (The cultural characteristics of the site is unfavourable)

HELP { }

<= Historic PRIORITY 100 0.200 -0.200

<= Archaeological PRIORITY 90 0.200 -0.200

<= Special PRIORITY 80 0.200 -0.200

<= Religious PRIORITY 70 0.200 -0.200

<= Social PRIORITY 60 0.200 -0.200

/*****
/***** All
/*****/

YESNO Historic

TRUE (The site is of historic heritage)

FALSE (The site is not of historic heritage)

HELP {a}

QUESTION (Is the site of historic heritage)

YESNO Archaeological

TRUE (The site is of archaeological importance)

FALSE (The site is not of archaeological importance)

HELP {a}

QUESTION (Is the site of archaeological importance)

YESNO Special

TRUE (The site is of special heritage)

FALSE (The site is not of special heritage)

HELP {a}

QUESTION (Is the site of special heritage)

YESNO religious

TRUE (The site is of religious heritage)

FALSE (The site is not of religious heritage)

HELP {a}

QUESTION (Is the site of religious heritage)

YESNO social

TRUE (The site is of social heritage)

FALSE (The site is not of social heritage)

HELP {a}

QUESTION (Is the site of social heritage)

/*
Ecological Characteristics of the Site and Its Surroundings

This Expert System Rule Base is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any of the following :

- 1) The Vegetation
 - 2) The Presence of Animals
 - 3) Natural and Semi-natural Communities
- */

DESCRIPTION (Ecological Characteristics of the Site and Its Surroundings)

GOAL Assessment

TRUE (The ecological characteristics of the site is favourable)
FALSE (The ecological characteristics of the site is unfavourable)
HELP ()
 <= Vegetation PRIORITY 100 -0.40 0.40
 <= Animals PRIORITY 90 -0.40 0.40
 <= Communities PRIORITY 80 -0.20 0.20

***** The Vegetation *****

SUBGOAL Vegetation

TRUE (The vegetation of the land will not be affected)
FALSE (The vegetation of the land will be affected)
HELP (a)

<= endangered species	0.050	-0.050
<= diversity of communities	-0.050	0.050
<= sand trapping vegetation	0.050	-0.050
<= communities of importance	0.050	-0.050
<= commun. of recreational value	0.050	-0.050
<= alien plants	0.050	-0.050
<= natural replenishment	0.050	-0.050
<= veld fires	0.050	-0.050
<= off road vehicles	0.050	-0.050
<= trampling of vegetation	0.050	-0.050
<= firewood collection	0.050	-0.050
<= overgrazing	0.050	-0.050
<= over exploitation	0.050	-0.050
<= loss in area	0.350	-0.350

SCALE loss in area

TRUE (There will be a large net loss in area of natural vegetation)
FALSE (There will be no net loss in area of natural vegetation)
HELP (a)

QUESTION (To what extent will there be a net loss in area of natural vegetation)

LABELS {None} {Small Area} {Large Area}

YESNO endangered species

TRUE (The are endangered plant species on the site)
FALSE (There are not endangered plant species on the site)
HELP (a)
QUESTION (Does the site contain endangered plant species)

YESNO diversity of communities

TRUE (The diversity of the plant species will not be affected)
FALSE (The diversity of the plant species will be affected)
HELP (a)
QUESTION (Will the diversity of the vegetation be affected)

YESNO sand trapping vegetation

TRUE (The site will have an impact on sand trapping vegetation)
FALSE (The site have no an impact on sand trapping vegetation)
HELP (a)
QUESTION (Will the site have an impact on sand trapping vegetation)

YESNO communities of importance

TRUE (There are plant communities of conservation or scientific importance)
FALSE (There are no plant communities of conservation or scientific importance)
HELP (a)
QUESTION (Are there plant communities of conservation or scientific importance)

YESNO commun. of recreational value

TRUE (There are plant communities of particular recreational value)
FALSE (There are no plant communities of particular recreational value)
HELP (a)
QUESTION (Are there plant communities of particular recreational value)

YESNO alien plants

TRUE (The site will cause the spread of invasive alien seeds and plants)
FALSE (The site will not cause the spread of invasive alien seeds and plants)
HELP (a)
QUESTION (Will the site cause the spread of invasive alien seeds and plants)

YESNO natural replenishment

TRUE (The site will affect the replenishment of existing species)
FALSE (The site will not affect the replenishment of existing species)
HELP (a)
QUESTION (Will the site affect the replenishment of existing species)

YESNO veld fires

TRUE (The site will enhance the likelihood of veldfires)
FALSE (The site will not enhance the likelihood of veldfires)
HELP (a)
QUESTION (Could the site enhance the likelihood of veldfires)

YESNO off road vehicles

TRUE (Offroad vehicles are more likely to damage the vegetation)
FALSE (Offroad vehicles are not more likely to damage the vegetation)
HELP (a)
QUESTION (Are offroad vehicles in the area more likely to damage the vegetation)

YESNO trampling of vegetation

TRUE (The site will increase the trampling of special areas of vegetation)
FALSE (The site will not increase the trampling of special areas of vegetation)
HELP (a)
QUESTION (Will the site increase the trampling of special areas of vegetation)

YESNO firewood collection

TRUE (The site will increase the amount of firewood collection)
FALSE (The site will not increase the amount of firewood collection)
HELP (a)
QUESTION (Will the site increase the amount of firewood collection)

YESNO overgrazing
 TRUE {The project will lead to an increase in overgrazing}
 FALSE {The project will not lead to an increase in overgrazing}
 HELP (a)
 QUESTION {Will the project lead to an increase in overgrazing}

YESNO over exploitation
 TRUE {The project will lead to overexploitation of the land}
 FALSE {The project will not lead to overexploitation of the land}
 HELP (a)
 QUESTION {Will the project lead to overexploitation of the land}

 /***** Animals *****/

SUBGOAL animals
 TRUE {The animals of the land will not be affected }
 FALSE { The animals of the land will be affected }
 HELP (a)

<= endangered animals	0.068	-0.068
<= diversity of animals	-0.068	0.068
<= scientific importance	0.068	-0.068
<= natural migration	0.068	-0.068
<= animals of recreation	0.068	-0.068
<= loss in animal numbers	0.455	-0.455
<= migrant species	0.068	-0.068
<= alien species	0.068	-0.068
<= survival of communities	0.068	-0.068

SCALE loss in animal numbers
 TRUE {There will be a large net loss in the number of wild animals}
 FALSE {There will be no net loss in the number of wild animals}
 HELP (a)
 QUESTION {To what extent will there be a net loss in the number of wild animals }
 LABELS {None} {Small No's} {Large No's}

YESNO endangered animals
 TRUE {The are endangered animals species on the site}
 FALSE {There are not endangered animals species on the site}
 HELP (a)
 QUESTION {Does the site contain endangered animals species}

YESNO diversity of animals
 TRUE {The diversity of the animals species will not be affected}
 FALSE {The diversity of the animals species will be affected}
 HELP (a)
 QUESTION {Will the diversity of the animals be affected}

YESNO scientific importance
 TRUE {There are animals on site with scientific or educational value}
 FALSE {There are no animals on site with scientific or educational value}
 HELP (a)
 QUESTION {Are there animals on site with scientific or educational value}

YESNO natural migration
 TRUE {The project will affect the natural migration routes of species}
 FALSE {The project will not affect the natural migration routes of species}
 HELP (a)

QUESTION {Will the project affect the natural migration routes of species}

YESNO animals of recreation
 TRUE {The project will affect the survival of animal communities of recreational value}
 FALSE {The project will not affect the survival of animal communities of recreational value}
 HELP (a)
 QUESTION {Will the project affect the survival of animal communities of recreational value}

YESNO migrant species
 TRUE {There are migrant or non-resident species that make use of the site}
 FALSE {There are no migrant or non-resident species that make use of the site}
 HELP (a)
 QUESTION {Are there migrant or non-resident species that make use of the site}

YESNO alien species
 TRUE {There are alien or domestic species living on the site}
 FALSE {There are no alien or domestic species living on the site}
 HELP (a)
 QUESTION {Are there alien or domestic species living on the site}

YESNO survival of communities
 TRUE {The project will affect the survival of animal communities in general}
 FALSE {The project will not affect the survival of animal communities in general}
 HELP (a)
 QUESTION {Will the project affect the survival of animal communities in general}

 /***** Communities *****/

SUBGOAL Communities
 TRUE {The absence of local communities/ecosystems makes the site suitable for development }
 FALSE {The presence of local communities/ecosystems makes the site unsuitable for development }
 HELP (a)

<= importance of natural communities	0.250	-0.250
<= development compatability	-0.250	0.250
<= appropriateness of methods	-0.250	0.250
<= ecological functioning	-0.250	0.250

YESNO importance of natural communities
 TRUE {The natural communities on the site do have local or national importance}
 FALSE {The natural communities on the site do not have local or national importance}
 HELP (a)
 QUESTION {Do the natural communities / ecosystems on the site have local or national importance}

YESNO development compatability
 TRUE {The project is in compatability with natural communities}
 FALSE {The project is not in compatability with natural communities}
 HELP (a)
 QUESTION {Is the project in compatability with natural communities / ecosystems}

YESNO appropriateness of methods

TRUE (The conservation methods to be employed are appropriate)
FALSE (The conservation methods to be employed are not appropriate)
HELP (a)
QUESTION (Are the conservation methods to be employed appropriate)

YESNO ecological functioning
TRUE (The ecological functioning of the community will not be affected)
FALSE (The ecological functioning of the community will be affected)
HELP (a)
QUESTION (Will the development adversely affect the ecological functioning
of wildlife communities)

/*

Energy Infrastructure Services

This expert system rulebase assesses the impact of the proposed project on energy supply infrastructure services.

*/

DESCRIPTION (Energy Infrastructure Services)

GOAL Assessment

TRUE (The development will produce very little effect on energy supply)

FALSE (The development will produce a significant effect on energy supply)

HELP {}

<= demand for power	PRIORITY 100	0.167 -0.167
<= planned provision	PRIORITY 90	-0.167 0.167
<= new transmission lines	PRIORITY 70	0.167 -0.167
<= emergency facilities	PRIORITY 60	-0.167 0.167
<= danger of power failure	PRIORITY 50	0.167 -0.167
<= alternative fuels	PRIORITY 40	-0.167 0.167

SCALE demand for power

TRUE (The proposal will have a large effect on the demand for power and associated effects on peak and base loads)

FALSE (The proposal will have very little effect on the demand for power and associated effects on peak and base loads)

HELP {a}

QUESTION (To what extent will the proposal effect demand for power and associated effects on peak and base loads)

LABELS {Little} {Average} {Large}

YESNO planned provision

TRUE (There is planned provision of power to this area)

FALSE (There is no planned provision of power to this area)

HELP {a}

QUESTION (Is there planned provision of power to the area)

YESNO new transmission lines

TRUE (New transmission lines will be required for the area)

FALSE (New transmission lines will not be required for the area)

HELP {a}

QUESTION (Will new transmission lines be required for the area)

MENU emergency facilities

TRUE (Current emergency power facilities are more than adequate)

FALSE (Current emergency power facilities are very inadequate)

HELP {a}

QUESTION (To what extent are current emergency power facilities adequate)

OPTIONS

1 (Very Adequate)

0 (OK)

-1 (Hopeless)

SCALE danger of power failure

TRUE (The extent of danger to the community in the case of a major power failure is high)

FALSE (The extent of danger to the community in the case of a major power failure is low)

HELP {a}

QUESTION (To what extent is there danger to the local community, environment or

processing units should there be a major power failure)
 LABELS {Low} {Average} {High}

YESNO alternative fuels

TRUE (Alternative fuel sources are available)

FALSE (Alternative fuel sources are not available)

HELP {a}

QUESTION (Are alternative fuel sources available for the project)

/*
Health and Safety

This expert system rulebase assesses the impact of the proposed project on health and safety aspects.

*/

DESCRIPTION (Health and Safety)

GOAL Assessment

TRUE (The health and safety characteristics of the project are favourable)

FALSE (The health and safety characteristics of the project are unfavourable)

HELP ()

<= effects in workplace PRIORITY 100 0.200 -0.200

<= effects in surrounds PRIORITY 90 0.450 -0.450

<= effect on recreation PRIORITY 90 0.350 -0.350

SCALE effects in workplace

TRUE (The extent to which workers will be effected is high)

FALSE (The extent to which workers will be effected is low)

HELP (a)

QUESTION (To what extent will workers be affected by noise, odours,
vapours, gasses, temperature, noise, radiation etc)

LABELS (Low) (Average) (High)

SCALE effects in surrounds

TRUE (The extent to which surrounding areas will be effected is high)

FALSE (The extent to which surrounding areas will be effected is low)

HELP (a)

QUESTION (To what extent will the surrounding area be effected by noise, odours,
vapours, gasses, fumes, noise, radiation etc)

LABELS (Low) (Average) (High)

SCALE effect on recreation

TRUE (Public recreational areas will be greatly affected)

FALSE (Public recreational areas will not be affected)

HELP (a)

QUESTION (To what extent will public recreational areas be affected by
any pollution)

LABELS (None) (Average) (High)

/*
This Expert System Rule Base is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any of the following :

- 1) The Nature of the Land
- 2) The Presence of Marine and Esturine Systems
- 3) The Nature of the Climate

*/

DESCRIPTION (Physical Characteristics of the site and Surroundings -
Land, Climate and Marine and Esturine Systems)

GOAL Assessment

TRUE (The physical characteristics of the site is favourable)
FALSE (The physical characteristics of the site is unfavourable)

HELP ()

<= Land	PRIORITY 80	-0.33	0.33
<= MarineSystems	PRIORITY 70	-0.33	0.33
<= Climate	PRIORITY 60	-0.33	0.33

/*
***** The Nature of the Land *****

SUBGOAL Land

TRUE (The topography of the land is suitable for development)
FALSE (The topography of the land is not suitable for development)

HELP (a)

<= surface	0.048	-0.048
<= substrata	0.048	-0.048
<= unstable rock	0.048	-0.048
<= seismic	0.048	-0.048
<= slope	0.048	-0.048
<= waterlogging	0.048	-0.048
<= soil bonding	0.048	-0.048
<= site stability	-0.048	0.048
<= subsistence	0.048	-0.048
<= soil strength	-0.048	0.048
<= erosion	-0.048	0.048
<= agri potential	0.048	-0.048
<= comm potential	-0.048	0.048
<= minerals	0.048	-0.048
<= cons. materials	-0.048	0.048
<= topsoil avail	-0.048	0.048
<= spoil management	-0.048	0.048
<= unique features	0.048	-0.048
<= sand dunes	0.048	-0.048
<= prom features	0.048	-0.048
<= exist degradation	-0.048	0.048

YESNO surface

TRUE (The project will be affected by the nature of the ground surface)
FALSE (The project will not be affected by the nature of the ground surface)

HELP (a)

QUESTION (Will the project be affected by the nature of the ground surface)

YESNO substrata

TRUE (The project will be affected by the nature of the rock substrata)
FALSE (The project will not be affected by the nature of the rock substrata)

HELP (a)

QUESTION (Will the project be affected by the nature of the rock substrata)

YESNO unstable rock

TRUE (There is unstable rock or faultlines on the site)
FALSE (There are no unstable rock or faultlines on the site)

HELP (a)

QUESTION (Is there unstable rock or faultlines on the site)

YESNO seismic

TRUE (Seismic activity is a problem)
FALSE (Seismic activity is not a problem)

HELP (a)

QUESTION (Is seismic activity a problem)

YESNO slope

TRUE (The slope of the land is too steep)
FALSE (The slope of the land is not too steep)

HELP (a)

QUESTION (Is the slope of the land too steep)

YESNO waterlogging

TRUE (There is a problem with waterlogging or depressions on the site)
FALSE (There is no problem with waterlogging or depressions on the site)

HELP (a)

QUESTION (Is there a problem with waterlogging or depressions on the site)

MENU soil bonding

TRUE (The soils on the site are bonded or binded in some way)
FALSE (The soils on the site are not bonded or binded in any way)

HELP (a)

QUESTION (Are the soils on the site bonded or binded in any way)

OPTIONS

1 (Bonded or binded eg clay)

0 (Mixture)

-1(Unbonded soils eg. sand)

YESNO site stability

TRUE (The site is stable)
FALSE (The site is not stable)

HELP (a)

QUESTION (Is the site stable)

YESNO subsistence

TRUE (There is evidence of surface subsistence on the site)
FALSE (There is no evidence of surface subsistence on the site)

HELP (a)

QUESTION (Is there evidence of surface subsistence on the site)

YESNO soil strength

TRUE (The compressive strength of the soils is adequate)
FALSE (The compressive strength of the soils is not adequate)

HELP (a)

QUESTION (Is the compressive strength of the soils adequate)

YESNO erosion

TRUE (Erosion is not a problem)
FALSE (Erosion is a problem)

HELP (a)

QUESTION (Will the site be protected against erosion)

YESNO agri potential
 TRUE (The soils on site have a good potential for use by agriculture)
 FALSE (The soils on site do not have a good potential for use by agriculture)
 HELP (a)
 QUESTION (Does the soils on site have a good potential for use by agriculture)

YESNO comm potential
 TRUE (The soils on site have a good potential for use by commerce)
 FALSE (The soils on site do not have a good potential for use by commerce)
 HELP (a)
 QUESTION (Does the soils on site have a good potential for use by commerce)

YESNO minerals
 TRUE (There are mineral deposits on the site)
 FALSE (There are no mineral deposits on the site)
 HELP (a)
 QUESTION (Are there mineral deposits on the site)

YESNO cons. materials
 TRUE (There is access or availability to construction materials)
 FALSE (There is no access or availability to construction materials)
 HELP (a)
 QUESTION (Is there access or availability to construction materials)

YESNO topsoil avail
 TRUE (There is availability of topsoil or fill material)
 FALSE (There is no availability of topsoil or fill material)
 HELP (a)
 QUESTION (Is there availability of topsoil or fill material)

YESNO spoil management
 TRUE (Spoil material will be managed properly)
 FALSE (Spoil material will not be managed properly)
 HELP (a)
 QUESTION (Will spoil material be managed properly)

YESNO unique features
 TRUE (There are unique physical or geological features on the site)
 FALSE (There are no unique physical or geological features on the site)
 HELP (a)
 QUESTION (Are there unique physical or geological features on the site)

YESNO sand dunes
 TRUE (There are mobile sanddunes on the site)
 FALSE (There are no mobile sanddunes on the site)
 HELP (a)
 QUESTION (Are there mobile sanddunes on the site)

YESNO prom features
 TRUE (There are prominent landscape features on the site)
 FALSE (There are no prominent landscape features on the site)
 HELP (a)
 QUESTION (Are there any prominent landscape features on the site)

YESNO exist degradation
 TRUE (There is existing physical degradation of the local environment)
 FALSE (There is no existing physical degradation of the local environment)
 HELP (a)
 QUESTION (Is there existing physical degradation of the local environment)

/***** Climate *****/
 /***** Climate *****/
 /***** Climate *****/

SUBGOAL climate
 TRUE (The climate of the area favours development of the site)
 FALSE (The climate of the area does not favour development of the site)
 HELP (a)
 <= wind strength 0.143 -0.143
 <= flash floods 0.143 -0.143
 <= rainfall patterns 0.143 -0.143
 <= temperature fluks 0.143 -0.143
 <= inversions 0.143 -0.143
 <= pollutants 0.143 -0.143
 <= global warming 0.143 -0.143

YESNO wind strength
 TRUE (The site will be adversely affected by the wind strength)
 FALSE (The site will not be adversely affected by the wind strength)
 HELP (a)
 QUESTION (Will the development be affected by the wind strength)

YESNO flash floods
 TRUE (The site will be affected by flash floods)
 FALSE (The site will not be affected by flash floods)
 HELP (a)
 QUESTION (Will the site be affected by flash floods)

YESNO rainfall patterns
 TRUE (The site will adversely affect the rainfall patterns)
 FALSE (The site will not adversely affect the rainfall patterns)
 HELP (a)
 QUESTION (Will the development adversely affect the rainfall patterns)

YESNO temperature fluks
 TRUE (The project will be affected by temperature flucuations)
 FALSE (The project will not be affected by temperature flucuations)
 HELP (a)
 QUESTION (Will the project be affected by temperature flucuations)

YESNO inversions
 TRUE (The project will be affected by the intensity of inversions)
 FALSE (The project will not be affected by the intensity of inversions)
 HELP (a)
 QUESTION (Will the project be affected by the intensity of inversions)

YESNO pollutants
 TRUE (The project will cause an influx of pollutants)
 FALSE (The project will not cause an influx of pollutants)
 HELP (a)
 QUESTION (Will the project cause an influx of pollutants)

YESNO global warming
 TRUE (The site will enhance global warming or a rise in sea level)
 FALSE (The site will not enhance global warming or a rise in sea level)
 HELP (a)
 QUESTION (Will the site enhance global warming or a rise in sea level)

/***** Climate *****/

/***** Marine Systems * *****/
 /*****/

SUBGOAL MarineSystems

TRUE {The absence of marine systems makes the site suitable for development }

FALSE {The presence of marine systems makes the site unsuitable for development }

HELP {a}

<= coastal_features	0.111	-0.111
<= altered_processes	0.111	-0.111
<= unstable_ecosystems	0.111	-0.111
<= sand_sources	0.111	-0.111
<= shorelines	0.111	-0.111
<= seabed	0.111	-0.111
<= coastal_islands	0.111	-0.111
<= estuary_systems	0.111	-0.111
<= river_mouths	0.111	-0.111

YESNO coastal_features

TRUE {The site will adversely affect coastal features}

FALSE {The site will not adversely affect coastal features}

HELP {a}

QUESTION {Will the site adversely affect coastal features}

YESNO altered_processes

TRUE {There are existing or altered processes affecting the project}

FALSE {There are no existing or altered processes affecting the project}

HELP {a}

QUESTION {Are there existing or altered processes affecting the project}

YESNO unstable_ecosystems

TRUE {There are inherently unstable ecosystems such as mobile sanddunes}

FALSE {There are no inherently unstable ecosystems such as mobile sanddunes}

HELP {a}

QUESTION {Are there inherently unstable ecosystems such as mobile sanddunes}

YESNO sand_sources

TRUE {There are sand sources such as mobile sand dunes}

FALSE {There are no sand sources such as mobile sand dunes}

HELP {a}

QUESTION {Are there sand sources such as mobile sand dunes}

YESNO shorelines

TRUE {The rocky or sandy shorelines will affect the project}

FALSE {The rocky or sandy shorelines will not affect the project}

HELP {a}

QUESTION {Will the rocky or sandy shorelines affect the project}

YESNO seabed

TRUE {The site will adversely affect the seabed}

FALSE {The site will not adversely affect the seabed}

HELP {a}

QUESTION {Will the development adversely affect the seabed}

YESNO coastal_islands

TRUE {Costal islands will be affected by the project}

FALSE {Costal islands will not be affected by the project}

HELP {a}

QUESTION {Will costal islands be affected by the project}

YESNO estuary_systems

TRUE {The site will affect the functioning of estuary systems}

FALSE {The site will not affect the functioning of estuary systems}

HELP {a}

QUESTION {Will the site affect the functioning of estuary systems}

YESNO river_mouths

TRUE {The site will adversely affect a river mouth}

FALSE {The site will not adversely affect a river mouth}

HELP {a}

QUESTION {Will the development adversely affect a river mouth}

/*
This Expert System Rule Base is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any of the following :

- 1) The Nature of the Land
- 2) The Presence of Fresh water systems
- 3) The Nature of the Climate

*/

DESCRIPTION (Physical Characteristics of the site and Surroundings -
Land, Climate and Freshwater Systems)

GOAL Assessment

TRUE (The physical characteristics of the site is favourable)
FALSE (The physical characteristics of the site is unfavourable)

HELP ()

<= Land	PRIORITY 90	-0.33	0.33
<= FreshWaterSystems	PRIORITY 80	-0.33	0.33
<= Climate	PRIORITY 70	-0.33	0.33

/*
***** The Nature of the Land *****

SUBGOAL Land

TRUE (The topography of the land is suitable for development)
FALSE (The topography of the land is not suitable for development)

HELP (a)

<= surface	0.048	-0.048
<= substrata	0.048	-0.048
<= unstable rock	0.048	-0.048
<= seismic	0.048	-0.048
<= slope	0.048	-0.048
<= waterlogging	0.048	-0.048
<= soil bonding	-0.048	0.048
<= site stability	-0.048	0.048
<= subsistence	0.048	-0.048
<= soil strength	-0.048	0.048
<= erosion	-0.048	0.048
<= agri potential	0.048	-0.048
<= comm potential	-0.048	0.048
<= minerals	0.048	-0.048
<= cons. materials	-0.048	0.048
<= topsoil avail	-0.048	0.048
<= spoil management	-0.048	0.048
<= unique features	0.048	-0.048
<= sand dunes	0.048	-0.048
<= prom features	0.048	-0.048
<= exist degradation	-0.048	0.048

YESNO surface

TRUE (The project will be affected by the nature of the ground surface)
FALSE (The project will not be affected by the nature of the ground surface)

HELP (a)

QUESTION (Will the project be affected by the nature of the ground surface)

YESNO substrata

TRUE (The project will be affected by the nature of the rock substrata)
FALSE (The project will not be affected by the nature of the rock substrata)

HELP (a)

QUESTION (Will the project be affected by the nature of the rock substrata)

YESNO unstable rock

TRUE (There is unstable rock or faultlines on the site)

FALSE (There are no unstable rock or faultlines on the site)

HELP (a)

QUESTION (Is there unstable rock or faultlines on the site)

YESNO seismic

TRUE (Seismic activity is a problem)

FALSE (Seismic activity is not a problem)

HELP (a)

QUESTION (Is seismic activity a problem)

YESNO slope

TRUE (The slope of the land is too steep)

FALSE (The slope of the land is not too steep)

HELP (a)

QUESTION (Is the slope of the land too steep)

YESNO waterlogging

TRUE (There is a problem with waterlogging or depressions on the site)

FALSE (There is no problem with waterlogging or depressions on the site)

HELP (a)

QUESTION (Is there a problem with waterlogging or depressions on the site)

MENU soil bonding

TRUE (The soils on the site are bonded or binded in some way)

FALSE (The soils on the site are not bonded or binded in any way)

HELP (a)

QUESTION (Are the soils on the site bonded or binded in any way)

OPTIONS

1 (Bonded or binded eg clay)

0 (Mixture)

-1(Unbonded soils eg. sand)

YESNO site stability

TRUE (The site is stable)

FALSE (The site is not stable)

HELP (a)

QUESTION (Is the site stable)

YESNO subsistence

TRUE (There is evidence of surface subsistence on the site)

FALSE (There is no evidence of surface subsistence on the site)

HELP (a)

QUESTION (Is there evidence of surface subsistence on the site)

YESNO soil strength

TRUE (The compressive strength of the soils is adequate)

FALSE (The compressive strength of the soils is not adequate)

HELP (a)

QUESTION (Is the compressive strength of the soils adequate)

YESNO erosion

TRUE (Erosion is not a problem)

FALSE (Erosion is a problem)

HELP (a)

QUESTION (Will the site be protected against erosion)

YESNO agri potential
 TRUE (The soils on site have a good potential for use by agriculture)
 FALSE (The soils on site do not have a good potential for use by agriculture)
 HELP (a)
 QUESTION (Does the soils on site have a good potential for use by agriculture)

YESNO comm potential
 TRUE (The soils on site have a good potential for use by commerce)
 FALSE (The soils on site do not have a good potential for use by commerce)
 HELP (a)
 QUESTION (Does the soils on site have a good potential for use by commerce)

YESNO minerals
 TRUE (There are mineral deposits on the site)
 FALSE (There are no mineral deposits on the site)
 HELP (a)
 QUESTION (Are there mineral deposits on the site)

YESNO cons. materials
 TRUE (There is access or availability to construction materials)
 FALSE (There is no access or availability to construction materials)
 HELP (a)
 QUESTION (Is there access or availability to construction materials)

YESNO topsoil avail
 TRUE (There is availability of topsoil or fill material)
 FALSE (There is no availability of topsoil or fill material)
 HELP (a)
 QUESTION (Is there availability of topsoil or fill material)

YESNO spoil management
 TRUE (Spoil material will be managed properly)
 FALSE (Spoil material will not be managed properly)
 HELP (a)
 QUESTION (Will spoil material be managed properly)

YESNO unique features
 TRUE (There are unique physical or geological features on the site)
 FALSE (There are no unique physical or geological features on the site)
 HELP (a)
 QUESTION (Are there unique physical or geological features on the site)

YESNO sand dunes
 TRUE (There are mobile sanddunes on the site)
 FALSE (There are no mobile sanddunes on the site)
 HELP (a)
 QUESTION (Are there mobile sanddunes on the site)

YESNO prom features
 TRUE (There are prominent landscape features on the site)
 FALSE (There are no prominent landscape features on the site)
 HELP (a)
 QUESTION (Are there any prominent landscape features on the site)

YESNO exist degradation
 TRUE (There is existing physical degradation of the local environment)
 FALSE (There is no existing physical degradation of the local environment)
 HELP (a)
 QUESTION (Is there existing physical degradation of the local environment)

/***** Climate *****/
 /***** Climate *****/
 /***** Climate *****/

SUBGOAL climate
 TRUE (The climate of the area favours development of the site)
 FALSE (The climate of the area does not favour development of the site)
 HELP (a)
 <= wind strength 0.143 -0.143
 <= flash floods 0.143 -0.143
 <= rainfall_patterns 0.143 -0.143
 <= temperature fluks 0.143 -0.143
 <= inversions 0.143 -0.143
 <= pollutants 0.143 -0.143
 <= global warming 0.143 -0.143

YESNO wind_strength
 TRUE (The site will be adversely affected by the wind strength)
 FALSE (The site will not be adversely affected by the wind strength)
 HELP (a)
 QUESTION (Will the development be affected by the wind strength)

YESNO flash floods
 TRUE (The site will be affected by flash floods)
 FALSE (The site will not be affected by flash floods)
 HELP (a)
 QUESTION (Will the site be affected by flash floods)

YESNO rainfall_patterns
 TRUE (The site will adversely affect the rainfall patterns)
 FALSE (The site will not adversely affect the rainfall patterns)
 HELP (a)
 QUESTION (Will the development adversely affect the rainfall patterns)

YESNO temperature fluks
 TRUE (The project will be affected by temperature flucuations)
 FALSE (The project will not be affected by temperature flucuations)
 HELP (a)
 QUESTION (Will the project be affected by temperature flucuations)

YESNO inversions
 TRUE (The project will be affected by the intensity of inversions)
 FALSE (The project will not be affected by the intensity of inversions)
 HELP (a)
 QUESTION (Will the project be affected by the intensity of inversions)

YESNO pollutants
 TRUE (The project will cause an influx of pollutants)
 FALSE (The project will not cause an influx of pollutants)
 HELP (a)
 QUESTION (Will the project cause an influx of pollutants)

YESNO global warming
 TRUE (The site will enhance global warming or a rise in sea level)
 FALSE (The site will not enhance global warming or a rise in sea level)
 HELP (a)
 QUESTION (Will the site enhance global warming or a rise in sea level)

/***** Climate *****/

***** The Nature of the Fresh Water Systems *****/
 *****/

SUBGOAL FreshWaterSystems

TRUE (The absence of fresh water systems makes the site suitable for development)

FALSE (The presence of fresh water systems makes the site unsuitable for development)

HELP (a)

<= streams	0.083	-0.083
<= river flow	0.083	-0.083
<= natural patterns	0.083	-0.083
<= eng patterns	-0.083	0.083
<= drainage limits	0.083	-0.083
<= watertable	0.083	-0.083
<= runoff	-0.083	0.083
<= absorb runoff	-0.083	0.083
<= floodplains	0.083	-0.083
<= quality water	0.083	-0.083
<= rec value water	0.083	-0.083
<= threats	0.083	-0.083

YESNO streams

TRUE (There are streams or river channels on the site)

FALSE (There are no streams or river channels on the site)

HELP (a)

QUESTION (Are there any streams or river channels on the site)

YESNO river flow

TRUE (There is significant river flow through the site)

FALSE (There is no significant river flow through the site)

HELP (a)

QUESTION (Is there any significant river flow through the site)

YESNO natural patterns

TRUE (There are natural drainage patterns on the site)

FALSE (There are no natural drainage patterns on the site)

HELP (a)

QUESTION (Are there natural drainage patterns on the site)

YESNO eng patterns

TRUE (There are engineered drainage patterns on the site)

FALSE (There are no engineered drainage patterns on the site)

HELP (a)

QUESTION (Are there any engineered drainage patterns on the site)

YESNO drainage limits

TRUE (The site does have drainage limitations)

FALSE (The site does not have drainage limitations)

HELP (a)

QUESTION (Does the site have drainage limitations)

YESNO watertable

TRUE (The watertable is close to the surface)

FALSE (The water table is not close to the surface)

HELP (a)

QUESTION (Is the watertable close to the surface of the land)

YESNO runoff

TRUE (There will be no increase in runoff)

FALSE (There will be a increased in runoff)

HELP (a)

QUESTION (Will the runoff be increased by the development of the site)

YESNO absorb runoff

TRUE (The ground does have the ability to absorb runoff)

FALSE (The ground does not have the ability to absorb runoff)

HELP (a)

QUESTION (Does the ground have the ability to absorb runoff)

YESNO floodplains

TRUE (The site will change the flood plain in some way)

FALSE (The site will not change the flood plain in some way)

HELP (a)

QUESTION (Will the site change the flood plain in some way)

YESNO quality water

TRUE (The site will affect the quantity or quality of the water on the site)

FALSE (The site will not affect the quantity or quality of the water on the site)

HELP (a)

QUESTION (Will the site affect the quantity or quality of the water on the site)

YESNO rec value water

TRUE (The site will affect the recreational or conservational value of the water)

FALSE (The site will not affect the recreational or conservational value of the water)

HELP (a)

QUESTION (Will the site affect the recreational or conservational value of the water)

YESNO threats

TRUE (The hydrological functioning of the waterway will be affected)

FALSE (The hydrological functioning of the waterway will not be affected)

HELP (a)

QUESTION (Will the hydrological functioning of the waterway be affected)

```
/*
This Expert System Rule Base is specifically designed to investigate
whether the proposed development could have a significant impact on,
or be constrained by any of the following :
```

- 1) The Nature of the Land
- 2) The Nature of the Climate

```
*/
DESCRIPTION ( Physical Characteristics of the site and Surroundings -
Land and Climate )
```

GOAL Assessment

```
TRUE ( The physical characteristics of the site is favourable)
FALSE ( The physical characteristics of the site is unfavourable )
```

```
HELP ( )
```

```
<= Land          PRIORITY 20  -0.50  0.50
<= Climate       PRIORITY 10  -0.50  0.50
```

```

/***** The Nature of the Land *****/
/***** The Nature of the Land *****/
/***** The Nature of the Land *****/
```

SUBGOAL Land

```
TRUE (The topography of the land is suitable for development )
FALSE ( The topography of the land is not suitable for development )
```

```
HELP (a)
```

```
<= surface      0.048  -0.048
<= substrata    0.048  -0.048
<= unstable rock 0.048  -0.048
<= seismic      0.048  -0.048
<= slope        0.048  -0.048
<= waterlogging 0.048  -0.048
<= soil bonding -0.048  0.048
<= site stability -0.048  0.048
<= subsistence  0.048  -0.048
<= soil strength -0.048  0.048
<= erosion      -0.048  0.048
<= agri potential 0.048  -0.048
<= comm potential -0.048  0.048
<= minerals     0.048  -0.048
<= cons. materials -0.048  0.048
<= topsoil avail -0.048  0.048
<= spoil management -0.048  0.048
<= unique features 0.048  -0.048
<= sand dunes    0.048  -0.048
<= prom features 0.048  -0.048
<= exist degradation -0.048  0.048
```

YESNO surface

```
TRUE (The project will be affected by the nature of the ground surface)
FALSE (The project will not be affected by the nature of the ground surface)
```

```
HELP (a)
```

```
QUESTION (Will the project be affected by the nature of the ground surface)
```

YESNO substrata

```
TRUE (The project will be affected by the nature of the rock substrata)
FALSE (The project will not be affected by the nature of the rock substrata)
```

```
HELP (a)
```

```
QUESTION (Will the project be affected by the nature of the rock substrata)
```

YESNO unstable rock

```
TRUE (There is unstable rock or faultlines on the site)
FALSE (There are no unstable rock or faultlines on the site)
```

```
HELP (a)
```

```
QUESTION (Is there unstable rock or faultlines on the site)
```

YESNO seismic

```
TRUE (Seismic activity is a problem)
FALSE (Seismic activity is not a problem)
```

```
HELP (a)
```

```
QUESTION (Is seismic activity a problem)
```

YESNO slope

```
TRUE (The slope of the land is too steep)
FALSE (The slope of the land is not too steep)
```

```
HELP (a)
```

```
QUESTION (Is the slope of the land too steep)
```

YESNO waterlogging

```
TRUE (There is a problem with waterlogging or depressions on the site)
FALSE (There is no problem with waterlogging or depressions on the site)
```

```
HELP (a)
```

```
QUESTION (Is there a problem with waterlogging or depressions on the site)
```

MENU soil bonding

```
TRUE (The soils on the site are bonded or binded in some way)
FALSE (The soils on the site are not bonded or binded in any way)
```

```
HELP (a)
```

```
QUESTION (Are the soils on the site bonded or binded in any way )
```

```
OPTIONS
```

```
1 (Bonded or binded eg clay)
```

```
0 (Mixture)
```

```
-1 (Unbonded soils eg. sand)
```

YESNO site stability

```
TRUE (The site is stable)
FALSE (The site is not stable)
```

```
HELP (a)
```

```
QUESTION (Is the site stable)
```

YESNO subsistence

```
TRUE (There is evidence of surface subsistence on the site)
FALSE (There is no evidence of surface subsistence on the site)
```

```
HELP (a)
```

```
QUESTION (Is there evidence of surface subsistence on the site)
```

YESNO soil strength

```
TRUE (The compressive strength of the soils is adequate )
FALSE (The compressive strength of the soils is not adequate )
```

```
HELP (a)
```

```
QUESTION (Is the compressive strength of the soils adequate )
```

YESNO erosion

```
TRUE (Erosion is not a problem)
FALSE (Erosion is a problem)
```

```
HELP (a)
```

```
QUESTION ( Will the site be protected against erosion )
```

YESNO agri potential

TRUE (The soils on site have a good potential for use by agriculture)
 FALSE (The soils on site do not have a good potential for use by agriculture)
 HELP (a)
 QUESTION (Does the soils on site have a good potential for use by agriculture)

YESNO comm potential
 TRUE (The soils on site have a good potential for use by commerce)
 FALSE (The soils on site do not have a good potential for use by commerce)
 HELP (a)
 QUESTION (Does the soils on site have a good potential for use by commerce)

YESNO minerals
 TRUE (There are mineral deposits on the site)
 FALSE (There are no mineral deposits on the site)
 HELP (a)
 QUESTION (Are there mineral deposits on the site)

YESNO cons. materials
 TRUE (There is access or availability to construction materials)
 FALSE (There is no access or availability to construction materials)
 HELP (a)
 QUESTION (Is there access or availability to construction materials)

YESNO topsoil avail
 TRUE (There is availability of topsoil or fill material)
 FALSE (There is no availability of topsoil or fill material)
 HELP (a)
 QUESTION (Is there availability of topsoil or fill material)

YESNO spoil management
 TRUE (Spoil material will be managed properly)
 FALSE (Spoil material will not be managed properly)
 HELP (a)
 QUESTION (Will spoil material be managed properly)

YESNO unique features
 TRUE (There are unique physical or geological features on the site)
 FALSE (There are no unique physical or geological features on the site)
 HELP (a)
 QUESTION (Are there unique physical or geological features on the site)

YESNO sand dunes
 TRUE (There are mobile sanddunes on the site)
 FALSE (There are no mobile sanddunes on the site)
 HELP (a)
 QUESTION (Are there mobile sanddunes on the site)

YESNO prom features
 TRUE (There are prominent landscape features on the site)
 FALSE (There are no prominent landscape features on the site)
 HELP (a)
 QUESTION (Are there any prominent landscape features on the site)

YESNO exist degradation
 TRUE (There is existing physical degradation of the local environment)
 FALSE (There is no existing physical degradation of the local environment)
 HELP (a)
 QUESTION (Is there existing physical degradation of the local environment)

/*****

***** Climate *****/
 /*****

SUBGOAL climate
 TRUE (The climate of the area favours development of the site)
 FALSE (The climate of the area does not favour development of the site)
 HELP (a)
 <= wind_strength 0.143 -0.143
 <= flash_floods 0.143 -0.143
 <= rainfall_patterns 0.143 -0.143
 <= temperature_fluks 0.143 -0.143
 <= inversions 0.143 -0.143
 <= pollutants 0.143 -0.143
 <= global_warming 0.143 -0.143

YESNO wind_strength
 TRUE (The site will be adversely affected by the wind strength)
 FALSE (The site will not be adversely affected by the wind strength)
 HELP (a)
 QUESTION (Will the development be affected by the wind strength)

YESNO flash_floods
 TRUE (The site will be affected by flash floods)
 FALSE (The site will not be affected by flash floods)
 HELP (a)
 QUESTION (Will the site be affected by flash floods)

YESNO rainfall_patterns
 TRUE (The site will adversely affect the rainfall patterns)
 FALSE (The site will not adversely affect the rainfall patterns)
 HELP (a)
 QUESTION (Will the development adversely affect the rainfall patterns)

YESNO temperature_fluks
 TRUE (The project will be affected by temperature flucuations)
 FALSE (The project will not be affected by temperature flucuations)
 HELP (a)
 QUESTION (Will the project be affected by temperature flucuations)

YESNO inversions
 TRUE (The project will be affected by the intensity of inversions)
 FALSE (The project will not be affected by the intensity of inversions)
 HELP (a)
 QUESTION (Will the project be affected by the intensity of inversions)

YESNO pollutants
 TRUE (The project will cause an influx of pollutants)
 FALSE (The project will not cause an influx of pollutants)
 HELP (a)
 QUESTION (Will the project cause an influx of pollutants)

YESNO global_warming
 TRUE (The site will enhance global warming or a rise in sea level)
 FALSE (The site will not enhance global warming or a rise in sea level)
 HELP (a)
 QUESTION (Will the site enhance global warming or a rise in sea level)

/*
Noise Pollution

This expert system rulebase assesses the impact of the proposed project on noise pollution in the area.

*/

DESCRIPTION (Noise Pollution)

GOAL Assessment

TRUE (The development will produce very little noise pollution effects)

FALSE (The development will produce significant noise pollution effects)

HELP ()

<= increase in ambient noise	PRIORITY 100	0.091	-0.091
<= time period of noise	PRIORITY 90	0.091	-0.091
<= creeping noise levels	PRIORITY 70	0.091	-0.091
<= quiet in residential areas	PRIORITY 60	-0.091	0.091
<= artificial lighting	PRIORITY 50	0.091	-0.091
<= functioning of community	PRIORITY 40	0.091	-0.091
<= individual protection	PRIORITY 30	-0.091	0.091
<= levels of annoyance	PRIORITY 25	0.091	-0.091
<= structural damage	PRIORITY 20	0.091	-0.091
<= effects on wildlife	PRIORITY 15	0.091	-0.091
<= reduction in wilderness quality	PRIORITY 10	0.091	-0.091

SCALE increase in ambient noise

TRUE (Ambient noise, vibration or illumination levels will be significantly effected)

FALSE (Ambient noise, vibration or illumination levels will not be significantly effected)

HELP (a)

QUESTION (To what extent will the development effect ambient noise, vibration, or illumination levels)

LABELS (Very Little) (Average) (Significant)

SCALE time period of noise

TRUE (The length of time that there will be noise, vibration or lighting impacts is many days)

FALSE (The length of time that there will be noise, vibration or lighting impacts is only a few minutes)

HELP (a)

QUESTION (What is the length of time that there will be noise, vibration or lighting impacts)

LABELS (Minutes) (Hours) (Days)

SCALE creeping noise levels

TRUE (The development will cause a significant exacerbation of 'creeping' ambient noise levels)

FALSE (The development will cause a no exacerbation of 'creeping' ambient noise levels)

HELP (a)

QUESTION (Will the development cause a exacerbation of 'creeping' ambient noise levels)

LABELS (None) (Average) (Significant)

YESNO quiet in residential areas

TRUE (There will be a period of peace and quiet in residential areas at some time)

FALSE (There will be no period of peace and quiet in residential areas at some time)

HELP (a)

QUESTION (Will there be peace and quiet in residential areas either during the day or at night time)

SCALE artificial lighting

TRUE (The quality of life will be affected due to artificial lighting)

FALSE (The quality of life will not be affected due to artificial lighting)

HELP (a)

QUESTION (To what extent will the quality of life be affected due to artificial lighting)

LABELS (None) (Average) (Significant)

SCALE functioning of community

TRUE (There will be significant effect on the functioning of community facilities)

FALSE (There will be very little effect on the functioning of community facilities)

HELP (a)

QUESTION (To what extent will the functioning of schools, hospitals and old peoples homes be effected by the noise and lighting)

LABELS (None) (Average) (Significant)

SCALE individual protection

TRUE (Individual protection against noise will be provided)

FALSE (Individual protection against noise will not be provided)

HELP (a)

QUESTION (Will individual protection against noise be provided)

SCALE levels of annoyance

TRUE (There will be significantly increased levels of annoyance and discomfort due to vibrations caused)

FALSE (There will be very little increased levels of annoyance and discomfort due to vibrations caused)

HELP (a)

QUESTION (To what degree will there be increased levels of annoyance and discomfort

due to vibrations caused by activities such as blasting and pile-driving)

LABELS (None) (Average) (Significant)

SCALE structural damage

TRUE (There will be significant structural damage caused to buildings)

FALSE (There will be very little structural damage caused to buildings)

HELP (a)

QUESTION (To what extent will there be structural damage caused to buildings due to vibrations)

LABELS (None) (Average) (Significant)

SCALE effects on wildlife

TRUE (There will be a significant impact on wildlife or sites of importance)

FALSE (There will be very little impact on wildlife or sites of importance)

HELP (a)

QUESTION (Will the noise, vibrations and lightings have a significant impact on wildlife or sites of special scientific interest)

LABELS (None) (Average) (Significant)

SCALE reduction in wilderness quality

TRUE (There will be a significant reduction in wilderness quality)

FALSE (There will be very little reduction in wilderness quality)

HELP (a)

QUESTION {Will the noise, vibrations and lightings cause a reduction of wilderness
S quality in declared wilderness areas}
LABELS {None} {Average} {Significant}

/*
Risk and Hazard

This expert system rulebase is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any risk or hazard that would result from the development of the project in question.
*/

DESCRIPTION { Risk and Hazard }

GOAL Assessment

TRUE {The risk and hazards associated with this site are sufficiently low}

FALSE {The risk and hazards associated with this site are high}

HELP { }

<= identity of hazard	PRIORITY 100	0.200	-0.200
<= probability of occurrence	PRIORITY 90	0.200	-0.200
<= extent of effect	PRIORITY 80	0.200	-0.200
<= workers safety	PRIORITY 70	0.200	-0.200
<= risk for other orgs	PRIORITY 60	0.200	-0.200

SCALE identity of hazard

TRUE {The level and identity of hazard to the public rate is high}

FALSE {The level and identity of hazard to the public rate is low}

HELP {a}

QUESTION {How does the level and identity of hazard to the public rate}

LABELS {Low} {Average} {High}

SCALE probability of occurrence

TRUE {The probability of occurrence is high}

FALSE {The probability of occurrence is low}

HELP {a}

QUESTION {How does the probability of occurrence rate}

LABELS {Low} {Average} {High}

SCALE extent of effect

TRUE {The extent of the effect is panoramic}

FALSE {The extent of the effect is local}

HELP {a}

QUESTION {What is the extent of effect of the risk}

LABELS {Local} {Regional} {Panoramic}

SCALE workers safety

TRUE {The workers degree of risk is high}

FALSE {The workers degree of risk is low}

HELP {a}

QUESTION {How would you rate the workers degree of risk}

LABELS {Low} {Average} {High}

SCALE risk for other orgs

TRUE {The level of risk and hazard for other living organisms is high}

FALSE {The level of risk and hazard for other living organisms is low}

HELP {a}

QUESTION {What is the level of risk and hazard for other living organisms}

LABELS {Low} {Average} {High}

/*
Social and Community Services and Facilities

This Expert System Rule Base is specifically designed to investigate whether the proposed development could have a significant impact on, or be constrained by any of the following :

- 1) Health Service Facilities
 - 2) Emergency Services
 - 2) Recreational Facilities
- */

DESCRIPTION (Social and Community Services and Facilities)

GOAL Assessment

TRUE (The current and potential landuse character is favourable)
FALSE (The current and potential landuse character is unfavourable)
HELP ()

<= Health	PRIORITY 100	-0.333	0.333
<= Emergency	PRIORITY 90	-0.333	0.333
<= Recreational	PRIORITY 80	-0.333	0.333

***** Health *****

SUBGOAL Health

TRUE (Health service facilities will not be required)
FALSE (Health service facilities will be required)
HELP (a)

<= temporary facilities	-0.20	0.20
<= adequacy on-site health	-0.20	0.20
<= adequacy primary health	-0.20	0.20
<= adequacy existing health	-0.20	0.20
<= need for additional	0.20	-0.20

YESNO temporary facilities

TRUE (Temporary facilities for health care will be adequate during the construction phase)

FALSE (Temporary facilities for health care will not be adequate during the construction phase)

HELP (a)
QUESTION (Are temporary facilities for health care adequate during the construction phase)

YESNO adequacy on-site health

TRUE (On-site health facilities are adequate)

FALSE (On-site health facilities are not adequate)

HELP (a)
QUESTION (Are on-site health facilities adequate)

YESNO adequacy primary health

TRUE (Facilities for primary health care are adequate)

FALSE (Facilities for primary health care are not adequate)

HELP (a)
QUESTION (Are facilities for primary health care adequate)

SCALE adequacy existing health

TRUE (Existing health service facilities are adequate to copy with the increased population)

FALSE (Existing health service facilities are not adequate to copy with the increased population)

HELP (a)

QUESTION (To what extent are existing health service facilities adequate to copy with the increased population)

LABELS (Poor) (Adequate) (Good)

YESNO need for additional

TRUE (There is a need for additional facilities)

FALSE (There is no need for additional facilities)

HELP (a)

QUESTION (Is there a need for additional facilities)

***** Emergency *****

SUBGOAL Emergency

TRUE (Emergency service facilities will not be required)

FALSE (Emergency service facilities will be required)

HELP (a)

<= adequacy existing emerg	-0.25	0.25
<= need for addi emerg services	0.25	-0.25
<= adequacy services by develop	-0.25	0.25
<= ability of local resources	-0.25	0.25

SCALE adequacy existing emerg

TRUE (Existing emergency services are adequate)

FALSE (Existing emergency services are not adequate)

HELP (a)

QUESTION (To what extent are existing emergency services adequate)

LABELS (Poor) (Average) (Good)

YESNO need for addi emerg services

TRUE (There is a need for additional emergency services)

FALSE (There is no need for additional emergency services)

HELP (a)

QUESTION (Is there a need for additional emergency services)

SCALE adequacy services by develop

TRUE (Emergency services provided by the developer are adequate)

FALSE (Emergency services provided by the developer are not adequate)

HELP (a)

QUESTION (To what extent are emergency services provided by the developer adequate)

LABELS (Poor) (Average) (Good)

SCALE ability of local resources

TRUE (Local resources do have the ability to deal with emergencies)

FALSE (Local resources do not have the ability to deal with emergencies)

HELP (a)

QUESTION (To what extent do local resources have the ability to deal with emergencies)

LABELS (Poor) (Average) (Good)

***** Recreational *****

SUBGOAL Recreational

TRUE (Recreational service facilities will not be required)
FALSE (Recreational service facilities will be required)
HELP (a)
<= adequacy existing recreatns -0.33 0.33
<= need for addi recrt facilities 0.33 -0.33
<= facilities in the work place -0.33 0.33

YESNO adequacy existing recreatns
TRUE (Existing recreational services are adequate)
FALSE (Existing recreational services are not adequate)
HELP (a)
QUESTION (Are existing recreational services adequate)

YESNO need for addi recrt facilities
TRUE (There is a need for additional facilities)
FALSE (There is no need for additional facilities)
HELP (a)
QUESTION (Is there a need for additional facilities)

YESNO facilities in the work place
TRUE (There will be recreational service facilities in the work place)
FALSE (There will not be recreational service facilities in the work place)
HELP (a)
QUESTION (Will there be recreational service facilities in the work place)

/*
Cumulative and Synergistic Effects

This expert system rulebase assesses the impact of the proposed project in relation to cumulative and synergistic effects.

*/

DESCRIPTION (Cumulative and Synergistic Effects)

GOAL Assessment

TRUE (The cumulative and synergistic effects of the site is favourable)

FALSE (The cumulative and synergistic effects of the site is unfavourable)

HELP ()

<= cumulative stresses PRIORITY 100 -0.333 0.333

<= neg synergistic effects PRIORITY 90 0.333 -0.333

<= precedent being set PRIORITY 80 0.333 -0.333

SCALE cumulative stresses

TRUE (The ability of the environments to assimilate stresses is high)

FALSE (The ability of the environments to assimilate stresses is low)

HELP (a)

QUESTION (How does the ability of the natural and social environments to assimilate cumulative stresses placed on them, rate)

LABELS (Low) (Average) (High)

SCALE neg synergistic effects

TRUE (The extent to which negative synergistic effects will arise is high)

FALSE (The extent to which negative synergistic effects will arise is low)

HELP (a)

QUESTION (To what extent will there arise the likelihood of negative synergistic effects)

LABELS (Low) (Average) (High)

SCALE precedent being set

TRUE (The extent to which the precedent will effect future dev. rights is high)

FALSE (The extent to which the precedent will effect future dev. rights is low)

HELP (a)

QUESTION (To what extent will the development effect existing or future development rights because of the precedent being set)

LABELS (Minimal) (Average) (Large)

/*

Transport Network Infrastructure Services

This expert system rulebase assesses the impact of the proposed project on transport network infrastructures.

*/

DESCRIPTION (Transport Network Infrastructure Services)

GOAL Assessment

TRUE (The development will produce very little effect on transport networks)
 FALSE (The development will produce significant effect on transport networks)
 HELP ()

<= existing systems	PRIORITY 100	-0.071	0.071
<= present patterns	PRIORITY 90	0.071	-0.071
<= generation of more traffic	PRIORITY 70	0.071	-0.071
<= adequacy of existing roads	PRIORITY 60	-0.071	0.071
<= adequacy of existing parking	PRIORITY 55	-0.071	0.071
<= adequacy of existing man	PRIORITY 50	-0.071	0.071
<= need for extra schemes	PRIORITY 45	0.071	-0.071
<= temporary access roads	PRIORITY 40	0.071	-0.071
<= rail service	PRIORITY 35	-0.071	0.071
<= rail capacity	PRIORITY 30	-0.071	0.071
<= additional rail links	PRIORITY 25	0.071	-0.071
<= harbour facilities	PRIORITY 20	-0.071	0.071
<= need for expanded harbour	PRIORITY 15	0.071	-0.071
<= adequacy of air transport	PRIORITY 10	-0.071	0.071

YESNO existing systems

TRUE (There are existing transportation systems in the area)
 FALSE (There are no existing transportation systems in the area)
 HELP (a)
 QUESTION (Are there existing transportation systems in the area)

SCALE present patterns

TRUE (The extent to which the project will effect present patters of circulation or movement of people and/or goods is very large)
 FALSE (The extent to which the project will effect present patters of circulation or movement of people and/or goods is very small)
 HELP (a)
 QUESTION (To what extent will the project effect present patters of circulation or movement of people and/or goods)
 LABELS (None) (Average) (Large)

SCALE generation of more traffic

TRUE (The project will generate more private and public traffic)
 FALSE (The project will not generate more private and public traffic)
 HELP (a)
 QUESTION (To what extent will the project generate more private and public traffi c)
 LABELS (None) (Average) (Large)

YESNO adequacy of existing roads

TRUE (Existing road networks are adequate)
 FALSE (Existing road networks are not adequate)
 HELP (a)
 QUESTION (Are existing road networks adequate)

YESNO adequacy of existing parking

TRUE (Existing parking facilities are adequate)
 FALSE (Existing parking facilities are not adequate)
 HELP (a)
 QUESTION (Are existing parking facilities adequate)

YESNO adequacy of existing man

TRUE (Existing traffic management schemes are adequate)
 FALSE (Existing traffic management schemes are not adequate)
 HELP (a)
 QUESTION (Are existing traffic management schemes adequate)

SCALE need for extra schemes

TRUE (There is a need for extra road schemes over and above those that have been planned)
 FALSE (There is no need for extra road schemes over and above those that have been planned)
 HELP (a)
 QUESTION (To what extent is there a need for extra road schemes over and above those that have been planned)
 LABELS (None) (Average) (Large)

YESNO temporary access roads

TRUE (Temporary access roads will be used for the development)
 FALSE (Temporary access roads will not be used for the development)
 HELP (a)
 QUESTION (Will temporary access roads be used for the development)

SCALE rail service

TRUE (The rail service is very viable)
 FALSE (The rail service is not very viable)
 HELP (a)
 QUESTION (To what extent is the rail service viable)
 LABELS (Not Viable) (Average) (Very Viable)

YESNO rail capacity

TRUE (The existing rail capacity is sufficient)
 FALSE (The existing rail capacity is not sufficient)
 HELP (a)
 QUESTION (Will the existing rail capacity be sufficient)

YESNO additional rail links

TRUE (The project will require additional rail links)
 FALSE (The project will not require additional rail links)
 HELP (a)
 QUESTION (The project require additional rail links)

SCALE harbour facilities

TRUE (Current harbour facilities are adequate)
 FALSE (Current harbour facilities are not adequate)
 HELP (a)
 QUESTION (Are current harbour facilities adequate for the project)
 LABELS (No) (Average) (Mostly)

YESNO need for expanded harbour

TRUE (The project will require expanded harbour and related facilities)
 FALSE (The project will not require expanded harbour and related facilities)
 HELP (a)
 QUESTION (Will the project require expanded harbour and related facilities)

YESNO adequacy of air transport

TRUE {Current air transport facilities are adequate for the project}
FALSE {Current air transport facilities are not adequate for the project}
HELP {a}
QUESTION {Are current air transport facilities adequate for the project}

/*

Waste Management Infrastructure Services

This expert system rulebase assesses the impact of the proposed project on waste management infrastructure services.

*/

DESCRIPTION (Waste Management Infrastructure Services)

GOAL Assessment

TRUE (The development will produce very little effect on current waste facilities)

FALSE (The development will produce significant effect on current waste facilities)

HELP ()

<= existing facilities	PRIORITY 100	-0.071	0.071
<= contribution to facilities	PRIORITY 90	-0.071	0.071
<= provision of facilities	PRIORITY 70	-0.071	0.071
<= need for new pipelines	PRIORITY 60	0.071	-0.071
<= transportation risk	PRIORITY 50	0.071	-0.071
<= emergency waste disposal	PRIORITY 45	-0.071	0.071
<= risk to community	PRIORITY 40	0.071	-0.071
<= groundwater pollution	PRIORITY 35	0.071	-0.071
<= scavengers at waste sites	PRIORITY 30	0.071	-0.071
<= windblown pollution	PRIORITY 25	0.071	-0.071
<= visual and smell effects	PRIORITY 20	0.071	-0.071
<= bird hazard	PRIORITY 15	0.071	-0.071
<= recycling	PRIORITY 10	-0.071	0.071
<= onsite waste man	PRIORITY 5	-0.071	0.071

YESNO existing facilities

TRUE (The efficiency and capacity of existing waste management facilities are adequate for the new project)

FALSE (The efficiency and capacity of existing waste management facilities are not adequate for the new project)

HELP (a)

QUESTION (Are the efficiency and capacity of existing waste management facilities adequate for the new project)

SCALE contribution to facilities

TRUE (The project will contribute greatly to centralised waste processing facilities)

FALSE (The project will not contribute to centralised waste processing facilities)

HELP (a)

QUESTION (To what extent will the project contribute to centralised waste processing facilities)

LABELS (None) (Average) (Large)

YESNO provision of facilities

TRUE (The project will provide necessary facilities)

FALSE (The project will not provide necessary facilities)

HELP (a)

QUESTION (Will the project provide necessary facilities)

YESNO need for new pipelines

TRUE (There will be a need for new pipelines)

FALSE (There will not be a need for new pipelines)

HELP (a)

QUESTION (Will there be a need for new pipelines)

SCALE transportation risk

TRUE (The risk associated with transporting the waste is very large)

FALSE (The risk associated with transporting the waste is very small)

HELP (a)

QUESTION (What environmental risk will be associated with transporting the waste)

LABELS (None) (Average) (Extreme)

YESNO emergency waste disposal

TRUE (The project will provide adequate emergency waste disposal facilities)

FALSE (The project will not provide adequate emergency waste disposal facilities)

HELP (a)

QUESTION (Will the project provide adequate emergency waste disposal facilities)

SCALE risk to community

TRUE (The risk to the community and environment should the facility break down is very large)

FALSE (The risk to the community and environment should the facility break down is very small)

HELP (a)

QUESTION (What risk is there to the community and environment should the facility

break down)

LABELS (None) (Average) (Extreme)

YESNO groundwater pollution

TRUE (There is a possibility that the groundwater will be polluted)

FALSE (There is no possibility that the groundwater will be polluted)

HELP (a)

QUESTION (Is there a possibility that the groundwater will be polluted)

YESNO scavengers at waste sites

TRUE (There is a danger of rodents and scavengers at waste sites)

FALSE (There is no a danger of rodents and scavengers at waste sites)

HELP (a)

QUESTION (Is there a danger of rodents and scavengers at waste sites)

SCALE windblown pollution

TRUE (The potential for windblown or waterblown pollution is very great)

FALSE (The potential for windblown or waterblown pollution is very little)

HELP (a)

QUESTION (What is the potential for windblown or waterblown pollution)

LABELS (None) (Average) (Large)

SCALE visual and smell effects

TRUE (There will be visual and smell effects of waste sites and treatment works)

FALSE (There will not be visual and smell effects of waste sites and treatment works)

HELP (a)

QUESTION (To what extent will there be visual and smell effects of waste sites and treatment works)

LABELS (None) (Average) (Extensive)

YESNO bird hazard

TRUE (There is a hazard to birds to air traffic near sewage ponds and landfill sites)

FALSE (There is no hazard to birds to air traffic near sewage ponds)

and landfill sites}

HELP {a}

QUESTION {Is there a hazard to birds to air traffic near sewage ponds
and landfill sites}

YESNO recycling

TRUE {There will be utilization of treated waste water and recycled
materials}

FALSE {There will be no utilization of treated waste water and recycled
materials}

HELP {a}

QUESTION {Will there be any utilization of treated waste water and recycled
materials}

SCALE onsite waste man

TRUE {There is on-site waste management potential}

FALSE {There is no on-site waste management potential}

HELP {a}

QUESTION {Is there on-site waste management potential}

/*
Water Infrastructure Services

This expert system rulebase assesses the impact of the proposed project on water supply infrastructure services.
*/

DESCRIPTION { Water Infrastructure Services }

GOAL Assessment

TRUE {The development will produce very little effect on the current water supply}
FALSE {The development will produce significant effect on the current water supply}

HELP { }

<= water rights	PRIORITY 100	-0.091	0.091
<= excessive requirements	PRIORITY 90	0.091	-0.091
<= planned provision	PRIORITY 70	-0.091	0.091
<= reliability of supply	PRIORITY 60	-0.091	0.091
<= groundwater reserves	PRIORITY 50	-0.091	0.091
<= emergency supply	PRIORITY 40	-0.091	0.091
<= new schemes	PRIORITY 25	0.091	-0.091
<= additional purification	PRIORITY 20	0.091	-0.091
<= impoundments	PRIORITY 15	0.091	-0.091
<= new pipelines	PRIORITY 10	0.091	-0.091
<= danger to community	PRIORITY 5	0.091	-0.091

YESNO water rights

TRUE {The site has water rights}
FALSE {The site has no water rights}
HELP {a}
QUESTION {Does the site have water rights}

YESNO excessive requirements

TRUE {The project will have wasteful or excessive water requirements}
FALSE {The project will not have wasteful or excessive water requirements}
HELP {a}
QUESTION {Will the project have wasteful or excessive water requirements}

YESNO planned provision

TRUE {There is planned provision for water supply to the area}
FALSE {There is no planned provision for water supply to the area}
HELP {a}
QUESTION {Is there a planned provision for water supply to the area}

SCALE reliability of supply

TRUE {The adequacy and reliability of the water supply rate is very good}
FALSE {The adequacy and reliability of the water supply rate is poor}
HELP {a}
QUESTION {How would the adequacy and reliability of the water supply rate}
LABELS {Poor} {Average} {Excellent}

SCALE groundwater reserves

TRUE {The groundwater reserves are very adequate}
FALSE {The groundwater reserves are not very inadequate}
HELP {a}
QUESTION {How adequate are the groundwater reserves}
LABELS {Poor} {Average} {Excellent}

SCALE emergency supply

TRUE {The emergency supply system is very adequate}
FALSE {The emergency supply system is not very inadequate}
HELP {a}
QUESTION {How adequate is the emergency supply system}
LABELS {Poor} {Average} {Excellent}

YESNO new schemes

TRUE {The project will require the construction of new supply reservoirs}
FALSE {The project will not require the construction of new supply reservoirs}
HELP {a}
QUESTION {Will the project require the construction of new supply reservoirs}

YESNO additional purification

TRUE {The project will require additional purification systems}
FALSE {The project will not require additional purification systems}
HELP {a}
QUESTION {Will the project require additional purification systems}

YESNO impoundments

TRUE {The project will need inappropriately sized or located impoundments}
FALSE {The project will not need inappropriately sized or located impoundments}
HELP {a}
QUESTION {Will the project need inappropriately sized or located impoundments}

YESNO new pipelines

TRUE {The project will require the construction of new pipelines}
FALSE {The project will not require the construction of new pipelines}
HELP {a}
QUESTION {Will the project require the construction of new pipelines}

SCALE danger to community

TRUE {The local community or industry will be badly affected by a major water supply failure}
FALSE {The local community or industry will not be affected by a major water supply failure}
HELP {a}
QUESTION {To what extent will the local community or industry be affected by a major water supply failure}
LABELS {None} {Average} {Badly}

/*
Water Pollution

This expert system rulebase assesses the impact of the proposed project on water pollution in the area.

*/

DESCRIPTION (Water Pollution)

GOAL Assessment

TRUE (The effect of the development on water pollution is minimal)

FALSE (The effect of the development on water pollution is great)

HELP ()

<= existing pollution	PRIORITY	100	0.077	-0.077
<= localised pollution levels	PRIORITY	90	0.077	-0.077
<= pollution of surface waters	PRIORITY	80	0.077	-0.077
<= concentration of pollutants	PRIORITY	70	0.077	-0.077
<= changes in salinity gradients	PRIORITY	60	0.077	-0.077
<= dispersal mechanisms	PRIORITY	50	-0.077	0.077
<= salinization of fresh waters	PRIORITY	40	0.077	-0.077
<= compounding effects	PRIORITY	30	0.077	-0.077
<= offensive odours	PRIORITY	25	0.077	-0.077
<= effluent effect	PRIORITY	20	0.077	-0.077
<= dependent natural communities	PRIORITY	15	0.077	-0.077
<= irrigation schemes	PRIORITY	10	0.077	-0.077
<= recreational activities	PRIORITY	5	0.077	-0.077

SCALE existing pollution

TRUE (Existing water in the area is very polluted)

FALSE (Existing water in the area is not polluted)

HELP (a)

QUESTION (To what extent is existing water in the area polluted)

LABELS (None) (Average) (High)

YESNO localised pollution levels

TRUE (There are high localised levels of pollution)

FALSE (There are no high localised levels of pollution)

HELP (a)

QUESTION (Are there high localised levels of pollution)

SCALE pollution of surface waters

TRUE (Surface waters are polluted from underground waters)

FALSE (Surface waters are not polluted from underground waters)

HELP (a)

QUESTION (To what extent is surface waters polluted from underground waters)

LABELS (Little) (Average) (Significant)

SCALE concentration of pollutants

TRUE (Concentrations of pollutants will significantly be affected by variations of

water flow)

FALSE (Concentrations of pollutants will be affected very little by variations of

water flow)

HELP (a)

QUESTION (Will the concentrations of pollutants be affected by variations of

water flow)

LABELS (Very Little) (Average) (Significant)

SCALE changes in salinity gradients

TRUE (The chances of localised pollution build up are high)

FALSE (The chances of localised pollution build up are low)

HELP (a)

QUESTION (What are the chances of localised pollution build-up through changes in salinity gradients and/or current movements)

LABELS (Low) (Average) (High)

YESNO dispersal mechanisms

TRUE (Effective dispersal mechanisms are present in the waters)

FALSE (Effective dispersal mechanisms are not present in the waters)

HELP (a)

QUESTION (Are effective dispersal mechanisms present in the waters)

YESNO salinization of fresh waters

TRUE (The development will effect the salinization of fresh waters)

FALSE (The development will not effect the salinization of fresh waters)

HELP (a)

QUESTION (Will the development effect the salinization of fresh waters)

SCALE compounding effects

TRUE (There will be a significant synergistic or compound effect with existing pollutants)

FALSE (There will be very little synergistic or compound effects with existing pollutants)

HELP (a)

QUESTION (To what extent will the developments effluent have a synergistic or compounding effect with existing pollutants)

LABELS (Very Little) (Average) (Significant)

YESNO offensive odours

TRUE (The effluent will produce offensive odours)

FALSE (The effluent will not produce offensive odours)

HELP (a)

QUESTION (Will the effluent produce offensive odours)

SCALE effluent effect

TRUE (The effluent will significantly effect the fauna and flora of the waterway)

FALSE (The effluent will not effect the fauna and flora of the waterway)

HELP (a)

QUESTION (To what extent will the effluent effect the flora and fauna of river, lake, canal, estuary or costal waters)

LABELS (Very Little) (Average) (Significant)

SCALE dependent natural communities

TRUE (The effluent will significantly effect dependent natural communities)

FALSE (The effluent will not effect dependent natural communities)

HELP (a)

QUESTION (To what extent will the effluent effect dependent natural communities through changes in aquatic fauna and flora)

LABELS (Very Little) (Average) (Significantly)

SCALE irrigation schemes

TRUE (The effluent will significantly effect neighbouring irrigation schemes)

FALSE (The effluent will not effect neighbouring irrigation schemes)

HELP (a)

QUESTION (To what extent will the effluent effect neighbouring irrigation schemes)

LABELS (Very Little) (Average) (Significant)

SCALE recreational activities

TRUE {The effluent will significantly effect recreational activities}

FALSE {The effluent will not effect recreational activities}

HELP {a}

QUESTION {To what extent will the effluent effect recreational activities}

LABELS {Very Little} {Average} {Significant}

/*
Air Pollution

This expert system rulebase assesses the impact of the proposed project on air pollution in the area.

*/

DESCRIPTION (Air Pollution Assessment)

GOAL Assessment

TRUE (The development will produce very little air pollution effects)

FALSE (The development will produce significant air pollution effects)

HELP ()

<= existing pollution	PRIORITY 100	0.100 -0.100
<= nature of pollution	PRIORITY 90	0.100 -0.100
<= extent of local buildup	PRIORITY 80	0.100 -0.100
<= compounding effects	PRIORITY 70	0.100 -0.100
<= smog formation	PRIORITY 60	0.100 -0.100
<= quantity and type matter	PRIORITY 50	0.100 -0.100
<= offensive odours	PRIORITY 40	0.100 -0.100
<= adjacent sensitive areas	PRIORITY 30	0.100 -0.100
<= effects on health crops etc	PRIORITY 20	0.100 -0.100
<= effects on stonework	PRIORITY 10	0.100 -0.100

SCALE existing pollution

TRUE (The existing levels of atmospheric pollution is high)

FALSE (The existing levels of atmospheric pollution is low)

HELP (a)

QUESTION (How would you rate the existing levels of atmospheric pollution)

LABELS (Low) (Average) (High)

SCALE nature of pollution

TRUE (The nature of the air pollution is highly detrimental)

FALSE (The nature of the air pollution is harmless)

HELP (a)

QUESTION (How would the you rate the nature of the air pollution ie ozone-depleting

gasses,acidic compounds being detrimental, and odours fairly harmless)

LABELS (Harmless) (Mildly Toxic) (Highly Detremental)

SCALE extent of local buildup

TRUE (The extent of the local build-up of pollutants due to inversions is frequent)

FALSE (The extent of the local build-up of pollutants due to inversions is low)

HELP (a)

QUESTION (What is the extent of the local build-up of pollutants due to inversions)

LABELS (None) (Average) (Frequently)

YESNO compounding effects

TRUE (There are compounding effects with existing pollutants or other chemicals)

FALSE (There are no compounding effects with existing pollutants or other chemicals)

HELP (a)

QUESTION (Are there compounding effects with existing pollutants or other chemicals

in the atmosphere (eg photo chemical smog))

SCALE smog formation

TRUE (The is a high probability that the pollution will enhance smog formation)

FALSE (The is a low probability that the pollution will enhance smog formation)

HELP (a)

QUESTION (To what extent will the pollution enhance smog formation and reduction in visibility)

LABELS (Very Little) (Average) (High Probability)

SCALE quantity and type matter

TRUE (The particulate matter is in relatively large quantities and is unstable)

FALSE (The particulate matter is in relatively small quantities and is stable)

HELP (a)

QUESTION (How would you rate the quantity and type of particulate matter produced

with reference to chemical stability, partical size etc)

LABELS (Small Quantities, Stable) (Large Quantities, Stable) (Large Quantities, Unstable)

YESNO offensive odours

TRUE (There will be a production of offensive odours)

FALSE (There will not be a production of offensive odours)

HELP (a)

QUESTION (Will there be a production of offensive odours)

MENU adjacent sensitive areas

TRUE (The pollution will effect adjacent sensitive areas)

FALSE (The pollution will not effect adjacent sensitive areas)

HELP (a)

QUESTION (Will the pollution effect adjacent sensitive areas)

OPTIONS

1 (Very Severly)

0 (Mildly)

-1 (Not at All)

SCALE effects on health crops etc

TRUE (Human health, crops, wildlife be affected to a large extent)

FALSE (Human health, crops, wildlife be affected very little)

HELP (a)

QUESTION (To what extent will human health, crops, wildlife be affected)

LABELS (Low) (Average) (High)

SCALE effects on stonework

TRUE (Stonework, buildings or works of art be affected to a large extent)

FALSE (Stonework, buildings or works of art be affected very little)

HELP (a)

QUESTION (To what extent will stonework, buildings or works of art be affected)

LABELS (Low) (Average) (High)

FILE LISTS

<u>File Name</u>	<u>File Type</u>
BachComp.pas	Program
BachList.pas	Program
CmndList.pas	Unit
CntrlUnt.pas	Unit
Editors.pas	Program
Esea.pas	Program
ExptMan.pas	Unit
General1.pas	Unit
General2.pas	Unit
Help.pas	Unit
HelpFile.pas	Unit
Install.pas	Program
ListADlg.pas	Unit
ListDDlg.pas	Unit
ListFDlg.pas	Unit
ListHDlg.pas	Unit
PkList1	Unit
StrmComp.pas	Program
StrmOut.pas	Program
StrmUnit.pas	Unit
Thesis2.pas	Program

<u>File Name</u>	<u>File Type</u>
ViewWin2.pas	Unit
ViewWinF.pas	Unit

```

{*****}
{
{
UNIT : AppMan
{
{
{*****}
{$X+}
Unit AppMan;

interface

uses { Standard Units }
    Crt,Dialogs,Drivers,Views,Objects,MsgBox,Menus,Memory,Dos,
    { Other }
    ViewWinF,
    { Farm Units }
    CmdndFarm,FdbList,OrchUnit,General2,ListADlg;

type
    string40 = string[40];

    TDinfoRec1 = record
        Instruction : string[50];
        Date : string[8];
        Problem : string[10];
        Method : string[20];
        TypeApp : string[12];
        Chemical : string[18];
        Concentration : string[10];
        QuantityPerTree : string[10];
        QuantityPerOrch : string[10];
        UnitCost : string[10];
    end;

    TDinfoRec3 = record
        Name : string[10];
    end;

{ DChooseData record for use in ViewStream Method selecting
  type of spray applications to use }

DChooseData = record
    CheckBoxData1: Word;
    CheckBoxData2: Word;
    InputLineData: string[28];
end;

PDBaseDialog = ^TDBaseDialog;
TDBaseDialog = object(TDialog)
    constructor Init;
    destructor Done;virtual;
    procedure HandleEvent(var Event:TEvent);virtual;
    procedure ViewStream;
    procedure AddApplications;
    procedure EditApplications;
    procedure DeleteApplications;
    procedure NoBlanks(var S:String40);
    procedure FillBlanks(var S:String; StrLength:Integer);

```

```

    procedure LoadStream;
    end;

    PInfoWindow = ^TInfoWindow;
    TInfoWindow = object(TWindow)
        constructor Init(Bounds: TRect; WinTitle: String; WindowNo: Word);
        procedure HandleEvent(var Event: TEvent); virtual;
    end;

    PInterior1 = ^TInterior1;
    TInterior1 = object(TView)
        constructor Init(var Bounds: TRect);
        procedure Draw; virtual;
    end;

    PChooseDialog = ^TChooseDialog;
    TChooseDialog = object(TDialog)
    end;

var
    LineCount,yy : Integer;
    Actual,AllControl: Word;
    TDinfo1 : TDinfoRec1;
    TDinfo3 : TDinfoRec3;
    Finished,DupeRec : Boolean;
    Collection : PDataCollection;
    DataPointer : Pointer;
    AppStreamName : String;

    { Checked }
    TransWCode : TransferWithCode; {declared in ListADlg}
    OrchardList : PCollection;
    OrchardAppList : PCollection;
    OrchardAppStream : TBufStream;
    DialogChooseData : DChooseData;
    Format1,Format2,Format3,Format4 : Boolean;

implementation

uses App;

{*****}
{ ***** TInterior1 ***** }
{*****}

constructor TInterior1.Init(var Bounds: TRect);
begin
    TView.Init(Bounds);
    GrowMode := gfGrowHiX + gfGrowHiY;
    Options := Options or ofFramed;
end;

procedure TInterior1.Draw;
var
    s1,s2,s3,s4 : string;
begin
    TView.Draw;
    Str(LineCount,s1);
    Str(yy,s4);
    s2:= ' Line Number = '+s1;

```

```

WriteStr(4, 2, s2, $02);
if DupeRec or (yy >= 1) then
begin
  s3:= ' Duplicate Records Found = '+s4;
  WriteStr(4, 4, s3, $02);
  DupeRec:=false;
end;
end;

{*****}
{***** TInfoWindow *****}
{*****}

constructor TInfoWindow.Init(Bounds: TRect; WinTitle: String; WindowNo: Word);
var
  S      : string[3];
  Interior1 : PInterior1;
begin
  Str(WindowNo, S);
  TWindow.Init(Bounds, WinTitle + ' ' + S, wnNoNumber);
  GetClipRect(Bounds);
  Bounds.Grow(-1, -1);
  Palette:=1;
  Interior1 := New(PInterior1, Init(Bounds));
  Insert(Interior1);
end;

procedure TInfoWindow.HandleEvent(var Event: TEvent);
begin
  ( Respond to Automatic Window Cancel )
  with Event do
    if (What=evBroadcast) AND (Command=cmCloseInfoWindow) then
      begin
        ClearEvent(Event);
        what := evCommand;
        Command := cmClose;
        PutEvent(Event);
      end;

  TWindow.HandleEvent(Event);
  ClearEvent(Event);
end;

{*****}
{***** TDBaseDialog *****}
{*****}

{*****}
{***** LoadStream *****}
{*****}

procedure TDBaseDialog.LoadStream;
begin
  with OrchardAppStream do
    begin
      Init(AppStreamName, stOpenRead, 1024); { Open stream }
      OrchardAppList := PCollection(Get);    { Load collection }
      Done;                                  { Shut down stream }
      if Status <> 0 then                    { Check for error }

```

```

begin
  MessageBox(#3'Error loading : '+AppStreamName, nil, mfInformation + mfOkBut
ton);
end;
end;
end;

{*****}
{***** Init *****}
{*****}

constructor TDBaseDialog.Init;
var
  R : TRect;
  C : Word;
  N : string[30];
  i : integer;
  Q : PView;
  LoadName : String;
  Event : TEvent;
begin
  finished:=false;
  N:='Orchard Spray Applications';
  R.Assign(1, 1, 65, 18);
  TDialog.Init(R, N);
  R.Move(7, 0);
  TDialog.Init(R, N);

  { Input lines }
  R.Assign(20, 2, 33, 3);
  Insert(New(PInputLine, Init(R, 50)));

  { Buttons }
  { Column 1 }
  R.Assign(5, 6, 19, 8);
  Insert(New(PButton, Init(R, ' ~L~ist ', cmViewAll, bfNormal)));
  R.Assign(5, 8, 19, 10);
  Insert(New(PButton, Init(R, ' ~A~dd ', cmAddApps, bfNormal)));
  R.Assign(5, 10, 19, 12);
  Insert(New(PButton, Init(R, ' ~E~dit ', cmEditApps, bfNormal)));
  R.Assign(5, 12, 19, 14);
  Insert(New(PButton, Init(R, ' ~D~elete ', cmDelApps, bfNormal)));

  { Column 2 }
  R.Assign(25, 14, 39, 16);
  Insert(New(PButton, Init(R, ' ~O~k ', cmOK, bfDefault)));

  { Labels }
  { Column 1 }
  R.Assign(5, 2, 19, 3);
  Insert(New(PLabel, Init(R, 'Orchard Name', Q)));
  R.Assign(5, 4, 19, 5);
  Insert(New(PLabel, Init(R, 'Select Action', Q)));
  R.Assign(20, 6, 61, 7);
  Insert(New(PLabel, Init(R, 'History of Applications for this Orchard', Q)));
  R.Assign(20, 8, 61, 9);
  Insert(New(PLabel, Init(R, 'a New Spray Application', Q)));
  R.Assign(20, 10, 61, 11);
  Insert(New(PLabel, Init(R, 'an Existing Spray Application', Q)));
  R.Assign(20, 12, 61, 13);

```

```

Insert(New(PLabel, Init(R, 'an Existing Spray Application', Q)));
( Load Stream )
( OrchardAppStream, OrchardAppList declared in OrchUnit.pas )
AppStreamName := TInfo3.Name+'.stm';
System.Delete(AppStreamName,5,3);
LoadStream;
end;

(*****
(*                               *)
(*                               *)
(*****

destructor TDBaseDialog.done;
begin
  ( Dont close stream if it didn't load )
  if OrchardAppStream.Status=0 then Dispose(OrchardAppList, Done);( Delete collec
tion )
  TDialog.Done;
end;

(*****
(*                               *)
(*                               *)
(*****

procedure TDBaseDialog.HandleEvent(var Event: TEvent);
var
  i      : integer;
begin
  ( Respond to CANCEL button and ESC )
  if ((Event.What = evKeyDown) and (Event.KeyCode = kbEsc)) or
    ((Event.What = evCommand) and (Event.Command = cmOK)) then
  begin
    ClearEvent(Event);
    Finished := true;
    Free;
    Exit;
  end;

  if (Event.What = evCommand) and (Event.Command = cmShutDown10) then
  begin
    ClearEvent(Event);
    Finished := true;
    Free;
    Exit;
  end;

  ( Respond to Down Arrow Key )
  if ((Event.What = evKeyDown) and (Event.KeyCode = kbDown)) then
  begin
    ClearEvent(Event);
    SelectNext(False);
  end;

  ( Respond to Up Arrow Key )
  if ((Event.What = evKeyDown) and (Event.KeyCode = kbUp)) then
  begin
    ClearEvent(Event);
    SelectNext(True);
  end;
end;

```

```

if Event.What=evBroadcast then
begin
  case Event.Command of
    cmFormat1 : begin
      ( From Query1Dialog - general2.pas )
      ClearEvent(Event);
      Format1 := True;
      end;
    cmFormat2 : begin
      ( From Query1Dialog - general2.pas )
      ClearEvent(Event);
      Format2 := True;
      end;
    cmFormat3 : begin
      ( From Query1Dialog - general2.pas )
      ClearEvent(Event);
      Format3 := True;
      end;
    cmFormat4 : begin
      ( From Query1Dialog - general2.pas )
      ClearEvent(Event);
      Format4 := True;
      end;
  end;( Case )
end;

TDialog.HandleEvent(Event);
if Event.What=evCommand then
begin
  case Event.Command of
    cmViewAll : ViewStream;
    cmAddApps : AddApplications;
    cmEditApps : EditApplications;
    cmDelApps : DeleteApplications;
  end;( Case )
  ClearEvent(Event);
end;

end; ( HandleEvent )

(*****
(* View Stream *)
(*****

procedure TDBaseDialog.ViewStream;

var
  outfile      : Text;
  i             : Integer;
  P             : PAppObject;
  Name         : PathStr;
  Show1Window  : PDemoWindow;
  R            : TRect;
  Bruce        : PView;
  DialogChoose : PChooseDialog;
  C            : Word;

begin
  ( Stream is loaded with Init )

```

```

Assign(outfile,'outtext.txt');
Rewrite(outfile);
Writeln(outfile);
Writeln(outfile,'      Spray Applications for this Orchard');
Writeln(outfile);
Writeln(outfile);
Writeln(outfile);
Writeln(outfile);

{ Print out object records }
for i:=0 to OrchardAppList^.Count -1 do
begin
  P:=OrchardAppList^.At(i);
  Writeln(outfile,'      ');
  P^.Date^,'      ':10-Length(P^.Date^),
  P^.Problem^,'      ':10-Length(P^.Problem^),
  P^.Method^,'      ':10-Length(P^.Method^),
  P^.TypeApp^,'      ':10-Length(P^.TypeApp^),
  P^.Chemical^,'      ':10-Length(P^.Chemical^));
end;

Writeln(outfile);Writeln(outfile);Writeln(outfile);
System.Close(outfile);

{ Show select dialog }

R.Assign(10, 3, 61, 21);
DialogChoose := New(PChooseDialog, Init(R, 'Output Configuration'));
with DialogChoose^ do
begin
  { CheckBoxes 1 }
  R.Assign(3, 3, 19, 6);
  Bruce := New(PCheckBoxes, Init(R,
    NewSItem('~F-oliars',
      NewSItem('~W-eedkiller',
        NewSItem('~F-ertilizer',
          nil)))
  ));
  Insert(Bruce);
  R.Assign(2, 2, 14, 3);
  Insert(New(PLabel, Init(R, 'Spray Types', Bruce)));

  { CheckBoxes 2 }
  R.Assign(23, 3, 45, 12);
  Bruce := New(PCheckBoxes, Init(R,
    NewSItem('~D-ate',
      NewSItem('~P-robem',
        NewSItem('~M-ethod',
          NewSItem('~C-hemical',
            NewSItem('~C-oncentration',
              NewSItem('~Q-quantity/Tree',
                NewSItem('~Q-quantity/Orchard',
                  NewSItem('~U-nit Cost',
                    NewSItem('~A-pplication Cost',
                      nil))))))))))
  ));
  Insert(Bruce);
  R.Assign(22, 2, 43, 3);
  Insert(New(PLabel, Init(R, 'Information Required', Bruce)));

```

```

  R.Assign(3, 13, 37, 14);
  Bruce := New(PInputLine, Init(R,28));
  Insert(Bruce);
  R.Assign(2, 12,19, 13);
  Insert(New(PLabel, Init(R, 'Instruction', Bruce)));

  { Buttons }
  R.Assign(2, 15, 16, 17);
  Insert(New(PButton, Init(R, '~C~ancel', cmCancel, bfNormal)));
  R.Assign(18, 15, 32, 17);
  Insert(New(PButton, Init(R, '~P~rint', cmOK, bfNormal)));
  R.Assign(34, 15, 48, 17);
  Insert(New(PButton, Init(R, '~V~iew', cmOK, bfDefault)));
end;

with DialogChooseData do
begin
  CheckboxData1 := 3;
  CheckboxData2 := 6;
  InputLineData := 'Select configuration.';
end;

DialogChoose^.SetData(DialogChooseData);
C := DeskTop^.ExecView(DialogChoose);
if C <> cmCancel then DialogChoose^.GetData(DialogChooseData);
Dispose(DialogChoose, Done);

{ Check bits using bits.pas whether bits for checkboxdata1/2 are set }

{ Show scroll window }

if C <> cmCancel then
begin
  Name := 'outtext.txt';
  WinCount :=1;
  Desktop^.GetExtent(R);
  Show1Window := New(PDemoWindow, Init(R, 'Spray Applications', WinCount, N
ame));
  Desktop^.Insert(Show1Window);
end;
end;

(*****
(* Add Applications
*****)

procedure TDBaseDialog.AddApplications;
var
  RecordNumber      : LongInt;
  TD1               : PEditDialog1;
  Event             : TEvent;
  Action,c1         : Word;
  PNewAppObject     : PAppObject;
  Date1,Problem1,Method1,TypeApp1,Chemical1,Concentration1,
  QuantityPerTree1,QuantityPerOrch1,UnitCost1 : String40;
  ConcReal,QPTReal,OPOReal,UnitCReal : Real;
  ErrorNo,NewCode,RecCode,i         : Integer;

```



```

begin
  Action := cmYes;
  repeat
    with TDinfo1 do
      begin
        Instruction      := 'Enter New Information';
        Date             := '';
        Problem          := '';
        Method           := '';
        TypeApp          := '';
        Chemical         := '';
        Concentration    := '';
        QuantityPerTree  := '';
        QuantityPerOrch  := '';
        UnitCost         := '';
      end; { TDinfo1 }

    { Get information from EditDialog1 }

    TD1:= New(PEditDialog1,Init);
    TD1^.SetData(TDinfo1);
    c1:=Desktop^.ExecView(TD1);
    if c1 <> cmCancel then TD1^.GetData(TDinfo1);
    Dispose(TD1,Done);

    { Test and Insert Information }
    if c1 <> cmCancel then
      begin
        Date1           :=TDinfo1.Date           ;
        Problem1        :=TDinfo1.Problem        ;
        Method1         :=TDinfo1.Method         ;
        TypeApp1        :=TDinfo1.TypeApp        ;
        Chemical1       :=TDinfo1.Chemical       ;
        Concentration1  :=TDinfo1.Concentration  ;
        QuantityPerTree1:=TDinfo1.QuantityPerTree;
        QuantityPerOrch1:=TDinfo1.QuantityPerOrch;
        UnitCost1      :=TDinfo1.UnitCost      ;

        NoBlanks(Date1);
        NoBlanks(Problem1);
        NoBlanks(Method1);
        NoBlanks(TypeApp1);
        NoBlanks(Chemical1);
        NoBlanks(Concentration1);
        NoBlanks(QuantityPerTree1);
        NoBlanks(QuantityPerOrch1);
        NoBlanks(UnitCost1);

        Val(Concentration1 ,ConcReal ,ErrorNo);
        Val(QuantityPerTree1 ,QPTReal ,ErrorNo);
        Val(QuantityPerOrch1 ,OPORReal ,ErrorNo);
        Val(UnitCost1 ,UnitCReal,ErrorNo);

        { Look for maximum code in list then increment by one }
        NewCode :=0;
        for i:= 0 to OrchardAppList^.Count-1 do
          begin
            RecCode := PAppObject(OrchardAppList^.At(i))^PCode^;
            if RecCode > NewCode then NewCode := RecCode;
          end;

```

```

        NewCode:=NewCode+1;

        { Save application data }
        PNewAppObject := New(PAppObject,Init(Date1,Problem1,Method1,TypeApp1,
                                              Chemical1,ConcReal,QPTReal,OPORReal,
                                              UnitCReal,NewCode));

        OrchardAppList^.Insert(PNewAppObject);

        OrchardAppStream.Init(AppStreamName,stCreate,1024);
        OrchardAppStream.Put(OrchardAppList);
        OrchardAppStream.Done; { Shut down stream }

        Action := MessageBox(#3' Add another record? ', nil,mfYesNoCancel);
      end
    else
      Action := MessageBox(#3'Duplicate code exists. Try another code? ',
                          nil,mfYesNoCancel);
    { end;} {cmCancel}
    until (Action = cmNo) or (Action = cmCancel);

  end; { AddApplications }

  (*****
  (* Edit Applications *)
  (*****
  procedure TDBaseDialog.EditApplications;
  var
    PQD                : PQuery1Dialog;
    Collection1         : PDataCollection4;
    Date2,Problem2,TypeApp2,Chemical2 : String;
    ArrayStr,Msg        : String;
    i,EditAppNo,ErrorNo,FormatNo      : Integer;
    Pld                          : PListADialog;
    Option,Control1,Control2,Control3 : Word;
    ConcReal,QPTReal,QPORReal,UCReal  : Real;
    ConcStr,QPTStr,QPOStr,UCStr       : String;
    QuantityPerTree1,QuantityPerOrch1,UnitCost1 : String40;
    Concentration1,Date1,Problem1,Method1,TypeApp1,Chemical1 : String40;
    TD1                          : PEditDialog1;
    Code2,CodeReturned           : Integer;
    FoundApp                     : PAppObject;

    function CodeMatch(App:PAppObject):Boolean;far;
    begin
      CodeMatch := CodeReturned = App^.PCode^;
    end;

  begin
    Option := cmYes;
    repeat
      Format1 := False;
      Format2 := False;
      Format3 := False;
      Format4 := False;
      Control1 :=0;
      Control2 :=0;
      Control3 :=0;

      { Get format for search }

```

```

PQD := New(PQuery1Dialog, Init);
Control1 := Desktop^.ExecView(PQD);
if Control1 <> cmCancel then
  Dispose(PQD, Done);

if Control1 <> cmCancel then
begin
  { Read in data for each App as a string, compile string and insert }
  Collection1 := New(PDataCollection4, Init(OrchardAppList^.Count-1, 5));
  Collection1^.Duplicates := True;

  if format1 = true then
  begin
    for i:= 0 to OrchardAppList^.Count-1 do
    begin
      Date2      := PAppObject(OrchardAppList^.At(i))^Date^;
      Problem2   := PAppObject(OrchardAppList^.At(i))^Problem^;
      TypeApp2   := PAppObject(OrchardAppList^.At(i))^TypeApp^;
      Chemical2  := PAppObject(OrchardAppList^.At(i))^Chemical^;
      Code2      := PAppObject(OrchardAppList^.At(i))^PCode^;

      FillBlanks(Date2, 8);
      FillBlanks(Problem2, 10);
      FillBlanks(TypeApp2, 12);
      FillBlanks(Chemical2, 18);
      ArrayStr:= Date2+ ' '+Problem2+ ' '+TypeApp2+ ' '+Chemical2;

      Collection1^.Insert(New(PRecD4, Init(ArrayStr, Code2)));
    end;
  end
  else if format2 = true then
  begin
    for i:= 0 to OrchardAppList^.Count-1 do
    begin
      Date2      := PAppObject(OrchardAppList^.At(i))^Date^;
      Problem2   := PAppObject(OrchardAppList^.At(i))^Problem^;
      TypeApp2   := PAppObject(OrchardAppList^.At(i))^TypeApp^;
      Chemical2  := PAppObject(OrchardAppList^.At(i))^Chemical^;
      Code2      := PAppObject(OrchardAppList^.At(i))^PCode^;

      FillBlanks(Date2, 8);
      FillBlanks(Problem2, 10);
      FillBlanks(TypeApp2, 12);
      FillBlanks(Chemical2, 18);
      ArrayStr:= Problem2+ ' '+Date2+ ' '+TypeApp2+ ' '+Chemical2;

      Collection1^.Insert(New(PRecD4, Init(ArrayStr, Code2)));
    end;
  end
  else if format3 = true then
  begin
    for i:= 0 to OrchardAppList^.Count-1 do
    begin
      Date2      := PAppObject(OrchardAppList^.At(i))^Date^;
      Problem2   := PAppObject(OrchardAppList^.At(i))^Problem^;
      TypeApp2   := PAppObject(OrchardAppList^.At(i))^TypeApp^;
      Chemical2  := PAppObject(OrchardAppList^.At(i))^Chemical^;
      Code2      := PAppObject(OrchardAppList^.At(i))^PCode^;

      FillBlanks(Date2, 8);

```

```

      FillBlanks(Problem2, 10);
      FillBlanks(TypeApp2, 12);
      FillBlanks(Chemical2, 18);
      ArrayStr:= TypeApp2+ ' '+Date2+ ' '+Problem2+ ' '+Chemical2;

      Collection1^.Insert(New(PRecD4, Init(ArrayStr, Code2)));
    end;
  end
  else { Format4 is true }
  begin
    for i:= 0 to OrchardAppList^.Count-1 do
    begin
      Date2      := PAppObject(OrchardAppList^.At(i))^Date^;
      Problem2   := PAppObject(OrchardAppList^.At(i))^Problem^;
      TypeApp2   := PAppObject(OrchardAppList^.At(i))^TypeApp^;
      Chemical2  := PAppObject(OrchardAppList^.At(i))^Chemical^;
      Code2      := PAppObject(OrchardAppList^.At(i))^PCode^;

      FillBlanks(Date2, 8);
      FillBlanks(Problem2, 10);
      FillBlanks(TypeApp2, 12);
      FillBlanks(Chemical2, 18);
      ArrayStr:= Chemical2+ ' '+Date2+ ' '+Problem2+ ' '+TypeApp2;

      Collection1^.Insert(New(PRecD4, Init(ArrayStr, Code2)));
    end;
  end;

  { Display ListADialog for Orchard Row Selection }

  FormatNo:=0;
  if format1 = true then FormatNo := 1;
  if format2 = true then FormatNo := 2;
  if format3 = true then FormatNo := 3;
  if format4 = true then FormatNo := 4;

  Pld := New(PListADialog, Init(Collection1, FormatNo));
  Control2 := Desktop^.ExecView(Pld);
  if Control2 <> cmCancel then
  begin
    Pld^.GetData(TransWCode);
    CodeReturned := TransWCode.RecCode;
  end;
  Dispose(Pld, Done);

  if Control2 <> cmCancel then
  begin
    { Search for Application with the code returned }

    FoundApp := OrchardAppList^.FirstThat(@CodeMatch);
    EditAppNo := OrchardAppList^.IndexOf(FoundApp);

    { Correct Application corresponding to corect code found }
    Str(PAppObject(OrchardAppList^.At(EditAppNo))^Concentration :5:2, Con
cStr);
    Str(PAppObject(OrchardAppList^.At(EditAppNo))^QuantityPerTree:5:2, QPT
Str );
    Str(PAppObject(OrchardAppList^.At(EditAppNo))^QuantityPerOrch:5:2, QPO
Str );
    Str(PAppObject(OrchardAppList^.At(EditAppNo))^UnitCost :5:2, UCS

```

```

tr );

    with TDinfo1 do
    begin
        Instruction      := 'Modify existing Record';
        Date             := PAppObject(OrchardAppList^.At(EditAppNo)).Date
^;
        Problem          := PAppObject(OrchardAppList^.At(EditAppNo)).Prob
lem^;
        Method           := PAppObject(OrchardAppList^.At(EditAppNo)).Meth
od^;
        TypeApp          := PAppObject(OrchardAppList^.At(EditAppNo)).Type
App^;
        Chemical         := PAppObject(OrchardAppList^.At(EditAppNo)).Chem
ical^;
        Concentration    := ConcStr;
        QuantityPerTree  := QPTStr;
        QuantityPerOrch  := QPOStr;
        UnitCost         := UCStr;
    end; { TDinfo1 }

    { Get information from EditDialog1 }

    TD1:= New(PEditDialog1,Init);
    TD1^.SetData(TDinfo1);
    Control3:=Desktop^.ExecView(TD1);
    if Control3 <> cmCancel then TD1^.GetData(TDinfo1);
    Dispose(TD1,Done);

    { Test and Insert Information }
    if Control3 <> cmCancel then
    begin
        Date1             :=TDinfo1.Date           ;
        Problem1          :=TDinfo1.Problem         ;
        Method1           :=TDinfo1.Method          ;
        TypeApp1          :=TDinfo1.TypeApp          ;
        Chemical1         :=TDinfo1.Chemical         ;
        Concentration1    :=TDinfo1.Concentration    ;
        QuantityPerTree1  :=TDinfo1.QuantityPerTree  ;
        QuantityPerOrch1  :=TDinfo1.QuantityPerOrch  ;
        UnitCost1         :=TDinfo1.UnitCost        ;

        NoBlanks(Date1);
        NoBlanks(Problem1);
        NoBlanks(Method1);
        NoBlanks(TypeApp1);
        NoBlanks(Chemical1);
        NoBlanks(Concentration1);
        NoBlanks(QuantityPerTree1);
        NoBlanks(QuantityPerOrch1);
        NoBlanks(UnitCost1);

        Val(Concentration1 ,ConcReal ,ErrorNo);
        Val(QuantityPerTree1 ,QPTReal ,ErrorNo);
        Val(QuantityPerOrch1 ,QPOReal ,ErrorNo);
        Val(UnitCost1 ,UCReal ,ErrorNo);

        PAppObject(OrchardAppList^.At(EditAppNo)).Date^ :=Date1;
        PAppObject(OrchardAppList^.At(EditAppNo)).Problem^ :=Problem1;
        PAppObject(OrchardAppList^.At(EditAppNo)).Method^ :=Method1;

```

```

        PAppObject(OrchardAppList^.At(EditAppNo)).TypeApp^ :=TypeApp1;
        PAppObject(OrchardAppList^.At(EditAppNo)).Concentration := ConcRea
al;
        PAppObject(OrchardAppList^.At(EditAppNo)).QuantityPerTree := QPTRea
l;
        PAppObject(OrchardAppList^.At(EditAppNo)).QuantityPerOrch := QPORea
l;
        PAppObject(OrchardAppList^.At(EditAppNo)).UnitCost := UCReal
;

        { Update Stream }
        OrchardAppStream.Init(AppStreamName,stCreate,1024);
        OrchardAppStream.Put(OrchardAppList);
        OrchardAppStream.Done; { Shut down stream }

        end; { Control3 }
        end; { Control2 }
        end; { Control1 }

        Msg := #3 + ' Edit another record ? ' ;
        Option := MessageBox(Msg, nil,mfYesNoCancel);

        until (Option = cmNo) or (Option = cmCancel);

    end; { EditApplications }

    (*****
    (* Delete Applications *)
    (*****
    procedure TDBaseDialog.DeleteApplications;

    var
        PQD                      : PQuery1Dialog;
        Collection1              : PDataCollection4;
        Date2,Problem2,TypeApp2,Chemical2 : String;
        ArrayStr,Msg             : String;
        i,EditAppNo,ErrorNo,FormatNo : Integer;
        Pld                      : PListADialog;
        Action,Control1,Control2  : Word;
        ConcReal,QPTReal,QPOReal,UCReal : Real;
        ConcStr,QPTStr,QPOStr,UCStr   : String;
        QuantityPerTree1,QuantityPerOrch1,UnitCost1 : String40;
        Concentration1,Date1,Problem1,Method1,TypeApp1,Chemical1 : String40;
        TD1                     : PEditDialog1;
        Code2,CodeReturned       : Integer;
        FoundApp                 : PAppObject;

        function CodeMatch(App:PAppObject):Boolean;far;
        begin
            CodeMatch := CodeReturned = App^.PCode^;
        end;

    begin
        Action := cmYes;
        repeat

            Format1 := False;
            Format2 := False;
            Format3 := False;
            Format4 := False;

```

```

{ Get format for search }
PQD := New(PQuery1Dialog,Init);
Control1 := Desktop^.ExecView(PQD);
if Control1 <> cmCancel then
Dispose(PQD, Done);

{ Read in data for each App as a string, compile string and insert }
Collection1 := New(PDataCollection4,Init(OrchardAppList^.Count-1, 5));
Collection1^.Duplicates := True;

{ }
if format1 = true then
begin
  for i:= 0 to OrchardAppList^.Count-1 do
  begin
    Date2      := PAppObject(OrchardAppList^.At(i))^Date^;
    Problem2    := PAppObject(OrchardAppList^.At(i))^Problem^;
    TypeApp2    := PAppObject(OrchardAppList^.At(i))^TypeApp^;
    Chemical2   := PAppObject(OrchardAppList^.At(i))^Chemical^;
    Code2       := PAppObject(OrchardAppList^.At(i))^PCode^;

    FillBlanks(Date2,8);
    FillBlanks(Problem2,10);
    FillBlanks(TypeApp2,12);
    FillBlanks(Chemical2,18);
    ArrayStr:=Date2+' '+Problem2+' '+TypeApp2+' '+Chemical2;

    Collection1^.Insert(New(PRecD4,Init(ArrayStr,Code2)));
  end;
end
else if format2 = true then
begin
  for i:= 0 to OrchardAppList^.Count-1 do
  begin
    Date2      := PAppObject(OrchardAppList^.At(i))^Date^;
    Problem2    := PAppObject(OrchardAppList^.At(i))^Problem^;
    TypeApp2    := PAppObject(OrchardAppList^.At(i))^TypeApp^;
    Chemical2   := PAppObject(OrchardAppList^.At(i))^Chemical^;
    Code2       := PAppObject(OrchardAppList^.At(i))^PCode^;

    FillBlanks(Date2,8);
    FillBlanks(Problem2,10);
    FillBlanks(TypeApp2,12);
    FillBlanks(Chemical2,18);
    ArrayStr:= Problem2+' '+Date2+' '+TypeApp2+' '+Chemical2;

    Collection1^.Insert(New(PRecD4,Init(ArrayStr,Code2)));
  end;
end
else if format3 = true then
begin
  for i:= 0 to OrchardAppList^.Count-1 do
  begin
    Date2      := PAppObject(OrchardAppList^.At(i))^Date^;
    Problem2    := PAppObject(OrchardAppList^.At(i))^Problem^;
    TypeApp2    := PAppObject(OrchardAppList^.At(i))^TypeApp^;
    Chemical2   := PAppObject(OrchardAppList^.At(i))^Chemical^;
    Code2       := PAppObject(OrchardAppList^.At(i))^PCode^;

```

```

    FillBlanks(Date2,8);
    FillBlanks(Problem2,10);
    FillBlanks(TypeApp2,12);
    FillBlanks(Chemical2,18);
    ArrayStr:= TypeApp2+' '+Date2+' '+Problem2+' '+Chemical2;

    Collection1^.Insert(New(PRecD4,Init(ArrayStr,Code2)));
  end;
end
else { Format4 is true }
begin
  for i:= 0 to OrchardAppList^.Count-1 do
  begin
    Date2      := PAppObject(OrchardAppList^.At(i))^Date^;
    Problem2    := PAppObject(OrchardAppList^.At(i))^Problem^;
    TypeApp2    := PAppObject(OrchardAppList^.At(i))^TypeApp^;
    Chemical2   := PAppObject(OrchardAppList^.At(i))^Chemical^;
    Code2       := PAppObject(OrchardAppList^.At(i))^PCode^;

    FillBlanks(Date2,8);
    FillBlanks(Problem2,10);
    FillBlanks(TypeApp2,12);
    FillBlanks(Chemical2,18);
    ArrayStr:= Chemical2+' '+Date2+' '+Problem2+' '+TypeApp2;

    Collection1^.Insert(New(PRecD4,Init(ArrayStr,Code2)));
  end;
end;

{ Display ListADialog for Orchard Row Selection }

FormatNo:=0;
if format1 = true then FormatNo := 1;
if format2 = true then FormatNo := 2;
if format3 = true then FormatNo := 3;
if format4 = true then FormatNo := 4;

Pld := New(PListADialog,Init(Collection1,FormatNo));
Control1 := Desktop^.ExecView(Pld);
if Control1 <> cmCancel then
begin
  Pld^.GetData(TransWCode);
  CodeReturned := TransWCode.RecCode;
end;
Dispose(Pld, Done);

if Control1 <> cmCancel then
begin
  { Search for Application with the code returned }

  FoundApp := OrchardAppList^.FirstThat(@CodeMatch);
  EditAppNo := OrchardAppList^.IndexOf(FoundApp);

  { Correct Application corresponding to corect code found }
  Str(PAppObject(OrchardAppList^.At(EditAppNo))^Concentration :5:2, ConcStr
r);
  Str(PAppObject(OrchardAppList^.At(EditAppNo))^QuantityPerTree:5:2, QPTStr
);
  Str(PAppObject(OrchardAppList^.At(EditAppNo))^QuantityPerOrch:5:2, QPOStr
);

```

```

); Str(PAppObject(OrchardAppList^.At(EditAppNo))^UnitCost :5:2, UCStr
);
with TDinfo1 do
begin
  Instruction := 'Modify existing Record';
  Date       := PAppObject(OrchardAppList^.At(EditAppNo))^Date^;
  Problem    := PAppObject(OrchardAppList^.At(EditAppNo))^Problem^;
^;
  Method     := PAppObject(OrchardAppList^.At(EditAppNo))^Method^;
;
  TypeApp    := PAppObject(OrchardAppList^.At(EditAppNo))^TypeApp^;
^;
  Chemical   := PAppObject(OrchardAppList^.At(EditAppNo))^Chemica
l^;
  Concentration := ConcStr;
  QuantityPerTree := QPTStr;
  QuantityPerOrch := QPOStr;
  UnitCost      := UCStr;
end; { TDinfo1 }

{ Get information from EditDialog1 }

TD1:= New(PEditDialog1,Init);
TD1^.SetData(TDinfo1);
Control2:=Desktop^.ExecView(TD1);
if Control2 <> cmCancel then TD1^.GetData(TDinfo1);
Dispose(TD1,Done);

{ Test and Insert Information }
if Control2 <> cmCancel then
begin
  Date1       :=TDinfo1.Date       ;
  Problem1    :=TDinfo1.Problem    ;
  Method1     :=TDinfo1.Method     ;
  TypeApp1    :=TDinfo1.TypeApp    ;
  Chemical1   :=TDinfo1.Chemical   ;
  Concentration1 :=TDinfo1.Concentration ;
  QuantityPerTree1 :=TDinfo1.QuantityPerTree;
  QuantityPerOrch1 :=TDinfo1.QuantityPerOrch;
  UnitCost1   :=TDinfo1.UnitCost   ;

  NoBlanks(Date1);
  NoBlanks(Problem1);
  NoBlanks(Method1);
  NoBlanks(TypeApp1);
  NoBlanks(Chemical1);
  NoBlanks(Concentration1);
  NoBlanks(QuantityPerTree1);
  NoBlanks(QuantityPerOrch1);
  NoBlanks(UnitCost1);

  Val(Concentration1 ,ConcReal ,ErrorNo);
  Val(QuantityPerTree1 ,QPTReal ,ErrorNo);
  Val(QuantityPerOrch1 ,QPOReal ,ErrorNo);
  Val(UnitCost1 ,UCReal ,ErrorNo);

  PAppObject(OrchardAppList^.At(EditAppNo))^Date^ :=Date1;
  PAppObject(OrchardAppList^.At(EditAppNo))^Problem^ :=Problem1;
  PAppObject(OrchardAppList^.At(EditAppNo))^Method^ :=Method1;

```

```

PAppObject(OrchardAppList^.At(EditAppNo))^TypeApp^ :=TypeApp1;
PAppObject(OrchardAppList^.At(EditAppNo))^Concentration := ConcReal;

PAppObject(OrchardAppList^.At(EditAppNo))^QuantityPerTree := QPTReal;
PAppObject(OrchardAppList^.At(EditAppNo))^QuantityPerOrch := QPOReal;
PAppObject(OrchardAppList^.At(EditAppNo))^UnitCost := UCReal;

{ Update Stream }
OrchardAppStream.Init(AppStreamName,stCreate,1024);
OrchardAppStream.Put(OrchardAppList);
OrchardAppStream.Done; { Shut down stream }

end; { Control2 }
end; { Control1 }
Msg := #3 + ' Edit another record ? ' ;
Action := MessageBox(Msg, nil,mfYesNoCancel);
until (Action = cmNo) or (Action = cmCancel);

end; { DeleteApplications }

(*****
(*                      Noblanks                      *)
*****)
Procedure TDbaseDialog.Noblanks(var S:String40);
var j : integer;
begin
  j:=1;
  while (j <=length(s)) do
  begin
    if s[j] = ' ' then System.Delete(s, j, 1)
    else inc(j);
  end;
end; { Noblanks }

(*****
(*                      Fill Blanks                      *)
*****)
Procedure TDbaseDialog.FillBlanks(var S:String; StrLength:Integer);
var j,l : integer;
begin
  j:=length(s);
  l:=StrLength;
  if j < l then
  begin
    while j < l do
    begin
      inc(j);
      s[j] := chr(32);
    end;
  end;
  if j > l then j:=l;
  s[0]:=chr(l);
end; { FillBlanks }

(*****
(*                      General Procedures                      *)
*****)

end. { Unit }

```

```

{*****}
{
{   Program for compiling and testing the   }
{   TBatchObject stream                     }
{
{   Filename : BachComp.pas   Author K G Mercer }
{   Date      : September 1993 }
{
{*****}

```

```

program BachComp;

```

```

uses
  Objects, StrmUnit, Crt;

```

```

var
  BatchRez      : TResourceFile;
  BatchList     : PCollection;
  RezStream     : PBatchStream;
  List1, List2, List3: PCollection;
  List4, List5, List6: PCollection;
  RezName       : String[12];
  Name          : String;

```

```

begin
  ClrScr;
  RezName := 'Batch.Rez';
  ExpertObjectStreamRegistration;

  { Make the collection and display it }

  RezStream := New(PBatchStream, Init(RezName, stCreate, 1024));
  BatchRez.Init(RezStream);

  MakeBatchCollection(BatchList);
  WriteBatchObjects(BatchList);
  BatchRez.Put(BatchList, 'KeyNames');

  MakeBatchDataCollection(List1, List2, List3, List4, List5, List6);
  BatchRez.Put(List1, 'DuNoon' );
  BatchRez.Put(List2, 'Milnerton' );
  BatchRez.Put(List3, 'SandyBay' );
  BatchRez.Put(List4, 'Clifton' );
  BatchRez.Put(List5, 'SeaPoint' );
  BatchRez.Put(List6, 'Somerset West');

  writeln;
  WriteBatchDataObjects(List1);
  WriteBatchDataObjects(List2);
  WriteBatchDataObjects(List3);
  WriteBatchDataObjects(List4);
  WriteBatchDataObjects(List5);
  WriteBatchDataObjects(List6);
  writeln;

  Dispose(List1, Done);
  Dispose(List2, Done);
  Dispose(List3, Done);
  Dispose(List4, Done);
  Dispose(List5, Done);

```

```

  Dispose(List6, Done);

  Name := PBatchObject(BatchList^.At(1))^ .BatchCallName^;
  List2 := PCollection(BatchRez.Get(Name));
  Writeln(name);
  WriteBatchDataObjects(List2);

  Dispose(List2, Done);
  Dispose(BatchList, Done);
  BatchRez.Done;

```

```

end.

```

```
{*****}
{
{   Program for listing the TBatchObject stream   }
{
{   Filename : BachList.pas   Author K G Mercer   }
{   Date    : September 1993   }
{
{*****}
```

```
program BachList;
```

```
uses
  Objects, StrmUnit, Crt, MsgBox;
```

```
var
  BatchRez      : TResourceFile;
  BatchList     : PCollection;
  RezStream     : PBatchStream;
  List1, List2, List3: PCollection;
  List4, List5, List6: PCollection;
  RezName       : String[12];
  Name          : String;
  CallKeyName   : String;
  i, CallKeyCode : Integer;
```

```
begin
  ClrScr;
  RezName := 'Batch.Rez';
  ExpertObjectStreamRegistration;

  { Make the collection and display it }

  BatchRez.Init(New(PBufStream, Init(RezName, stOpen, 1024)));
  if BatchRez.Stream^.Status <> 0 then
    MessageBox(#3'Error loading stream', nil, mfInformation + mfOkButton);

  BatchList := PCollection(BatchRez.Get('KeyNames'));
  WriteBatchObjects(BatchList);

  writeln;
  for i:= 0 to BatchList^.Count-1 do
  begin
    CallKeyName := PBatchObject(BatchList^.At(i))^ .BatchCallName^;
    CallKeyCode := PBatchObject(BatchList^.At(i))^ .PCode^;
    writeln('ListKey name : ', CallKeyName, ' ', CallKeyCode);
    WriteBatchDataObjects(PCollection(BatchRez.Get(CallKeyName)));
    writeln;
  end;

  Dispose(BatchList, Done);
  BatchRez.Done;
end.
```

```

(*****)
(
(   Commands for all the Units   )
(
(   Filename : CmndList.pas   Author K G Mercer   )
(   Date      : September 1993   )
(
(*****)
Unit Cmndlist;

{$D-}
{$O+}
{$F+}

interface

uses Drivers;

const
(   Range 0..99      reserved,   can be disabled )
(   Range 100..255   unreserved, can be disabled )
(   Range 256..999   reserved,   cannot be disabled )
(   Range 1000..65535 unreserved, cannot be disabled )

(   History list IDs           )

    hlChangeDir      = 1;
    hlOpenListDlg    = 2;

(   Ticks for screen saver 18.2 ticks/sec )

    SaveTicks        = 2185;

(   Misc UI commands           )

    cmFileOpen        = 100;
    cmNewWin          = 101;
    cmAnalyse1        = 102;
    cmAnalyse2        = 103;
    cmAnalyse3        = 104;
    cmAnalyse4        = 105;
    cmAnalyse5        = 106;
    cmAnalyse6        = 107;

(   Commands for Data Base   )

    cmListAll         = 120;
    cmFindRec         = 121;
    cmSrchRec         = 122;
    cmNextRec         = 123;
    cmPrevRec         = 124;
    cmAddRec          = 125;
    cmUpdateRec       = 126;
    cmDelRec          = 127;
    cmRebuild         = 128;
    cmInxDBBase1      = 129;
    cmInxDBBase2      = 130;
    cmRAnalysis        = 131;
    cmReadIn          = 132;
    cmCrum            = 133;

```

```

    cmScrTog          = 134;

    cmTrigger         = 140;
    cmNone            = 141;
    cmtextField       = 142;
    cmColorChg        = 143;
    cmExSystem1       = 144;
    cmExSystem2       = 145;
    cmExSystem3       = 146;
    cmRBaseShow       = 147;
    cmRBaseEdit       = 148;
    cmRBaseSYN        = 149;
    cmShutDown10      = 150;
    cmSelect          = 151;
    cmViewAll         = 152;
    cmEditRec         = 153;
    cmDeleteRow       = 154;
    cmAddRow          = 155;
    cmRunRec          = 156;
    cmShutDown12      = 157;
    cmTry             = 158;
    cmBatchFiles      = 159;
    cmAddBatchObject  = 160;
    cmDelBatchObject  = 161;
    cmAddDataBatchObject = 162;
    cmDelDataBatchObject = 163;
    cmEditWeight      = 164;
    cmInfo            = 165;
    cmRunBAREC        = 166;
    cmDoGraphics      = 167;
    cmCheckAnExit     = 168;
    cmCheckWeight     = 169;

(   List & form-oriented commands )
(   (Cannot be disabled)         )

    cmChangeDir       = 2000;
    cmVideoMode       = 2001;
    cmDosShell        = 2002;
    cmAboutBox1       = 2003;
    cmTempExit        = 2004;

(   List & form-oriented commands )
(   (Cannot be disabled)         )

    cmListOpen        = 3000;
    cmFormEdit        = 3001;
    cmFormNew         = 3002;
    cmFormSave        = 3003;
    cmFormDel         = 3004;
    cmListSave        = 3005;

(   Broadcast commands           )

    cmTopForm         = 3050;
    cmRegisterForm    = 3051;
    cmEditingForm     = 3052;
    cmCanCloseForm    = 3053;
    cmCloseForm       = 3054;
    cmTopList         = 3055;

```



```
cmEditingFile      = 3056;  
cmBuild            = 3057;  
cmExtractData1     = 3060;  
cmExtractData2     = 3061;  
cmExtractData1Complete = 3063;  
cmExtractData2Complete = 3064;  
cmTransfer1        = 3065;  
cmShowAll          = 3066;  
cmCloseInfoWindow = 3067;  
cmShutDown1        = 3069;  
cmShutDown2        = 3070;  
cmCodeWanted       = 3071;  
cmFileDescrip       = 3072;  
cmUpdatePopILine1 = 3073;  
cmPopWindow1       = 3074;  
cmShutDown20       = 3075;  
cmShutDown30       = 3076;  
cmNewBatchObjBrdcst = 3077;  
cmDelBatchObjBrdcst = 3078;  
cmUpdatePane2      = 3079;  
cmBAddDataBatchObject = 3080;  
cmFileDescrip2      = 3081;
```

```
{ cmGrIdle registered in GrDial.pas }
```

```
implementation
```

```
end.
```

```
{*****}
{
{   Unit for control program (control.pas)   }
{                                           }
{   Filename : CntrlUnt.pas   Author K G Mercer }
{   Date      : September 1993 }
{                                           }
{*****}
unit Cntrlunt;

interface

uses Dos;

Const
    CntrlFileName ='Ctrlinfo.dat';
    MaxMembers = 5;

Type
    RecordTypes = (FinRec,InterRec);

    Member = Record
        WhereTo      : string[15];
        Parameter    : string[50];
        TypeRec      : RecordTypes;
    end;

implementation

end.
```

```

{*****}
{
{   Program for listing the TBatchObject stream   }
{
{   Filename : Editors.pas   Author K G Mercer   }
{   Date      : September 1993                 }
{
{
{   Originally from Turbo Pascal Version 6.0 :   }
{   Turbo Vision Unit                           }
{   Modified extensively for use with the program }
{
{*****}

```

```
unit Editors;
```

```
($F+,I-,O+,S-,V-,X+,D-)
```

```
interface
```

```
uses Drivers, Objects, Views;
```

```

const
  cmSave      = 80;
  cmSaveAs    = 81;
  cmFind      = 82;
  cmReplace   = 83;
  cmSearchAgain = 84;

```

```

const
  cmCharLeft  = 500;
  cmCharRight = 501;
  cmWordLeft  = 502;
  cmWordRight = 503;
  cmLineStart = 504;
  cmLineEnd   = 505;
  cmLineUp    = 506;
  cmLineDown  = 507;
  cmPageUp    = 508;
  cmPageDown  = 509;
  cmTextStart = 510;
  cmTextEnd   = 511;
  cmNewLine   = 512;
  cmBackSpace = 513;
  cmDelChar   = 514;
  cmDelWord   = 515;
  cmDelStart  = 516;
  cmDelEnd    = 517;
  cmDelLine   = 518;
  cmInsMode   = 519;
  cmStartSelect = 520;
  cmHideSelect = 521;
  cmIndentMode = 522;
  cmUpdateTitle = 523;

```

```

const
  edOutOfMemory = 0;
  edReadError   = 1;
  edWriteError  = 2;
  edCreateError = 3;

```

```

  edSaveModify   = 4;
  edSaveUntitled = 5;
  edSaveAs       = 6;
  edFind         = 7;
  edSearchFailed = 8;
  edReplace      = 9;
  edReplacePrompt = 10;

```

```

const
  efCaseSensitive   = $0001;
  efWholeWordsOnly  = $0002;
  efPromptOnReplace = $0004;
  efReplaceAll      = $0008;
  efDoReplace       = $0010;
  efBackupFiles     = $0100;

```

```

const
  CIndicator = #2#3;
  CEditor    = #6#7;
  CMemo      = #26#27;

```

```

const
  MaxLineLength = 256;

```

```

type
  TEditorDialog = function(Dialog: Integer; Info: Pointer): Word;

```

```

type
  PIndicator = ^TIndicator;
  TIndicator = object(TView)
    Location: TPoint;
    Modified: Boolean;
    constructor Init(var Bounds: TRect);
    procedure Draw; virtual;
    function GetPalette: PPalette; virtual;
    procedure SetState(AState: Word; Enable: Boolean); virtual;
    procedure SetValue(ALocation: TPoint; AModified: Boolean);
  end;

```

```

type
  PEditBuffer = ^TEditBuffer;
  TEditBuffer = array[0..65519] of Char;

```

```

type
  PEditor = ^TEditor;
  TEditor = object(TView)
    HScrollBar: PScrollBar;
    VScrollBar: PScrollBar;
    Indicator: PIndicator;
    Buffer: PEditBuffer;
    BufSize: Word;
    BufLen: Word;
    GapLen: Word;
    SelStart: Word;
    SelEnd: Word;
    CurPtr: Word;
    CurPos: TPoint;
    Delta: TPoint;
    Limit: TPoint;
    DrawLine: Integer;

```

```

DrawPtr: Word;
DelCount: Word;
InsCount: Word;
IsValid: Boolean;
CanUndo: Boolean;
Modified: Boolean;
Selecting: Boolean;
Overwrite: Boolean;
AutoIndent: Boolean;
constructor Init(var Bounds: TRect;
  AHScrollBar, AVScrollBar: PScrollBar;
  AIndicator: PIndicator; ABufSize: Word);
constructor Load(var S: TStream);
destructor Done; virtual;
function BufChar(P: Word): Char;
function BufPtr(P: Word): Word;
procedure ChangeBounds(var Bounds: TRect); virtual;
procedure ConvertEvent(var Event: TEvent); virtual;
function CursorVisible: Boolean;
procedure DeleteSelect;
procedure DoneBuffer; virtual;
procedure Draw; virtual;
function GetPalette: PPalette; virtual;
procedure HandleEvent(var Event: TEvent); virtual;
procedure InitBuffer; virtual;
function InsertBuffer(var P: PEditBuffer; Offset, Length: Word;
  AllowUndo, SelectText: Boolean): Boolean;
function InsertFrom(Editor: PEditor): Boolean; virtual;
function InsertText(Text: Pointer; Length: Word;
  SelectText: Boolean): Boolean;
procedure ScrollTo(X, Y: Integer);
function Search(FindStr: String; Opts: Word): Boolean;
function SetBufSize(NewSize: Word): Boolean; virtual;
procedure SetCmdState(Command: Word; Enable: Boolean);
procedure SetSelect(NewStart, NewEnd: Word; CurStart: Boolean);
procedure SetState(ASState: Word; Enable: Boolean); virtual;
procedure Store(var S: TStream);
procedure TrackCursor(Center: Boolean);
procedure Undo;
procedure UpdateCommands; virtual;
function Valid(Command: Word): Boolean; virtual;
private
LockCount: Byte;
UpdateFlags: Byte;
KeyState: Integer;
function CharPos(P, Target: Word): Integer;
function CharPtr(P: Word; Target: Integer): Word;
function ClipCopy: Boolean;
procedure ClipCut;
procedure ClipPaste;
procedure DeleteRange(StartPtr, EndPtr: Word; DelSelect: Boolean);
procedure DoUpdate;
procedure DoSearchReplace;
procedure DrawLines(Y, Count: Integer; LinePtr: Word);
procedure FormatLine(var DrawBuf; LinePtr: Word;
  Width: Integer; Colors: Word);
procedure Find;
function GetMousePtr(Mouse: TPoint): Word;
function HasSelection: Boolean;
procedure HideSelect;

```

```

function IsClipboard: Boolean;
function LineEnd(P: Word): Word;
function LineMove(P: Word; Count: Integer): Word;
function LineStart(P: Word): Word;
procedure Lock;
procedure NewLine;
function NextChar(P: Word): Word;
function NextLine(P: Word): Word;
function NextWord(P: Word): Word;
function PrevChar(P: Word): Word;
function PrevLine(P: Word): Word;
function PrevWord(P: Word): Word;
procedure Replace;
procedure SetBufLen(Length: Word);
procedure SetCurPtr(P: Word; SelectMode: Byte);
procedure StartSelect;
procedure ToggleInsMode;
procedure Unlock;
procedure Update(AFlags: Byte);
end;

type
  TMemData = record
    Length: Word;
    Buffer: TEditBuffer;
  end;

type
  PMemo = ^TMemData;
  TMemData = object(TEditor)
    constructor Load(var S: TStream);
    function DataSize: Word; virtual;
    procedure GetData(var Rec); virtual;
    function GetPalette: PPalette; virtual;
    procedure HandleEvent(var Event: TEvent); virtual;
    procedure SetData(var Rec); virtual;
    procedure Store(var S: TStream);
  end;

type
  PFileEditor = ^TFileEditor;
  TFileEditor = object(TEditor)
    FileName: FNameStr;
    constructor Init(var Bounds: TRect;
      AHScrollBar, AVScrollBar: PScrollBar;
      AIndicator: PIndicator; AFileName: FNameStr);
    constructor Load(var S: TStream);
    procedure DoneBuffer; virtual;
    procedure HandleEvent(var Event: TEvent); virtual;
    procedure InitBuffer; virtual;
    function LoadFile: Boolean;
    function Save: Boolean;
    function SaveAs: Boolean;
    function SaveFile: Boolean;
    function SetBufSize(NewSize: Word): Boolean; virtual;
    procedure Store(var S: TStream);
    procedure UpdateCommands; virtual;
    function Valid(Command: Word): Boolean; virtual;
  end;

```

```

type
  PEditWindow = ^TEditWindow;
  TEditWindow = object(TWindow)
    Editor: PFileEditor;
    constructor Init(var Bounds: TRect;
      FileName: FNameStr; ANumber: Integer);
    constructor Load(var S: TStream);
    procedure Close; virtual;
    function GetTitle(MaxSize: Integer): TTitleStr; virtual;
    procedure HandleEvent(var Event: TEvent); virtual;
    procedure Store(var S: TStream);
  end;

function DefEditorDialog(Dialog: Integer; Info: Pointer): Word;

const
  WordChars: set of Char = ['0'..'9', 'A'..'Z', '-', 'a'..'z'];
  EditorDialog: TEditorDialog = DefEditorDialog;
  EditorFlags: Word = efBackupFiles + efPromptOnReplace;
  FindStr: String[80] = '';
  ReplaceStr: String[80] = '';
  Clipboard: PEditor = nil;

type
  TFindDialogRec = record
    Find: String[80];
    Options: Word;
  end;

type
  TReplaceDialogRec = record
    Find: String[80];
    Replace: String[80];
    Options: Word;
  end;

const
  REditor: TStreamRec = (
    ObjType: 70;
    VmtLink: ofs(TypeOf(TEditor)^);
    Load: @TEditor.Load;
    Store: @TEditor.Store
  );
  RMemo: TStreamRec = (
    ObjType: 71;
    VmtLink: ofs(TypeOf(TMemo)^);
    Load: @TRemo.Load;
    Store: @TRemo.Store
  );
  RFileEditor: TStreamRec = (
    ObjType: 72;
    VmtLink: ofs(TypeOf(TFileEditor)^);
    Load: @RFileEditor.Load;
    Store: @RFileEditor.Store
  );
  RIndicator: TStreamRec = (
    ObjType: 73;
    VmtLink: ofs(TypeOf(TIndicator)^);
    Load: @RIndicator.Load;
    Store: @RIndicator.Store
  );

```

```

);
REditWindow: TStreamRec = (
  ObjType: 74;
  VmtLink: ofs(TypeOf(TEditWindow)^);
  Load: @TEditWindow.Load;
  Store: @TEditWindow.Store
);

procedure RegisterEditors;

implementation

uses Memory, Buffers, Dos;

const
  ufUpdate = $01;
  ufLine = $02;
  ufView = $04;

const
  smExtend = $01;
  smDouble = $02;

const
  sfSearchFailed = $FFFF;

const
  FirstKeys: array[0..37 * 2] of Word = (37,
    Ord(^A), cmWordLeft, Ord(^C), cmPageDown,
    Ord(^D), cmCharRight, Ord(^E), cmLineUp,
    Ord(^F), cmWordRight, Ord(^G), cmDelChar,
    Ord(^H), cmBackSpace, Ord(^K), $FF02,
    Ord(^L), cmSearchAgain, Ord(^M), cmNewLine,
    Ord(^O), cmIndentMode, Ord(^Q), $FF01,
    Ord(^R), cmPageUp, Ord(^S), cmCharLeft,
    Ord(^T), cmDelWord, Ord(^U), cmUndo,
    Ord(^V), cmInsMode, Ord(^X), cmLineDown,
    Ord(^Y), cmDelline, kbLeft, cmCharLeft,
    kbRight, cmCharRight, kbCtrlLeft, cmWordLeft,
    kbCtrlRight, cmWordRight, kbHome, cmLineStart,
    kbEnd, cmLineEnd, kbUp, cmLineUp,
    kbDown, cmLineDown, kbPgUp, cmPageUp,
    kbPgDn, cmPageDown, kbCtrlPgUp, cmTextStart,
    kbCtrlPgDn, cmTextEnd, kbIns, cmInsMode,
    kbDel, cmDelChar, kbShiftIns, cmPaste,
    kbShiftDel, cmCut, kbCtrlIns, cmCopy,
    kbCtrlDel, cmClear);
  QuickKeys: array[0..8 * 2] of Word = (8,
    Ord(^A'), cmReplace, Ord(^C'), cmTextEnd,
    Ord(^D'), cmLineEnd, Ord(^F'), cmFind,
    Ord(^H'), cmDelStart, Ord(^R'), cmTextStart,
    Ord(^S'), cmLineStart, Ord(^Y'), cmDelEnd);
  BlockKeys: array[0..5 * 2] of Word = (5,
    Ord(^B'), cmStartSelect, Ord(^C'), cmPaste,
    Ord(^H'), cmHideSelect, Ord(^K'), cmCopy,
    Ord(^Y'), cmCut);
  KeyMap: array[0..2] of Pointer = (@FirstKeys, @QuickKeys, @BlockKeys);

function DefEditorDialog(Dialog: Integer; Info: Pointer): Word;
begin

```

```

    DefEditorDialog := cmCancel;
end;

function Min(X, Y: Integer): Integer; assembler;
asm
    MOV     AX,X
    CMP     AX,Y
    JLE     @@1
    MOV     AX,Y
@@1:
end;

function Max(X, Y: Integer): Integer; assembler;
asm
    MOV     AX,X
    CMP     AX,Y
    JGE     @@1
    MOV     AX,Y
@@1:
end;

function MinWord(X, Y: Word): Word; assembler;
asm
    MOV     AX,X
    CMP     AX,Y
    JBE     @@1
    MOV     AX,Y
@@1:
end;

function MaxWord(X, Y: Word): Word; assembler;
asm
    MOV     AX,X
    CMP     AX,Y
    JAE     @@1
    MOV     AX,Y
@@1:
end;

function CountLines(var Buf; Count: Word): Integer; assembler;
asm
    LES     DI,Buf
    MOV     CX,Count
    XOR     DX,DX
    MOV     AL,0DH
@@1:
    JCXZ    @@2
    REPNE   SCASB
    JNE     @@2
    INC     DX
    JMP     @@1
@@2:
end;

function ScanKeyMap(KeyMap: Pointer; KeyCode: Word): Word; assembler;
asm
    PUSH    DS
    LDS     SI,KeyMap
    MOV     DX,KeyCode
    CLD

```

```

    LODSW
    MOV     CX,AX
@@1:
    LODSW
    MOV     BX,AX
    LODSW
    OR      BL,BL
    JE      @@2
    CMP     BL,DL
    JNE     @@3
    OR      BH,BH
    JE      @@4
@@2:
    CMP     BH,DH
    JE      @@4
@@3:
    LOOP    @@1
    XOR     AX,AX
@@4:
    POP     DS

function Scan(var Block; Size: Word; Str: String): Word; assembler;
asm
    PUSH    DS
    LES     DI,Block
    LDS     SI,Str
    MOV     CX,Size
    JCXZ     @@3
    CLD
    LODSB
    CMP     AL,1
    JB      @@5
    JA      @@1
    LODSB
    REPNE   SCASB
    JNE     @@3
    JMP     @@5
@@1:
    XOR     AH,AH
    MOV     BX,AX
    DEC     BX
    MOV     DX,CX
    SUB     DX,AX
    JB      @@3
    LODSB
    INC     DX
    INC     DX
@@2:
    DEC     DX
    MOV     CX,DX
    REPNE   SCASB
    JNE     @@3
    MOV     DX,CX
    MOV     CX,BX
    REP     CMPSB
    JE      @@4
    SUB     CX,BX
    ADD     SI,CX
    ADD     DI,CX
    INC     DI
    OR      DX,DX
    JNE     @@2
@@3:
    XOR     AX,AX
    JMP     @@6
@@4:
    SUB     DI,BX

```

```

@5:  MOV     AX,DI
      SUB     AX,WORD PTR Block
@6:  DEC     AX
      POP     DS
end;

function IScan(var Block; Size: Word; Str: String): Word; assembler;
var
  S: String;
asm
  PUSH     DS
  MOV     AX,SS
  MOV     ES,AX
  LEA     DI,S
  LDS     SI,Str
  XOR     AH,AH
  LODSB
  STOSB
  MOV     CX,AX
  MOV     BX,AX
  JCXZ    @9
@1:  LODSB
  CMP     AL,'a'
  JB      @2
  CMP     AL,'z'
  JA      @2
  SUB     AL,20H
@2:  STOSB
  LOOP    @1
  SUB     DI,BX
  LDS     SI,Block
  MOV     CX,Size
  JCXZ    @8
  CLD
  SUB     CX,BX
  JB      @8
  INC     CX
@4:  MOV     AH,ES:[DI]
  AND     AH,$DF
@5:  LODSB
  AND     AL,$DF
  CMP     AL,AH
  LOOPNE  @5
  JNE     @8
  DEC     SI
  MOV     DX,CX
  MOV     CX,BX
@6:  REPE    CMPSB
  JE      @10
  MOV     AL,DS:[SI-1]
  CMP     AL,'a'
  JB      @7
  CMP     AL,'z'
  JA      @7
  SUB     AL,20H
@7:  CMP     AL,ES:[DI-1]
  JE      @6
  SUB     CX,BX
  ADD     SI,CX
  ADD     DI,CX

```

```

      INC     SI
      MOV     CX,DX
      JNE     @4
@8:  XOR     AX,AX
      JMP     @11
@9:  MOV     AX,1
      JMP     @11
@10: SUB     SI,BX
      MOV     AX,SI
      SUB     AX,WORD PTR Block
      INC     AX
@11: DEC     AX
      POP     DS
end;

( TIndicator )

constructor TIndicator.Init(var Bounds: TRect);
var
  R: TRect;
begin
  TView.Init(Bounds);
  GrowMode := gfGrowLoY + gfGrowHiY;
end;

procedure TIndicator.Draw;
var
  Color: Byte;
  Frame: Char;
  L: array[0..1] of Longint;
  S: String[15];
  B: TDrawBuffer;
begin
  if State and sfDragging = 0 then
  begin
    Color := GetColor(1);
    Frame := #205;
  end else
  begin
    Color := GetColor(2);
    Frame := #196;
  end;
  MoveChar(B, Frame, Color, Size.X);
  if Modified then WordRec(B[0]).Lo := 15;
  L[0] := Location.Y + 1;
  L[1] := Location.X + 1;
  FormatStr(S, '%d:%d', L);
  MoveStr(B[8 - Pos(':', S)], S, Color);
  WriteBuf(0, 0, Size.X, 1, B);
end;

function TIndicator.GetPalette: PPalette;
const
  P: string[Length(CIndicator)] = CIndicator;
begin
  GetPalette := @P;
end;

procedure TIndicator.SetState(AState: Word; Enable: Boolean);
begin

```

```

    TView.SetState(AState, Enable);
    if AState = sfDragging then DrawView;
end;

procedure TIndicator.SetValue(ALocation: TPoint; AModified: Boolean);
begin
    if (Longint(Location) <> Longint(ALocation)) or
        (Modified <> AModified) then
    begin
        Location := ALocation;
        Modified := AModified;
        DrawView;
    end;
end;

{ TEditor }

constructor TEditor.Init(var Bounds: TRect;
    AHScrollBar, AVScrollBar: PScrollBar;
    AIndicator: PIndicator; ABufSize: Word);
begin
    TView.Init(Bounds);
    GrowMode := gfGrowHiX + gfGrowHiY;
    Options := Options or ofSelectable;
    EventMask := evMouseDown + evKeyDown + evCommand + evBroadcast;
    ShowCursor;
    HScrollBar := AHScrollBar;
    VScrollBar := AVScrollBar;
    Indicator := AIndicator;
    BufSize := ABufSize;
    CanUndo := True;
    InitBuffer;
    if Buffer <> nil then IsValid := True else
    begin
        EditorDialog(edOutOfMemory, nil);
        BufSize := 0;
    end;
    SetBufLen(0);
end;

constructor TEditor.Load(var S: TStream);
begin
    TView.Load(S);
    GetPeerViewPtr(S, HScrollBar);
    GetPeerViewPtr(S, VScrollBar);
    GetPeerViewPtr(S, Indicator);
    S.Read(BufSize, SizeOf(Word));
    S.Read(CanUndo, SizeOf(Boolean));
    InitBuffer;
    if Buffer <> nil then IsValid := True else
    begin
        EditorDialog(edOutOfMemory, nil);
        BufSize := 0;
    end;
    Lock;
    SetBufLen(0);
end;

destructor TEditor.Done;
begin

```

```

    DoneBuffer;
    TView.Done;
end;

function TEditor.BufChar(P: Word): Char; assembler;
asm
    LES     DI,Self
    MOV     BX,P
    CMP     BX,ES:[DI].TEditor.CurPtr
    JB      @a1
    ADD     BX,ES:[DI].TEditor.GapLen
@a1:      LES     DI,ES:[DI].TEditor.Buffer
    MOV     AL,ES:[DI+BX]
end;

function TEditor.BufPtr(P: Word): Word; assembler;
asm
    LES     DI,Self
    MOV     AX,P
    CMP     AX,ES:[DI].TEditor.CurPtr
    JB      @a1
    ADD     AX,ES:[DI].TEditor.GapLen
@a1:
end;

procedure TEditor.ChangeBounds(var Bounds: TRect);
begin
    SetBounds(Bounds);
    Delta.X := Max(0, Min(Delta.X, Limit.X - Size.X));
    Delta.Y := Max(0, Min(Delta.Y, Limit.Y - Size.Y));
    Update(uvView);
end;

function TEditor.CharPos(P, Target: Word): Integer;
var
    Pos: Integer;
begin
    Pos := 0;
    while P < Target do
    begin
        if BufChar(P) = #9 then Pos := Pos or 7;
        Inc(Pos);
        Inc(P);
    end;
    CharPos := Pos;
end;

function TEditor.CharPtr(P: Word; Target: Integer): Word;
var
    Pos: Integer;
begin
    Pos := 0;
    while (Pos < Target) and (P < BufLen) and (BufChar(P) <> #13) do
    begin
        if BufChar(P) = #9 then Pos := Pos or 7;
        Inc(Pos);
        Inc(P);
    end;
    if Pos > Target then Dec(P);
    CharPtr := P;
end;

```



```

end;

function TEditor.ClipCopy: Boolean;
begin
  ClipCopy := False;
  if (Clipboard <> nil) and (Clipboard <> @Self) then
  begin
    ClipCopy := Clipboard^.InsertFrom(@Self);
    Selecting := False;
    Update(ufUpdate);
  end;
end;

procedure TEditor.ClipCut;
begin
  if ClipCopy then DeleteSelect;
end;

procedure TEditor.ClipPaste;
begin
  if (Clipboard <> nil) and (Clipboard <> @Self) then InsertFrom(Clipboard);
end;

procedure TEditor.ConvertEvent(var Event: TEvent);
var
  Key: Word;
begin
  if Event.What = evKeyDown then
  begin
    Key := Event.KeyCode;
    if KeyState <> 0 then
    begin
      if (Lo(Key) >= $01) and (Lo(Key) <= $1A) then Inc(Key, $40);
      if (Lo(Key) >= $61) and (Lo(Key) <= $7A) then Dec(Key, $20);
    end;
    Key := ScanKeyMap(KeyMap[KeyState], Key);
    KeyState := 0;
    if Key <> 0 then
    begin
      if Hi(Key) = $FF then
      begin
        KeyState := Lo(Key);
        ClearEvent(Event);
      end else
      begin
        Event.What := evCommand;
        Event.Command := Key;
      end;
    end;
  end;
end;

function TEditor.CursorVisible: Boolean;
begin
  CursorVisible := (CurPos.Y >= Delta.Y) and (CurPos.Y < Delta.Y + Size.Y);
end;

procedure TEditor.DeleteRange(StartPtr, EndPtr: Word; DelSelect: Boolean);
begin
  if HasSelection and DelSelect then DeleteSelect else
  begin
    SetSelect(CurPtr, EndPtr, True);
  end;
end;

```

```

DeleteSelect;
SetSelect(StartPtr, CurPtr, False);
DeleteSelect;
end;

procedure TEditor.DeleteSelect;
begin
  InsertText(nil, 0, False);
end;

procedure TEditor.DoneBuffer;
begin
  if Buffer <> nil then FreeMem(Buffer, BufSize);
end;

procedure TEditor.DoSearchReplace;
var
  I: Word;
  C: TPoint;
begin
  repeat
    I := cmCancel;
    if not Search(FindStr, EditorFlags) then
    begin
      if EditorFlags and (efReplaceAll + efDoReplace) <>
        (efReplaceAll + efDoReplace) then
        EditorDialog(edSearchFailed, nil)
      end
    else if EditorFlags and efDoReplace <> 0 then
    begin
      I := cmYes;
      if EditorFlags and efPromptOnReplace <> 0 then
      begin
        MakeGlobal(Cursor, C);
        I := EditorDialog(edReplacePrompt, Pointer(C));
      end;
      if I = cmYes then
      begin
        Lock;
        InsertText(@ReplaceStr[1], Length(ReplaceStr), False);
        TrackCursor(False);
        Unlock;
      end;
    end;
  until (I = cmCancel) or (EditorFlags and efReplaceAll = 0);
end;

procedure TEditor.DoUpdate;
begin
  if UpdateFlags <> 0 then
  begin
    SetCursor(CurPos.X - Delta.X, CurPos.Y - Delta.Y);
    if UpdateFlags and ufView <> 0 then DrawView else
    if UpdateFlags and ufLine <> 0 then
      DrawLines(CurPos.Y - Delta.Y, 1, LineStart(CurPtr));
    if HScrollBar <> nil then
      HScrollBar^.SetParams(Delta.X, 0, Limit.X - Size.X, Size.X div 2, 1);
    if VScrollBar <> nil then
      VScrollBar^.SetParams(Delta.Y, 0, Limit.Y - Size.Y, Size.Y - 1, 1);
  end;
end;

```

```

    if Indicator <> nil then Indicator^.SetValue(CurPos, Modified);
    if State and sfActive <> 0 then UpdateCommands;
    UpdateFlags := 0;
end;
end;

```

```

procedure TEditor.Draw;

```

```

begin
    if DrawLine <> Delta.Y then
    begin
        DrawPtr := LineMove(DrawPtr, Delta.Y - DrawLine);
        DrawLine := Delta.Y;
    end;
    DrawLines(0, Size.Y, DrawPtr);
end;

```

```

procedure TEditor.DrawLines(Y, Count: Integer; LinePtr: Word);

```

```

var
    Color: Word;
    B: array[0..MaxLineLength - 1] of Word;
begin
    Color := GetColor($0201);
    while Count > 0 do
    begin
        FormatLine(B, LinePtr, Delta.X + Size.X, Color);
        WriteBuf(0, Y, Size.X, 1, B[Delta.X]);
        LinePtr := NextLine(LinePtr);
        Inc(Y);
        Dec(Count);
    end;
end;

```

```

procedure TEditor.Find;

```

```

var
    FindRec: TFindDialogRec;
begin
    with FindRec do
    begin
        Find := FindStr;
        Options := EditorFlags;
        if EditorDialog(edFind, @FindRec) <> cmCancel then
        begin
            FindStr := Find;
            EditorFlags := Options and not efDoReplace;
            DoSearchReplace;
        end;
    end;
end;

```

```

procedure TEditor.FormatLine(var DrawBuf; LinePtr: Word;
    Width: Integer; Colors: Word); assembler;

```

```

asm
    PUSH    DS
    LDS     BX,Self
    LES     DI,DrawBuf
    MOV     SI,LinePtr
    XOR     DX,DX
    CLD
    MOV     AH,Colors.Byte[0]
    MOV     CX,DS:[BX].TEditor.SelStart

```

```

    CALL    @a10
    MOV     AH,Colors.Byte[1]
    MOV     CX,DS:[BX].TEditor.CurPtr
    CALL    @a10
    ADD     SI,DS:[BX].TEditor.GapLen
    MOV     CX,DS:[BX].TEditor.SelEnd
    ADD     CX,DS:[BX].TEditor.GapLen
    CALL    @a10
    MOV     AH,Colors.Byte[0]
    MOV     CX,DS:[BX].TEditor.BufSize
    CALL    @a10
    JMP     @a31
@a10:      SUB     CX,SI
    JA      @a11
    RETN
@a11:      LDS     BX,DS:[BX].TEditor.Buffer
    ADD     SI,BX
    MOV     BX,Width
@a12:      LODSB
    CMP     AL,' '
    JB      @a20
@a13:      STOSW
    INC     DX
@a14:      CMP     DX,BX
    JAE     @a30
    LOOP    @a12
    LDS     BX,Self
    SUB     SI,DS:[BX].TEditor.Buffer.Word[0]
    RETN
@a20:      CMP     AL,0DH
    JE      @a30
    CMP     AL,09H
    JNE     @a13
    MOV     AL,' '
@a21:      STOSW
    INC     DX
    TEST    DL,7
    JNE     @a21
    JMP     @a14
@a30:      POP     CX
@a31:      MOV     AL,' '
    MOV     CX,Width
    SUB     CX,DX
    JBE     @a32
    REP     STOSW
@a32:      POP     DS
end;

```

```

function TEditor.GetMousePtr(Mouse: TPoint): Word;
begin

```

```

    MakeLocal(Mouse, Mouse);
    Mouse.X := Max(0, Min(Mouse.X, Size.X - 1));
    Mouse.Y := Max(0, Min(Mouse.Y, Size.Y - 1));
    GetMousePtr := CharPtr(LineMove(DrawPtr, Mouse.Y + Delta.Y - DrawLine),
        Mouse.X + Delta.X);
end;

```

```

function TEditor.GetPalette: PPalette;

```

```

const
    P: String[Length(CEditor)] = CEditor;

```

```

begin
  GetPalette := @P;
end;

procedure TEditor.HandleEvent(var Event: TEvent);
var
  ShiftState: Byte absolute $40:$17;
  CenterCursor: Boolean;
  SelectMode: Byte;
  I: Integer;
  NewPtr: Word;
  D, Mouse: TPoint;
begin
  procedure CheckScrollBar(P: PScrollBar; var D: Integer);
  begin
    if (Event.InfoPtr = P) and (P.Value <> D) then
    begin
      D := P.Value;
      Update(uvView);
    end;
  end;

  TView.HandleEvent(Event);
  ConvertEvent(Event);
  CenterCursor := not CursorVisible;
  SelectMode := 0;
  if Selecting or (ShiftState and $03 <> 0) then SelectMode := smExtend;
  case Event.What of
    evMouseDown:
      begin
        if Event.Double then SelectMode := SelectMode or smDouble;
        repeat
          Lock;
          if Event.What = evMouseAuto then
            begin
              MakeLocal(Event.Where, Mouse);
              D := Delta;
              if Mouse.X < 0 then Dec(D.X);
              if Mouse.X >= Size.X then Inc(D.X);
              if Mouse.Y < 0 then Dec(D.Y);
              if Mouse.Y >= Size.Y then Inc(D.Y);
              ScrollTo(D.X, D.Y);
            end;
          SetCurPtr(GetMousePtr(Event.Where), SelectMode);
          SelectMode := SelectMode or smExtend;
          Unlock;
        until not MouseEvent(Event, evMouseMove + evMouseAuto);
      end;
    evKeyDown:
      case Event.CharCode of
        #9, #32..#255:
          begin
            Lock;
            if Overwrite and not HasSelection then
              if CurPtr <> LineEnd(CurPtr) then SelEnd := NextChar(CurPtr);
            InsertText(@Event.CharCode, 1, False);
            TrackCursor(CenterCursor);
            Unlock;
          end;
      end;
  end;

```

```

  else
    Exit;
  end;
end;
evCommand:
  case Event.Command of
    cmFind: Find;
    cmReplace: Replace;
    cmSearchAgain: DoSearchReplace;
  else
    begin
      Lock;
      case Event.Command of
        cmCut: ClipCut;
        cmCopy: ClipCopy;
        cmPaste: ClipPaste;
        cmUndo: Undo;
        cmClear: DeleteSelect;
        cmCharLeft: SetCurPtr(PrevChar(CurPtr), SelectMode);
        cmCharRight: SetCurPtr(NextChar(CurPtr), SelectMode);
        cmWordLeft: SetCurPtr(PrevWord(CurPtr), SelectMode);
        cmWordRight: SetCurPtr(NextWord(CurPtr), SelectMode);
        cmLineStart: SetCurPtr(LineStart(CurPtr), SelectMode);
        cmLineEnd: SetCurPtr(LineEnd(CurPtr), SelectMode);
        cmLineUp: SetCurPtr(LineMove(CurPtr, -1), SelectMode);
        cmLineDown: SetCurPtr(LineMove(CurPtr, 1), SelectMode);
        cmPageUp: SetCurPtr(LineMove(CurPtr, -(Size.Y - 1)), SelectMode);
        cmPageDown: SetCurPtr(LineMove(CurPtr, Size.Y - 1), SelectMode);
        cmTextStart: SetCurPtr(0, SelectMode);
        cmTextEnd: SetCurPtr(BufLen, SelectMode);
        cmNewLine: NewLine;
        cmBackSpace: DeleteRange(PrevChar(CurPtr), CurPtr, True);
        cmDelChar: DeleteRange(CurPtr, NextChar(CurPtr), True);
        cmDelWord: DeleteRange(CurPtr, NextWord(CurPtr), False);
        cmDelStart: DeleteRange(LineStart(CurPtr), CurPtr, False);
        cmDelEnd: DeleteRange(CurPtr, LineEnd(CurPtr), False);
        cmDelLine: DeleteRange(LineStart(CurPtr), NextLine(CurPtr), False);
        cmInsMode: ToggleInsMode;
        cmStartSelect: StartSelect;
        cmHideSelect: HideSelect;
        cmIndentMode: AutoIndent := not AutoIndent;
      end;
    end;
    Unlock;
    Exit;
  end;
  TrackCursor(CenterCursor);
  Unlock;
end;
evBroadcast:
  case Event.Command of
    cmScrollBarChanged:
      begin
        CheckScrollBar(HScrollBar, Delta.X);
        CheckScrollBar(VScrollBar, Delta.Y);
      end;
  else
    Exit;
  end;
end;
ClearEvent(Event);

```

```

end;

function TEditor.HasSelection: Boolean;
begin
    HasSelection := SelStart <> SelEnd;
end;

procedure TEditor.HideSelect;
begin
    Selecting := False;
    SetSelect(CurPtr, CurPtr, False);
end;

procedure TEditor.InitBuffer;
begin
    Buffer := MemAlloc(BufSize);
end;

function TEditor.InsertBuffer(var P: PEditBuffer; Offset, Length: Word;
    AllowUndo, SelectText: Boolean): Boolean;
var
    SelLen, DelLen, SelLines, Lines: Word;
    NewSize: Longint;
begin
    InsertBuffer := True;
    Selecting := False;
    SelLen := SelEnd - SelStart;
    if (SelLen = 0) and (Length = 0) then Exit;
    DelLen := 0;
    if AllowUndo then
        if CurPtr = SelStart then DelLen := SelLen else
            if SelLen > InsCount then DelLen := SelLen - InsCount;
    NewSize := Longint(BufLen + DelCount - SelLen + DelLen) + Length;
    if NewSize > BufLen + DelCount then
        if (NewSize > $FFF0) or not SetBufSize(NewSize) then
            begin
                EditorDialog(edOutOfMemory, nil);
                InsertBuffer := False;
                Exit;
            end;
    SelLines := CountLines(Buffer^[BufPtr(SelStart)], SelLen);
    if CurPtr = SelEnd then
        begin
            if AllowUndo then
                begin
                    if DelLen > 0 then Move(Buffer^[SelStart],
                        Buffer^[CurPtr + GapLen - DelCount - DelLen], DelLen);
                    Dec(InsCount, SelLen - DelLen);
                end;
            CurPtr := SelStart;
            Dec(CurPos.Y, SelLines);
        end;
    if Delta.Y > CurPos.Y then
        begin
            Dec(Delta.Y, SelLines);
            if Delta.Y < CurPos.Y then Delta.Y := CurPos.Y;
        end;
    if Length > 0 then Move(P^[Offset], Buffer^[CurPtr], Length);
    Lines := CountLines(Buffer^[CurPtr], Length);
    Inc(CurPtr, Length);

```

```

    Inc(CurPos.Y, Lines);
    DrawLine := CurPos.Y;
    DrawPtr := LineStart(CurPtr);
    CurPos.X := CharPos(DrawPtr, CurPtr);
    if not SelectText then SelStart := CurPtr;
    SelEnd := CurPtr;
    Inc(BufLen, Length - SelLen);
    Dec(GapLen, Length - SelLen);
    if AllowUndo then
        begin
            Inc(DelCount, DelLen);
            Inc(InsCount, Length);
        end;
    Inc(Limit.Y, Lines - SelLines);
    Delta.Y := Max(0, Min(Delta.Y, Limit.Y - Size.Y));
    if not IsClipboard then Modified := True;
    SetBufSize(BufLen + DelCount);
    if (SelLines = 0) and (Lines = 0) then Update(ufLine) else Update(ufView);
end;

function TEditor.InsertFrom(Editor: PEditor): Boolean;
begin
    InsertFrom := InsertBuffer(Editor^.Buffer,
        Editor^.BufPtr(Editor^.SelStart),
        Editor^.SelEnd - Editor^.SelStart, CanUndo, IsClipboard);
end;

function TEditor.InsertText(Text: Pointer; Length: Word;
    SelectText: Boolean): Boolean;
begin
    InsertText := InsertBuffer(PEditBuffer(Text),
        0, Length, CanUndo, SelectText);
end;

function TEditor.IsClipboard: Boolean;
begin
    IsClipboard := Clipboard = @Self;
end;

function TEditor.LineEnd(P: Word): Word; assembler;
asm
    PUSH    DS
    LDS     SI, Self
    LES     BX, DS:[SI].TEditor.Buffer
    MOV     DI, P
    MOV     AL, 0DH
    CLD
    MOV     CX, DS:[SI].TEditor.CurPtr
    SUB     CX, DI
    JBE     @@1
    ADD     DI, BX
    REPNE   SCASB
    JE      @@2
    MOV     DI, DS:[SI].TEditor.CurPtr
@@1:    MOV     CX, DS:[SI].TEditor.BufLen
    SUB     CX, DI
    JCXZ    @@4
    ADD     BX, DS:[SI].TEditor.GapLen
    ADD     DI, BX
    REPNE   SCASB

```

```

    JNE @a3
@a2: DEC DI
@a3: SUB DI,BX
@a4: MOV AX,DI
    POP DS
end;

function TEditor.LineMove(P: Word; Count: Integer): Word;
var
    Pos: Integer;
    I: Word;
begin
    I := P;
    P := LineStart(P);
    Pos := CharPos(P, I);
    while Count <> 0 do
    begin
        I := P;
        if Count < 0 then
        begin
            P := PrevLine(P);
            Inc(Count);
        end else
        begin
            P := NextLine(P);
            Dec(Count);
        end;
    end;
    if P <> I then P := CharPtr(P, Pos);
    LineMove := P;
end;

function TEditor.LineStart(P: Word): Word; assembler;
asm
    PUSH DS
    LDS SI,Self
    LES BX,DS:[SI].TEditor.Buffer
    MOV DI,P
    MOV AL,0DH
    STD
    MOV CX,DI
    SUB CX,DS:[SI].TEditor.CurPtr
    JBE @a1
    ADD BX,DS:[SI].TEditor.GapLen
    ADD DI,BX
    DEC DI
    REPNE SCASB
    JE @a2
    SUB BX,DS:[SI].TEditor.GapLen
    MOV DI,DS:[SI].TEditor.CurPtr
@a1: MOV CX,DI
    JCXZ @a4
    ADD DI,BX
    DEC DI
    REPNE SCASB
    JNE @a3
@a2: INC DI
    INC DI
    SUB DI,BX
    CMP DI,DS:[SI].TEditor.CurPtr

```

```

    JE @a4
    CMP DI,DS:[SI].TEditor.BufLen
    JE @a4
    CMP ES:[BX+DI].Byte,0AH
    JNE @a4
    INC DI
    JMP @a4
@a3: XOR DI,DI
@a4: MOV AX,DI
    POP DS
end;

procedure TEditor.Lock;
begin
    Inc(LockCount);
end;

procedure TEditor.NewLine;
const
    CrLf: array[1..2] of Char = #13#10;
var
    I, P: Word;
begin
    P := LineStart(CurPtr);
    I := P;
    while (I < CurPtr) and ((Buffer^[I] = ' ') or (Buffer^[I] = #9)) do Inc(I);
    InsertText(@CrLf, 2, False);
    if AutoIndent then InsertText(@Buffer^[P], I - P, False);
end;

function TEditor.NextChar(P: Word): Word; assembler;
asm
    PUSH DS
    LDS SI,Self
    MOV DI,P
    CMP DI,DS:[SI].TEditor.BufLen
    JE @a2
    INC DI
    CMP DI,DS:[SI].TEditor.BufLen
    JE @a2
    LES BX,DS:[SI].TEditor.Buffer
    CMP DI,DS:[SI].TEditor.CurPtr
    JB @a1
    ADD BX,DS:[SI].TEditor.GapLen
@a1: CMP ES:[BX+DI-1].Word,0A0DH
    JNE @a2
    INC DI
@a2: MOV AX,DI
    POP DS
end;

function TEditor.NextLine(P: Word): Word;
begin
    NextLine := NextChar(LineEnd(P));
end;

function TEditor.NextWord(P: Word): Word;
begin
    while (P < BufLen) and (BufChar(P) in WordChars) do
        P := NextChar(P);

```

```

while (P < BufLen) and not (BufChar(P) in WordChars) do
  P := NextChar(P);
NextWord := P;
end;

```

```

function TEditor.PrevChar(P: Word): Word; assembler;
asm

```

```

    PUSH    DS
    LDS     SI,Self
    MOV     DI,P
    OR      DI,DI
    JE      @a2
    DEC     DI
    JE      @a2
    LES     BX,DS:[SI].TEditor.Buffer
    CMP     DI,DS:[SI].TEditor.CurPtr
    JB      @a1
    ADD     BX,DS:[SI].TEditor.GapLen
@a1:  CMP     ES:[BX+DI-1].Word,0A0DH
    JNE     @a2
    DEC     DI
@a2:  MOV     AX,DI
    POP     DS
end;

```

```

function TEditor.PrevLine(P: Word): Word;
begin
  PrevLine := LineStart(PrevChar(P));
end;

```

```

function TEditor.PrevWord(P: Word): Word;
begin
  while (P > 0) and not (BufChar(PrevChar(P)) in WordChars) do
    P := PrevChar(P);
  while (P > 0) and (BufChar(PrevChar(P)) in WordChars) do
    P := PrevChar(P);
  PrevWord := P;
end;

```

```

procedure TEditor.Replace;
var
  ReplaceRec: TReplaceDialogRec;
begin
  with ReplaceRec do
  begin
    Find := FindStr;
    Replace := ReplaceStr;
    Options := EditorFlags;
    if EditorDialog(edReplace, @ReplaceRec) <> cmCancel then
    begin
      FindStr := Find;
      ReplaceStr := Replace;
      EditorFlags := Options or efDoReplace;
      DoSearchReplace;
    end;
  end;
end;

```

```

procedure TEditor.ScrollTo(X, Y: Integer);
begin

```

```

  X := Max(0, Min(X, Limit.X - Size.X));
  Y := Max(0, Min(Y, Limit.Y - Size.Y));
  if (X <> Delta.X) or (Y <> Delta.Y) then
  begin
    Delta.X := X;
    Delta.Y := Y;
    Update(ufView);
  end;
end;

```

```

function TEditor.Search(FindStr: String; Opts: Word): Boolean;
var
  I, Pos: Word;
begin
  Search := False;
  Pos := CurPtr;
  repeat
    if Opts and efCaseSensitive <> 0 then
      I := Scan(Buffer^[BufPtr(Pos)], BufLen - Pos, FindStr)
    else I := IScan(Buffer^[BufPtr(Pos)], BufLen - Pos, FindStr);
    if (I <> sfSearchFailed) then
    begin
      Inc(I, Pos);
      if (Opts and efWholeWordsOnly = 0) or
        not (((I <> 0) and (BufChar(I - 1) in WordChars)) or
          ((I + Length(FindStr) <> BufLen) and
            (BufChar(I + Length(FindStr)) in WordChars))) then
      begin
        Lock;
        SetSelect(I, I + Length(FindStr), False);
        TrackCursor(not CursorVisible);
        Unlock;
        Search := True;
        Exit;
      end else Pos := I + 1;
    end;
  until I = sfSearchFailed;
end;

```

```

procedure TEditor.SetBufLen(Length: Word);
begin
  BufLen := Length;
  GapLen := BufSize - Length;
  SelStart := 0;
  SelEnd := 0;
  CurPtr := 0;
  Longint(CurPos) := 0;
  Longint(Delta) := 0;
  Limit.X := MaxLineLength;
  Limit.Y := CountLines(Buffer^[GapLen], BufLen) + 1;
  DrawLine := 0;
  DrawPtr := 0;
  DelCount := 0;
  InsCount := 0;
  Modified := False;
  Update(ufView);
end;

```

```

function TEditor.SetBufSize(NewSize: Word): Boolean;
begin

```

```

SetBufSize := NewSize <= BufSize;
end;

procedure TEditor.SetCmdState(Command: Word; Enable: Boolean);
var
  S: TCommandSet;
begin
  S := [Command];
  if Enable and (State and sfActive <> 0) then
    EnableCommands(S) else DisableCommands(S);
end;

procedure TEditor.SetCurPtr(P: Word; SelectMode: Byte);
var
  Anchor: Word;
begin
  if SelectMode and smExtend = 0 then Anchor := P else
    if CurPtr = SelStart then Anchor := SelEnd else Anchor := SelStart;
  if P < Anchor then
    begin
      if SelectMode and smDouble <> 0 then
        begin
          P := PrevLine(NextLine(P));
          Anchor := NextLine(PrevLine(Anchor));
        end;
      SetSelect(P, Anchor, True);
    end else
    begin
      if SelectMode and smDouble <> 0 then
        begin
          P := NextLine(P);
          Anchor := PrevLine(NextLine(Anchor));
        end;
      SetSelect(Anchor, P, False);
    end;
end;

procedure TEditor.SetSelect(NewStart, NewEnd: Word; CurStart: Boolean);
var
  Flags: Byte;
  P, L: Word;
begin
  if CurStart then P := NewStart else P := NewEnd;
  Flags := ufUpdate;
  if (NewStart <> SelStart) or (NewEnd <> SelEnd) then
    if (NewStart <> NewEnd) or (SelStart <> SelEnd) then
      Flags := ufView;
  if P <> CurPtr then
    begin
      if P > CurPtr then
        begin
          L := P - CurPtr;
          Move(Buffer^[CurPtr + GapLen], Buffer^[CurPtr], L);
          Inc(CurPos.Y, CountLines(Buffer^[CurPtr], L));
          CurPtr := P;
        end else
        begin
          L := CurPtr - P;
          CurPtr := P;
          Dec(CurPos.Y, CountLines(Buffer^[CurPtr], L));
        end;
      end;
    end;
end;

```

```

Move(Buffer^[CurPtr], Buffer^[CurPtr + GapLen], L);
end;
DrawLine := CurPos.Y;
DrawPtr := LineStart(P);
CurPos.X := CharPos(DrawPtr, P);
DelCount := 0;
InsCount := 0;
SetBufSize(BufLen);
end;
SelStart := NewStart;
SelEnd := NewEnd;
Update(Flags);
end;

procedure TEditor.SetState(AState: Word; Enable: Boolean);
begin
  TView.SetState(AState, Enable);
  case AState of
    sfActive:
      begin
        if HScrollBar <> nil then HScrollBar^.SetState(sfVisible, Enable);
        if VScrollBar <> nil then VScrollBar^.SetState(sfVisible, Enable);
        if Indicator <> nil then Indicator^.SetState(sfVisible, Enable);
        UpdateCommands;
      end;
    sfExposed:
      if Enable then Unlock;
  end;
end;

procedure TEditor.StartSelect;
begin
  HideSelect;
  Selecting := True;
end;

procedure TEditor.Store(var S: TStream);
begin
  TView.Store(S);
  PutPeerViewPtr(S, HScrollBar);
  PutPeerViewPtr(S, VScrollBar);
  PutPeerViewPtr(S, Indicator);
  S.Write(BufSize, SizeOf(Word));
  S.Write(CanUndo, SizeOf(Boolean));
end;

procedure TEditor.ToggleInsMode;
begin
  Overwrite := not Overwrite;
  SetState(sfCursorIns, not GetState(sfCursorIns));
end;

procedure TEditor.TrackCursor(Center: Boolean);
begin
  if Center then
    ScrollTo(CurPos.X - Size.X + 1, CurPos.Y - Size.Y div 2) else
    ScrollTo(Max(CurPos.X - Size.X + 1, Min(Delta.X, CurPos.X)),
      Max(CurPos.Y - Size.Y + 1, Min(Delta.Y, CurPos.Y)));
end;

```

```

procedure TEditor.Undo;
var
  Length: Word;
begin
  if (DelCount <> 0) or (InsCount <> 0) then
  begin
    SelStart := CurPtr - InsCount;
    SelEnd := CurPtr;
    Length := DelCount;
    DelCount := 0;
    InsCount := 0;
    InsertBuffer(Buffer, CurPtr + GapLen - Length, Length, False, True);
  end;
end;

procedure TEditor.Unlock;
begin
  if LockCount > 0 then
  begin
    Dec(LockCount);
    if LockCount = 0 then DoUpdate;
  end;
end;

procedure TEditor.Update(AFlags: Byte);
begin
  UpdateFlags := UpdateFlags or AFlags;
  if LockCount = 0 then DoUpdate;
end;

procedure TEditor.UpdateCommands;
begin
  SetCmdState(cmUndo, (DelCount <> 0) or (InsCount <> 0));
  if not IsClipboard then
  begin
    SetCmdState(cmCut, HasSelection);
    SetCmdState(cmCopy, HasSelection);
    SetCmdState(cmPaste, (Clipboard <> nil) and (Clipboard^.HasSelection));
  end;
  SetCmdState(cmClear, HasSelection);
  SetCmdState(cmFind, True);
  SetCmdState(cmReplace, True);
  SetCmdState(cmSearchAgain, True);
end;

function TEditor.Valid(Command: Word): Boolean;
begin
  Valid := IsValid;
end;

{ TEditor }

constructor TEditor.Load(var S: TStream);
var
  Length: Word;
begin
  TEditor.Load(S);
  S.Read(Length, SizeOf(Word));
  if IsValid then
  begin

```

```

    S.Read(Buffer^[BufSize - Length], Length);
    SetBufLen(Length);
  end
  else S.Seek(S.GetPos + Length);
end;

function TEditor.DataSize: Word;
begin
  DataSize := BufSize + SizeOf(Word);
end;

procedure TEditor.GetData(var Rec);
var
  Data: TEditorData absolute Rec;
begin
  Data.Length := BufLen;
  Move(Buffer^, Data.Buffer, CurPtr);
  Move(Buffer^[CurPtr + GapLen], Data.Buffer[CurPtr], BufLen - CurPtr);
  FillChar(Data.Buffer[BufLen], BufSize - BufLen, 0);
end;

function TEditor.GetPalette: PPalette;
const
  P: String[Length(CMemo)] = CMemo;
begin
  GetPalette := @P;
end;

procedure TEditor.HandleEvent(var Event: TEvent);
begin
  if (Event.What <> evKeyDown) or (Event.KeyCode <> kbTab) then
    TEditor.HandleEvent(Event);
end;

procedure TEditor.SetData(var Rec);
var
  Data: TEditorData absolute Rec;
begin
  Move(Data.Buffer, Buffer^[BufSize - Data.Length], Data.Length);
  SetBufLen(Data.Length);
end;

procedure TEditor.Store(var S: TStream);
begin
  TEditor.Store(S);
  S.Write(BufLen, SizeOf(Word));
  S.Write(Buffer^, CurPtr);
  S.Write(Buffer^[CurPtr + GapLen], BufLen - CurPtr);
end;

{ TFileEditor }

constructor TFileEditor.Init(var Bounds: TRect;
  AHScrollBar, AVScrollBar: PScrollBar;
  AIndicator: PIndicator; AFileName: FNameStr);
begin
  TEditor.Init(Bounds, AHScrollBar, AVScrollBar, AIndicator, 0);
  if AFileName <> '' then
  begin
    FileName := FExpand(AFileName);

```



```

    if IsValid then IsValid := LoadFile;
  end;
end;

constructor TFileEditor.Load(var S: TStream);
var
  SStart, SEnd, Curs: Word;
begin
  TEditor.Load(S);
  S.Read(Filename[0], SizeOf(Char));
  S.Read(Filename[1], Length(Filename));
  if IsValid then IsValid := LoadFile;
  S.Read(SStart, SizeOf(Word));
  S.Read(SEnd, SizeOf(Word));
  S.Read(Curs, SizeOf(Word));
  if IsValid and (SEnd <= BufLen) then
  begin
    SetSelect(SStart, SEnd, Curs = SStart);
    TrackCursor(True);
  end;
end;

procedure TFileEditor.DoneBuffer;
begin
  if Buffer <> nil then DisposeBuffer(Buffer);
end;

procedure TFileEditor.HandleEvent(var Event: TEvent);
begin
  TEditor.HandleEvent(Event);
  case Event.What of
    evCommand:
      case Event.Command of
        cmSave: Save;
        cmSaveAs: SaveAs;
      else
        Exit;
      end;
    else
      Exit;
  end;
  ClearEvent(Event);
end;

procedure TFileEditor.InitBuffer;
begin
  NewBuffer(Pointer(Buffer));
end;

function TFileEditor.LoadFile: Boolean;
var
  Length: Word;
  FSize: Longint;
  F: File;
begin
  LoadFile := False;
  Length := 0;
  Assign(F, FileName);
  Reset(F, 1);
  if IOResult <> 0 then LoadFile := True else

```

```

begin
  FSize := FileSize(F);
  if (FSize > $FFF0) or not SetBufSize(Word(FSize)) then
    EditorDialog(edOutOfMemory, nil) else
  begin
    BlockRead(F, Buffer^[BufSize - Word(FSize)], FSize);
    if IOResult <> 0 then EditorDialog(edReadError, @FileName) else
    begin
      LoadFile := True;
      Length := FSize;
    end;
  end;
  Close(F);
end;
SetBufLen(Length);
end;

function TFileEditor.Save: Boolean;
begin
  if FileName = '' then Save := SaveAs else Save := SaveFile;
end;

function TFileEditor.SaveAs: Boolean;
begin
  SaveAs := False;
  if EditorDialog(edSaveAs, @FileName) <> cmCancel then
  begin
    FileName := FExpand(FileName);
    Message(Owner, evBroadcast, cmUpdateTitle, nil);
    SaveAs := SaveFile;
    if IsClipboard then FileName := '';
  end;
end;

function TFileEditor.SaveFile: Boolean;
var
  F: File;
  BackupName: FNameStr;
  D: DirStr;
  N: NameStr;
  E: ExtStr;
begin
  SaveFile := False;
  if EditorFlags and efBackupFiles <> 0 then
  begin
    FSplit(FileName, D, N, E);
    BackupName := D + N + '.BAK';
    Assign(F, BackupName);
    Erase(F);
    Assign(F, FileName);
    Rename(F, BackupName);
    InOutRes := 0;
  end;
  Assign(F, FileName);
  Rewrite(F, 1);
  if IOResult <> 0 then EditorDialog(edCreateError, @FileName) else
  begin
    BlockWrite(F, Buffer^, CurPtr);
    BlockWrite(F, Buffer^[CurPtr + GapLen], BufLen - CurPtr);
    if IOResult <> 0 then EditorDialog(edWriteError, @FileName) else

```

```

begin
  Modified := False;
  Update(ufUpdate);
  SaveFile := True;
end;
Close(F);
end;
end;

function TFileEditor.SetBufSize(NewSize: Word): Boolean;
var
  N: Word;
begin
  SetBufSize := False;
  if NewSize > $F000 then NewSize := $FFF0 else
    NewSize := (NewSize + $0FFF) and $F000;
  if NewSize <> BufSize then
    begin
      if NewSize > BufSize then
        if not SetBufferSize(Buffer, NewSize) then Exit;
        N := BufLen - CurPtr + DelCount;
        Move(Buffer^[BufSize - N], Buffer^[NewSize - N], N);
        if NewSize < BufSize then SetBufferSize(Buffer, NewSize);
        BufSize := NewSize;
        GapLen := BufSize - BufLen;
      end;
      SetBufSize := True;
    end;
end;

procedure TFileEditor.Store(var S: TStream);
begin
  TEditor.Store(S);
  S.Write(FileName, Length(FileName) + 1);
  S.Write(SelStart, SizeOf(Word) * 3);
end;

procedure TFileEditor.UpdateCommands;
begin
  TEditor.UpdateCommands;
  SetCmdState(cmSave, True);
  SetCmdState(cmSaveAs, True);
end;

function TFileEditor.Valid(Command: Word): Boolean;
var
  D: Integer;
begin
  if Command = cmValid then Valid := IsValid else
    begin
      Valid := True;
      if Modified then
        begin
          if FileName = '' then D := edSaveUntitled else D := edSaveModify;
          case EditorDialog(D, @FileName) of
            cmYes: Valid := Save;
            cmNo: Modified := False;
            cmCancel: Valid := False;
          end;
        end;
      end;
    end;
end;
end;

```

```

end;

( TEditWindow )

constructor TEditWindow.Init(var Bounds: TRect;
  FileName: FNameStr; ANumber: Integer);
var
  HScrollBar, VScrollBar: PScrollBar;
  Indicator: PIndicator;
  R: TRect;
begin
  TWindow.Init(Bounds, '', ANumber);
  Options := Options or ofTileable;
  R.Assign(18, Size.Y - 1, Size.X - 2, Size.Y);
  HScrollBar := New(PScrollBar, Init(R));
  HScrollBar.Hide;
  Insert(HScrollBar);
  R.Assign(Size.X - 1, 1, Size.X, Size.Y - 1);
  VScrollBar := New(PScrollBar, Init(R));
  VScrollBar.Hide;
  Insert(VScrollBar);
  R.Assign(2, Size.Y - 1, 16, Size.Y);
  Indicator := New(PIndicator, Init(R));
  Indicator.Hide;
  Insert(Indicator);
  GetExtent(R);
  R.Grow(-1, -1);
  Editor := New(PFileEditor, Init(
    R, HScrollBar, VScrollBar, Indicator, FileName));
  Insert(Editor);
end;

constructor TEditWindow.Load(var S: TStream);
begin
  TWindow.Load(S);
  GetSubViewPtr(S, Editor);
end;

procedure TEditWindow.Close;
begin
  if Editor^.IsClipboard then Hide else TWindow.Close;
end;

function TEditWindow.GetTitle(MaxSize: Integer): TTitleStr;
begin
  if Editor^.IsClipboard then GetTitle := 'Clipboard' else
    if Editor^.FileName = '' then GetTitle := 'Untitled' else
      GetTitle := Editor^.FileName;
end;

procedure TEditWindow.HandleEvent(var Event: TEvent);
begin
  TWindow.HandleEvent(Event);
  if (Event.What = evBroadcast) and (Event.Command = cmUpdateTitle) then
    begin
      Frame^.DrawView;
      ClearEvent(Event);
    end;
end;
end;

```

```
procedure TEditWindow.Store(var S: TStream);
begin
  TWindow.Store(S);
  PutSubviewPtr(S, Editor);
end;
```

```
procedure RegisterEditors;
begin
  RegisterType(REditor);
  RegisterType(RMemo);
  RegisterType(RFileEditor);
  RegisterType(RIndicator);
  RegisterType(REditWindow);
end;

end.
```

```

{*****}
{
  Unit for operating the overall system
}
{
  Filename : Esea.pas      Author K G Mercer
  Date    : September 1993
}
{*****}
{$M 4000,0,0}
program Esea;

Uses Dos,Cntrlnt,Crt;

var KnowAllFile : File of Member;
    Membership  : 0..MaxMembers;
    OneMember   : Member;
    x,y         : string;
    Finished    : Boolean;

{*****}
Procedure Initilize;
{*****}
var
  i:integer;
begin
  Assign(KnowAllFile,CntrlFileName);
  {$I-}
  Rewrite(KnowAllfile);
  {$I+}

  For i:= 0 to MaxMembers do
  begin
    Seek(KnowAllFile,i);
    With OneMember do
    begin
      if i = 0 then
      begin
        WhereTo   := 'Thesis2.exe';
        Parameter := 'xyz MessageOK';
        TypeRec   := InterRec;
      end
      else
      begin
        WhereTo   := '0';
        Parameter := '0';
        TypeRec   := InterRec;
      end;
    end;
    Write(KnowAllFile,OneMember);
  end;
  Close(KnowAllFile);
end;

{*****}
Procedure ListAll;
{*****}
var
  i: integer;
begin
  ClrScr;

```

```

  {$I-}
  Reset(KnowAllfile);
  {$I+}
  GotoXY(0,2);
  Writeln(' Record WhereTo Parameter TypeRec');
  For i:= 0 to MaxMembers do
  begin
    Seek(KnowAllFile,i);
    Read(KnowAllFile,OneMember);
    GotoXY(3,i+4);
    Writeln(i);
    GotoXY(10,i+4);
    Writeln(OneMember.WhereTo);
    GotoXY(25,i+4);
    Writeln(OneMember.Parameter);
    GotoXY(40,i+4);
    if OneMember.TypeRec = FinRec then Writeln('FinRec');
    if OneMember.TypeRec = InterRec then Writeln('InterRec');
    Writeln;
  end;
  Readln(x);
end;

{*****}
Procedure RunNext;
{*****}
label
  CheckOut;
var
  i,x,RecNo : integer;
  Found     : Boolean;
  Send,Parcel : String;
  TypeString : RecordTypes;

begin
  Assign(KnowAllFile,CntrlFileName);
  {$I-}
  Reset(KnowAllfile);
  {$I+}
  Found:=False;
  i:=0;

  Repeat;
    Seek(KnowAllFile,i);
    Read(KnowAllFile,OneMember);
    If OneMember.WhereTo = '0' then Found:=True else Inc(i);
  Until Found;

  if ((Found=False) or (i=0)) then
  begin
    Close(KnowAllFile);
    Finished := True;
    Goto CheckOut;
  end;

  RecNo:=i-1;
  Seek(KnowAllFile,RecNo);
  Read(KnowAllFile,OneMember);
  TypeString := OneMember.TypeRec;

```

```
    if TypeString = FinRec then
        begin
        {      ListAll;
          }      Close(KnowAllFile);
                Finished := True;
        end;

    if TypeString = InterRec then
        begin
            Send := OneMember.WhereTo;
            Parcel := OneMember.Parameter;
        {      ListAll;
          }      Seek(KnowAllFile, RecNo);
                OneMember.WhereTo := '0';
                OneMember.Parameter := '0';
                OneMember.TypeRec := InterRec;
                Write(KnowAllFile, OneMember);
        {      ListAll;
          }      Close(KnowAllFile);
                Exec(GetEnv('COMSPEC'), '/C '+send+ ' '+parcel);
        end;
    CheckOut:
end;

BEGIN
    Initilize;
    Finished := False;
    Repeat
        RunNext;
    Until Finished;
END.
```

```

{*****}
{
{   Unit for managing the running of expert systems }
{
{   Filename : ExptMan.pas      Author K G Mercer }
{   Date    : September 1993 }
{
{   Objects  : TDBaseDialog    }
{             : TInfoWindow    }
{             : TInterior1     }
{             : TChooseDialog  }
{
{   Other    : String40        }
{             : TDinfoRec1     - record }
{             : TDinfoRec3     - record }
{             : DChooseData    - record }
{*****}
{$X+}
Unit ExptMan;

interface

uses { Standard Units }
     Crt, Dialogs, Drivers, Views, Objects, MsgBox, Menus, Memory, Dos,
     { Other }
     ViewWinF, CntrlUnit, ViewWin2,
     { Expert System related Units }
     CmdnList, StrmUnit, General1, General2, ListADlg, ListFDlg,
     { Help }
     Help;

type
string40 = string[40];

TDinfoRec1 = record
    DisplayName : string[35];
    ProgToRun   : string[8];
    RBName      : string[8];
    Description1 : string[60];
    Description2 : string[60];
    Description3 : string[60];
end;

TDinfoRec3 = record
    Name : String[20];
end;

{ DChooseData record for use in ViewStream Method selecting
  type of spray applications to use }

DChooseData = record
    CheckBoxData1: Word;
    CheckBoxData2: Word;
    InputLineData: string[28];
end;

PDBaseDialog = ^TDBaseDialog;
TDBaseDialog = object(TDialog)
    constructor Init;
    destructor Done; virtual;

```

```

    procedure HandleEvent(var Event:TEvent); virtual;
    procedure ViewStream;
    procedure RunExpertRBases;
    procedure RunBatchAnalysis;
    procedure AddExpertRBases;
    procedure EditExpertRBases;
    procedure DeleteExpertRBases;
    procedure NoBlanks(var S:String);
    procedure LoadStream;
end;

PInfoWindow = ^TInfoWindow;
TInfoWindow = object(TWindow)
    constructor Init(Bounds: TRect; WinTitle: String; WindowNo: Word);
    procedure HandleEvent(var Event: TEvent); virtual;
end;

PInterior1 = ^TInterior1;
TInterior1 = object(TView)
    constructor Init(var Bounds: TRect);
    procedure Draw; virtual;
end;

PChooseDialog = ^TChooseDialog;
TChooseDialog = object(TDialog)
end;

var
    LineCount, yy      : Integer;
    Actual, AllControl: Word;
    TDinfo1            : TDinfoRec1;
    TDinfo3            : TDinfoRec3;
    Finished, DupeRec  : Boolean;
    DataPointer        : Pointer;

{ Checked }
AppStreamName      : String;
TransWCode         : TransferWithCode; {declared in ListADlg}
ExpertList         : PCollection;
ExpertStream       : TBufStream;
BatchList          : PCollection;
BatchDataList      : PCollection;
BatchStream        : TBufStream;
DialogChooseData   : DChooseData;
TInfoR1            : TDinfoRec3;
KnowAllFile        : File of Member;
RezName, StreamName: String[12];
BatchRez           : TResourceFile;

implementation

uses App;

{*****}
{ ***** TInterior1 ***** }
{*****}

constructor TInterior1.Init(var Bounds: TRect);
begin
    TView.Init(Bounds);

```

```

GrowMode := gfGrowHiX + gfGrowHiY;
Options := Options or ofFramed;
end;

procedure TInterior1.Draw;
var
  s1,s2,s3,s4 : string;
begin
  TView.Draw;
  Str(LineCount,s1);
  Str(yy,s4);
  s2:= ' Line Number = '+s1;
  WriteStr(4, 2,s2,$02);
  if DupeRec or (yy >= 1) then
  begin
    s3:= ' Duplicate Records Found = '+s4;
    WriteStr(4, 4,s3,$02);
    DupeRec:=false;
  end;
end;

{*****}
{***** TInfoWindow *****}
{*****}

constructor TInfoWindow.Init(Bounds: TRect; WinTitle: String; WindowNo: Word);
var
  S : string[3];
  Interior1 : PInterior1;
begin
  Str(WindowNo, S);
  TWindow.Init(Bounds, WinTitle + ' ' + S, wnNoNumber);
  GetClipRect(Bounds);
  Bounds.Grow(-1,-1);
  Palette:=1;
  Interior1 := New(PInterior1, Init(Bounds));
  Insert(Interior1);
end;

procedure TInfoWindow.HandleEvent(var Event: TEvent);
begin
  { Respond to Automatic Window Cancel }
  with Event do
    if (What=evBroadcast) AND (Command=cmCloseInfoWindow) then
    begin
      ClearEvent(Event);
      what := evCommand;
      Command := cmClose;
      PutEvent(Event);
    end;

  TWindow.HandleEvent(Event);
end;

{*****}
{***** TDBaseDialog *****}
{*****}
{*****}

```

```

(* LoadStream *)
{*****}
procedure TDBaseDialog.LoadStream;
begin
  with ExpertStream do
  begin
    Init(AppStreamName, stOpenRead, 1024); { Open stream }
    ExpertList := PCollection(Get); { Load collection }
    Done; { Shut down stream }
    if Status <> 0 then { Check for error }
    begin
      MessageBox(#3'Error loading : '+AppStreamName, nil, mfInformation + mfOkBut
ton);
    end;
  end;
end;

{*****}
{***** Init *****}
{*****}
constructor TDBaseDialog.Init;
var
  R : TRect;
  C : Word;
  N : string[30];
  i : integer;
  Q : PView;
  LoadName : String;
  Event : TEvent;
begin
  finished:=false;
  N:='Expert System Control Centre';
  R.Assign(1,1, 65, 21);
  R.Move(7,0);
  TDialog.Init(R,N);

  { Buttons }
  { Column 1 }
  R.Assign(5, 2, 19, 4);
  Insert(New(PButton, Init(R, ' ~R~un ', cmRunRec, bfNormal)));
  R.Assign(19, 4, 46, 6);
  Insert(New(PButton, Init(R, ' ~R~un Batch Analysis', cmRunBAREc, bfNormal)));

  R.Assign(3, 8, 17, 10);
  Insert(New(PButton, Init(R, ' ~L~ist ', cmViewAll, bfNormal)));
  R.Assign(3, 10, 17, 12);
  Insert(New(PButton, Init(R, ' ~A~dd ', cmAddRec, bfNormal)));
  R.Assign(3, 12, 17, 14);
  Insert(New(PButton, Init(R, ' ~E~dit ', cmEditRec, bfNormal)));
  R.Assign(3, 14, 17, 16);
  Insert(New(PButton, Init(R, ' ~D~elete ', cmDelRec, bfNormal)));

  { Column 2 }
  R.Assign(15, 17, 28, 19);
  Insert(New(PButton, Init(R, ' ~H~elp ', cmHelp, bfNormal)));
  R.Assign(35, 17, 48, 19);
  Insert(New(PButton, Init(R, ' ~Q~uit ', cmOK, bfDefault)));

```

```

{ Labels }
{ Column 1 }
R.Assign(20, 2, 61,3);
Insert(New(PLabel, Init(R, 'a Registered Expert System Rulebase', Q)));
{ R.Assign(20, 4, 61,5);
  Insert(New(PLabel, Init(R, '', Q)));
} R.Assign(1, 6, 63,7);
Insert(New(PLabel, Init(R, 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaa', Q)));
R.Assign(17, 8, 61,9);
Insert(New(PLabel, Init(R, 'of Registered Rulebases', Q)));
R.Assign(17, 10, 61,11);
Insert(New(PLabel, Init(R, 'an Expert System Rulebase to the Register', Q)));
R.Assign(17, 12, 61,13);
Insert(New(PLabel, Init(R, 'an Expert System Rulebase Registration Info.' , Q)
));
R.Assign(17, 14, 61,15);
Insert(New(PLabel, Init(R, 'an Expert System Rulebase from the Register', Q)));

```

```

{ ExpertStream,ExpertList declared in StrmUnit.pas }
{ ExpertStream registration done in Thesis2.pas }
AppStreamName := 'ExptList.Stm';
end;
```

```
(*****  
(*                                     Done                                     *)  
(***)
```

```
destructor TDBaseDialog.done;
begin
    { ExpertList closed at end of each module }
    TDialog.Done;
end;
```

```
(*****  
(*                               Handle Event                               *)  
(***)
```

```
procedure TDBaseDialog.HandleEvent(var Event: TEvent);
var
  i      : integer;
```

begin

```

( Respond to CANCEL button and ESC )
if ((Event.What = evKeyDown) and (Event.KeyCode = kbEsc)) or
   ((Event.What = evCommand) and (Event.Command = cmOK)) then
begin
  ClearEvent(Event);
  Finished := true;
  Free;
  Exit;
end;
end;

```

```
if (Event.What = evBroadcast) and (Event.Command = cmShutDown12) then
begin
  ClearEvent(Event);
  Finished := true;
  Free;
  Message(Application, evBroadcast, cmShutDown20, nil);
```

```
Exit;
end;

( Respond to Down Arrow Key )
( Respond to Down Arrow Key )
if ((Event.What = evKeyDown) and (Event.KeyCode = kbDown)) then
begin
    ClearEvent(Event);
    SelectNext(False);
end;

( Respond to Up Arrow Key )
if ((Event.What = evKeyDown) and (Event.KeyCode = kbUp)) then
begin
    ClearEvent(Event);
    SelectNext(True);
end;
```

TDialog.HandleEvent(Event):

```
if Event.What=evCommand then
begin
    case Event.Command of
        cmViewAll      : ViewStream;
        cmRunRec       : RunExpertRBases;
        cmRunBAREC     : RunBatchAnalysis;
        cmAddRec       : AddExpertRBases;
        cmEditRec      : EditExpertRBases;
        cmDelRec       : DeleteExpertRBases;
```

```

    end; { Case }
    ClearEvent(Event);
end;
end: { HandleEvent }

```

```

(*****
* Batch Analysis
*****

```

```
procedure TDBaseDialog.RunBatchAnalysis;
label
```

```
Checkout;  
type  
String2 = String[2];  
String4 = String[4];
```

```
var
  PLFD : PListFDialog;
  Collection1 : PDataCollection1;
  Control1,Control2,Option: Word;
  CodeReturned,i,IndexNo : Integer;
  FoundBO : PBatchObject;
  RetrieveKeyName,Name : String;
  RetrieveBatchName : String;
  ESList : PCollection;
  AddFirst,Found : Boolean;
  ESysName2,Weight2Str : String;
  Weight2,Code2,RBCode2 : Integer;
  ArrayStr,ProgName2,RBName2 : String;
  TotalWeight,RBCode3 : Integer;
  Outfile : Text;
  ShowWindow : PDemoWindow2;
```



```

R          : TRect;
OneMember  : Member;
PEditBDObject, PNewBDObject : PBatchDataObject;
ESysDescrip2, NewBatchName   : String;
ExpertNameFoundPtr          : PExpertObject;
ESysDescrip3, ProgName3, RName3 : String;
RBaseCodeStr, BatchNameStr2    : String2;
BatchNameStr4                 : String4;

procedure CallWrite1(P: PBatchObject); far;
begin
  Collection1^.Insert(New(PRecD1, Init(P^.BatchName^, P^.PCode^)));
end;

function CodeMatch(Batch: PBatchObject): Boolean; far;
begin
  CodeMatch := CodeReturned = Batch^.PCode^;
end;

function NameMatch(PEO: PExpertObject): Boolean; far;
begin
  NameMatch := ESysName2 = PEO^.Name^;
end;

begin
  ( Retrieve Batchlist from Resource File )
  RezName := 'Batch.Rez';
  BatchRez.Init(New(PBufStream, Init(RezName, stOpen, 1024)));
  if BatchRez.Stream^.Status <> 0 then
    MessageBox(#3'Error loading Resourcefile', nil, mfInformation + mfOkButton)
  ;

  BatchList := PCollection(BatchRez.Get('KeyNames'));
  BatchRez.Done;

  Collection1 := New(PDataCollection1, Init(BatchList^.Count - 1, 5));

  for i:=0 to BatchList^.Count - 1 do
  begin
    CallWrite1(BatchList^.At(i));
  end;

  ( Display list and Select BatchObject to Run )
  DisableCommands([cmDelBatchObject, cmAddBatchObject]);
  PLFD := New(PListFDDialog, Init(Collection1));
  PLFD^.HelpCtx := hcRunBatchRulebases;
  Control1 := Desktop^.ExecView(PLFD);
  if Control1 <> cmCancel then
  begin
    PLFD^.GetData(TransWCode);
    CodeReturned := TransWCode.RecCode;
  end;
  Dispose(PLFD, Done);
  EnableCommands([cmDelBatchObject, cmAddBatchObject]);

  ( Search for BatchObject with the code returned )

  if Control1 <> cmCancel then
  begin
    FoundBO := BatchList^.FirstThat(@CodeMatch);

```

```

IndexNo := BatchList^.IndexOf(FoundBO);

RetrieveBatchName := PBatchObject(BatchList^.At(IndexNo)).BatchName^;
RetrieveKeyName := PBatchObject(BatchList^.At(IndexNo)).BatchCallName^;

( Retrieve corresponding Expert System List from resource file )
RezName := 'Batch.Rez';
BatchRez.Init(New(PBufStream, Init(RezName, stOpen, 1024)));
if BatchRez.Stream^.Status <> 0 then
  MessageBox(#3'Error loading stream', nil, mfInformation + mfOkButton);

ESList := PCollection(BatchRez.Get(RetrieveKeyName));

with ExpertStream do
begin
  StreamName := 'ExptList.Stm';
  Init(StreamName, stOpenRead, 1024);
  ExpertList := PCollection(Get);
  Done;
  if Status <> 0 then
  begin
    MessageBox(#3'Error loading : '+StreamName, nil, mfInformation + mfOkBut
ton);
  end;
end;

( Check that ESList is uptodate )

TotalWeight := 0;
for i:=0 to ESList^.Count - 1 do
begin
  PEditBDObject := PBatchDataObject(ESList^.At(i));
  ESysName2 := PEditBDObject^.ESysName^;
  ESysDescrip2 := PEditBDObject^.ESysDescrip^;
  ProgName2 := PEditBDObject^.ProgName^;
  RName2 := PEditBDObject^.RName^;
  RCode2 := PEditBDObject^.RCode^;
  Weight2 := PEditBDObject^.PWeight^;
  Code2 := PEditBDObject^.PCode^;

  TotalWeight := TotalWeight + Weight2;

  ExpertNameFoundPtr := ExpertList^.FirstThat(@NameMatch);
  if ExpertNameFoundPtr <> nil then
  begin
    ESysDescrip3 := ExpertNameFoundPtr^.Descrip1^;
    ProgName3 := ExpertNameFoundPtr^.ProgName^;
    RName3 := ExpertNameFoundPtr^.RuleBaseName^;
    RCode3 := ExpertNameFoundPtr^.PCode^;

    if ((ESysDescrip2 <> ESysDescrip3) or (ProgName2 <> ProgName3) or
      (RName2 <> RName3) or (RCode2 <> RCode3)) then
    begin
      ESList^.AtFree(i);
      ESList^.Pack;
      PNewBDObject := New(PBatchDataObject, Init(ESysName2, ESysDescrip3,
        ProgName3, RName3, RCode3, Weight2, Code
2));

```

```

        ESList^.AtInsert(i,PNewBDOObject);
    end;
end
else
begin
    MessageBox(#3'Problem ! - Could not find '+#13#13#3+
        'rulebase : '+ESysName2, nil, mfInformation + mfOkButton
);
    BatchRez.Done;
    Goto CheckOut;
end;
end;

BatchRez.Put(ESList,RetrieveKeyName);
BatchRez.Done;

if TotalWeight <> 100 then
begin
    MessageBox(#3'The sum of the weightings'+#13#3+
        'does not = 100, re-check'+#13#3+
        'batchrun setup',nil, mfInformation + mfOkButton);
    Goto CheckOut;
end;

( Make up dos batch file to run )

( Obtain first 4 and last 2 characters in RetrieveBatchName )
if Length(RetrieveBatchName) >= 6 then
begin
    BatchNameStr4 := Copy(RetrieveBatchName,1,4);
    BatchNameStr2 := Copy(RetrieveBatchName,Length(RetrieveBatchName)-1,2);
    NewBatchName := BatchNameStr4 + BatchNameStr2;
end
else NewBatchName := RetrieveBatchName;

Assign(outfile,'Runfile.bat');
Rewrite(outfile);
for i:=0 to ESList^.Count -1 do
begin
    ProgName2 := PBatchDataObject(ESList^.At(i))^ProgName^;
    RBName2 := PBatchDataObject(ESList^.At(i))^RBName^;
    RBCode2 := PBatchDataObject(ESList^.At(i))^RBCode^;
    Str(RBCode2:2,RBaseCodeStr);
    if RBaseCodeStr[1]=' ' then RBaseCodeStr[1]:=chr(48);

    Writeln(outfile,ProgName2,' run -f',RBName2,' -r',NewBatchName,RBaseCod
eStr,'/d');
end;
System.Close(outfile);
Dispose(ESList,Done);

( View batchfile )

Option := MessageBox(#3'Do you wish to view the batchfile'+#13#3+
    'before executing', nil, mfYesNoCancel);
if Option = cmYes then
begin
    Name:='runfile.bat';
    R.Assign(1,1,60,10);
    R.Move(9,2);

```

```

    ShowWindow := New(PDemoWindow2, Init(R, '', Name));
    Control2 := Desktop^.ExecView(ShowWindow);
    Dispose(ShowWindow, Done);
end;

Option := MessageBox(#3'Do you wish to quit before executing'+#13#3+
    'this batchfile', nil, mfYesNoCancel);
if Option = cmNo then
begin
    ( Run Batchfile )

    Assign(KnowAllFile,CntrlFileName);
    ($I-)
    ReSet(KnowAllfile);
    ($I+)
    i:= 1;
    Found:=False;
    While not Found do
    begin
        Seek(KnowAllFile,i);
        Read(KnowAllFile,OneMember);
        If OneMember.WhereTo = '0' then Found:=True else inc(i);
    end;

    if i=0 then i:=1;
    Seek(KnowAllFile,i-1);
    Read(KnowAllFile,OneMember);
    OneMember.WhereTo := 'Thesis2.exe xyz';
    OneMember.Parameter := '';
    OneMember.TypeRec := InterRec;
    Write(KnowAllFile,OneMember);

    Seek(KnowAllFile,i);
    Read(KnowAllFile,OneMember);
    OneMember.WhereTo := 'Runfile.bat';
    OneMember.Parameter := '';
    OneMember.TypeRec := InterRec;
    Write(KnowAllFile,OneMember);

    System.Close(KnowAllFile);

    if Message(Application,evBroadcast,cmShutDown12,nil) = nil then
        MessageBox('no body handled', nil,mfYesNoCancel);
    end; ( Option )
end;( Control1 )
CheckOut:

end; ( RunBatchAnalysis )

(*****
(* Run ExpertSystem Rulebases
*****
)
procedure TDBaseDialog.RunExpertRBases;
label
    Start,Finished,Again;
type
    String2 = String[2];
    String4 = String[4];
var
    PLFD
        : PListFDDialog;

```

```

Collection1      : PDataCollection1;
Collection4      : PDataCollection4;
Name2,Descrip1_2,Descrip2_2 : String;
Descrip3_2,Msg   : String;
i,EditAppNo,ErrorNo : Integer;
PLA2d            : PListA2Dialog;
PED2            : PEditDialog2;
Option1,Option2,Control1,Control2 : Word;
Option3,Option4,Option5 : Word;
DisplayName,ProgToRun,RunString : String;
Code2,CodeReturned : Integer;
FoundApp         : PExpertObject;
PNewBatchObject  : PBatchObject;
TempName,TempProgName,TempRuleBaseName,
TempDescrip1,TempDescrip2,TempDescrip3 : PString;
BatchName,ParamStr,NewBatchName : String;
NewCode,RecCode,RBaseCode,IndexNo : Integer;
NameMatch,Found : Boolean;
RBaseCodeStr,BatchNameStr2 : String2;
BatchNameStr4 : String4;
OneMember      : Member;
Event          : TEvent;
FoundBO        : PBatchObject;

function CodeMatch(Batch:PBatchObject):Boolean;far;
begin
    CodeMatch := CodeReturned = Batch^.PCode^;
end;

function CodeMatch1(E0:PExpertObject):Boolean;far;
begin
    CodeMatch1 := CodeReturned = E0^.PCode^;
end;

procedure CallWrite1(P: PBatchObject); far;
begin
    Collection1^.Insert(New(PRecD1, Init(P^.BatchName^,P^.PCode^)));
end;

begin
    repeat
        Start:
        LoadStream;
        ( Read in data for each App as a string, compile string and insert )
        Collection4 := New(PDataCollection4,Init(ExpertList^.Count-1, 5));
        Collection4^.Duplicates := True;

        for i:= 0 to ExpertList^.Count-1 do
            begin
                Name2      := PExpertObject(ExpertList^.At(i))^Name^;
                Descrip1_2 := PExpertObject(ExpertList^.At(i))^Descrip1^;
                Descrip2_2 := PExpertObject(ExpertList^.At(i))^Descrip2^;
                Descrip3_2 := PExpertObject(ExpertList^.At(i))^Descrip3^;
                Code2      := PExpertObject(ExpertList^.At(i))^PCode^;

                Collection4^.Insert(New(PRecD4,Init(Name2,Descrip1_2,Descrip2_2,
                    Descrip3_2,Code2)));
            end;
        ( Display ListA2Dialog for Rulebase Selection )
    
```

```

PLA2d := New(PListA2Dialog,Init(Collection4));
PLA2d^.HelpCtx := hcRunRulebase;
Control1 := Desktop^.ExecView(PLA2d);
if Control1 <> cmCancel then
    begin
        PLA2d^.GetData(TransWCode);
        CodeReturned := TransWCode.RecCode;
    end;
Dispose(PLA2d, Done);

if Control1 <> cmCancel then
    begin
        ( Search for Application with the code returned )

        FoundApp := ExpertList^.FirstThat(@CodeMatch1);
        EditAppNo := ExpertList^.IndexOf(FoundApp);

        ( Correct Application corresponding to corect code found )
        DisplayName := PExpertObject(ExpertList^.At(EditAppNo))^Name^;
        ProgToRun := PExpertObject(ExpertList^.At(EditAppNo))^ProgName^;
        RunString := PExpertObject(ExpertList^.At(EditAppNo))^RuleBaseName^;
        RBaseCode := PExpertObject(ExpertList^.At(EditAppNo))^PCode^;

        ( Get Batchname for Rulebase Execution )

        Option1 := MessageBox(#3'Do you wish to use this rulebase'+#13+
            #3'for a batch run?', nil, mfYesNoCancel);
        if Option1 = cmNo then
            begin
                ( Assemble Parameters without Result File )

                BatchName[0]:=chr(6);
                ParamStr :='-f'+RunString;

                MessageBox(#3'Running rulebase without linking'+#13+
                    #3'+to Batch runs.'+#13+#3'Parameters are : '+ParamStr, nil, mfInform
                    ation + mfOkButton);

                Option4 := MessageBox(#3'Do you wish to quit before running'+#13+#3+
                    'this rulebase', nil, mfYesNoCancel);
                if Option4 = cmNo then
                    begin
                        ( Execute rulebase )
                        Assign(KnowAllFile,CntrlFileName);
                        ($I-)
                        ReSet(KnowAllfile);
                        ($I+)
                        i:= 0;
                        Found:=False;
                        While not Found do
                            begin
                                Seek(KnowAllFile,i);
                                Read(KnowAllFile,OneMember);
                                If OneMember.WhereTo ='0' then Found:=true else inc(i);
                            end;

                        if i=0 then i:=1;
                        Seek(KnowAllFile,i-1);
                        OneMember.WhereTo := 'Thesis2.exe xyz';
                        OneMember.Parameter := '';
                    end;
                
```

```

OneMember.TypeRec      := InterRec;
Write(KnowAllFile,OneMember);

Seek(KnowAllFile,i);
OneMember.WhereTo      := ProgToRun;
OneMember.Parameter    := ParamStr;
OneMember.TypeRec      := InterRec;
Write(KnowAllFile,OneMember);

System.Close(KnowAllFile);

ClearEvent(Event);

if Message(Application,evBroadcast,cmShutDown12,nil) = nil then
  MessageBox('no body handled', nil,mfYesNoCancel);
  Goto Finished;
end; { Option4 }
Goto Again;
end;

{ Retrieve Batchlist from Resource File}
RezName := 'Batch.Rez';
BatchRez.Init(New(PBufStream,Init(RezName,stOpen,1024)));
if BatchRez.Stream^.Status <> 0 then
  MessageBox(#3'Error loading Resourcefile', nil, mfInformation + mfOkB
utton);

BatchList := PCollection(BatchRez.Get('KeyNames'));
BatchRez.Done;

Collection1 := New(PDataCollection1,Init(BatchList^.Count -1, 5));

for i:=0 to BatchList^.Count -1 do
begin
  CallWrite1(BatchList^.At(i));
end;

{ Display list and Select BatchObject to Run }
DisableCommands([cmDelBatchObject,cmAddBatchObject]);
PLFD := New(PListFDDialog,Init(Collection1));
PLFD^.HelpCtx := hcRunRulebaseAndB;
Control2 := Desktop^.ExecView(PLFD);
if Control2 <> cmCancel then
begin
  PLFD^.GetData(TransWCode);
  CodeReturned := TransWCode.RecCode;
end;
Dispose(PLFD, Done);
EnableCommands([cmDelBatchObject,cmAddBatchObject]);

{ Search for BatchObject with the code returned }

if Control2 <> cmCancel then
begin
  FoundBO := BatchList^.FirstThat(@CodeMatch);
  IndexNo := BatchList^.IndexOf(FoundBO);

  BatchName := PBatchObject(BatchList^.At(IndexNo)).BatchName^;
  Option2 := MessageBox('Rulebase to run = '+BatchName+#13+

```

```

'Batchname of run = '+BatchName+#13+
'Is the information correct?', nil, mfYesNoCancel);
if Option2 = cmNo then goto start;

{ Assemble parameters }
Str(RBaseCode:2,RBaseCodeStr);
if RBaseCodeStr[1]=' ' then RBaseCodeStr[1]:=chr(48);

{ Obtain first 4 and last 2 characters in BatchName }
if Length(BatchName) >= 6 then
begin
  BatchNameStr4 := Copy(BatchName,1,4);
  BatchNameStr2 := Copy(BatchName,Length(BatchName)-1,2);
  NewBatchName := BatchNameStr4 + BatchNameStr2;
end
else NewBatchName := BatchName;

ParamStr :='-f'+runstring+' -r'+NewBatchName+RBaseCodeStr;
MessageBox(#3'Parameters are : '#13+#13#3+ParamStr, nil, mfInformatio
n + mfOkButton);

Option5 := MessageBox(#3'Do you wish to quit before running'+#13#3+
'this rulebase', nil, mfYesNoCancel);
if Option5 = cmNo then
begin
  { Execute rulebase }
  Assign(KnowAllFile,CntrlFileName);
  {$I-}
  ReSet(KnowAllfile);
  {$I+}
  i:= 1;
  Found:=False;
  While not Found do
  begin
    Seek(KnowAllFile,i);
    Read(KnowAllFile,OneMember);
    If OneMember.WhereTo = '0' then Found:=true else inc(i);
  end;

  if i=0 then i:=1;
  Seek(KnowAllFile,i-1);
  OneMember.WhereTo := 'Thesis2.exe xyz';
  OneMember.Parameter := '';
  OneMember.TypeRec := InterRec;
  Write(KnowAllFile,OneMember);

  Seek(KnowAllFile,i);
  Read(KnowAllFile,OneMember);
  OneMember.WhereTo := ProgToRun;
  OneMember.Parameter := ParamStr;
  OneMember.TypeRec := InterRec;
  Write(KnowAllFile,OneMember);

  System.Close(KnowAllFile);

  ClearEvent(Event);

  if Message(Application,evBroadcast,cmShutDown12,nil) = nil then
    MessageBox('no body handled', nil,mfYesNoCancel);
    Goto Finished;

```

```

    end; { Option5 }
  end; { Control 2 }
end; { Control 1 }
Again:
Msg := #3 + ' Run another ExpertSystem Rulebase? ';
Option3 := MessageBox(Msg, nil, mfYesNoCancel);

until (Option3 = cmNo) or (Option3 = cmCancel);
Finished:
end; { RunExpertRBases }

(*****
(* View Stream *)
*****)
procedure TDBaseDialog.ViewStream;

var
  outfile      : Text;
  i            : Integer;
  P            : PExpertObject;
  Name        : PathStr;
  Show1Window : PDemoWindow;
  R           : TRect;
  Bruce       : PView;
  DialogChoose : PChooseDialog;
  C           : Word;

begin
  { Load Stream }
  LoadStream;
  Assign(outfile, 'outtext.txt');
  Rewrite(outfile);
  Writeln(outfile);
  Writeln(outfile, 'Expert System Rulebase Information Listing');
  Writeln(outfile, '-----');
  Writeln(outfile);
  Writeln(outfile);
  Writeln(outfile);
  Writeln(outfile, 'Total number of rulebases registered is : ', ExpertList^.Count);
  Writeln(outfile);
  Writeln(outfile);
  Writeln(outfile);
  { Print out object records }
  for i:=0 to ExpertList^.Count -1 do
  begin
    P:=ExpertList^.At(i);
    Writeln(outfile, 'Rulebase Display Name : ', P^.Name^);
    Writeln(outfile, 'System run name : ', P^.ProgName^);
    Writeln(outfile, 'Rulebase file name : ', P^.RuleBaseName^);
    Writeln(outfile, 'Display description : ', P^.Descrip1^);
    Writeln(outfile, 'Description 2 : ', P^.Descrip2^);
    Writeln(outfile, 'Description 3 : ', P^.Descrip3^);
    Writeln(outfile);
  end;
  Writeln(outfile);
end;

```

```

  Writeln(outfile); Writeln(outfile); Writeln(outfile);
  System.Close(outfile);

  { Show scroll window }
  if C <> cmCancel then
  begin
    Name := 'outtext.txt';
    WinCount := 1;
    Desktop^.GetExtent(R);
    Show1Window := New(PDemoWindow, Init(R, 'ExpertSystem', WinCount, Name));

    Desktop^.Insert>Show1Window;
  end;
  Dispose(ExpertList, Done);

end;

(*****
(* Add Applications *)
*****)

procedure TDBaseDialog.AddExpertRBases;
label
  Finish, ReStart;
var
  RecordNumber : LongInt;
  TD1          : PInputDialog1;
  Event        : TEvent;
  Action1, Control1 : Word;
  PNewExpertObject : PExpertObject;
  DisplayName, ProgToRun, RBName, Description1, Description2,
  Description3 : String;
  ErrorNo, NewCode, RecCode, i : Integer;
  Action2 : Word;

begin
  Action2 := cmYes;
  repeat
    LoadStream;
    with TDinfo1 do
    begin
      DisplayName := '';
      ProgToRun := '';
      RBName := '';
      Description1 := '';
      Description2 := '';
      Description3 := '';
    end; { TDinfo1 }

    { Get information from InputDialog1 - general1.pas }

    ReStart:
    TD1:= New(PInputDialog1, Init);
    TD1^.SetData(TDinfo1);
    TD1^.HelpCtx := hcRulebaseInput;
    Control1:=Desktop^.ExecView(TD1);
    if Control1 <> cmCancel then TD1^.GetData(TDinfo1);
    Dispose(TD1, Done);
  end;
end;

```

```

{ Test and Insert Information }
if Control1 <> cmCancel then
begin
  DisplayName := TDinfo1.DisplayName ;
  ProgToRun   := TDinfo1.ProgToRun   ;
  RBName      := TDinfo1.RBName      ;
  Description1 := TDinfo1.Description1 ;
  Description2 := TDinfo1.Description2 ;
  Description3 := TDinfo1.Description3 ;

  { Check validity of input }
  if ((DisplayName='')or(ProgToRun='')or(RBName=''))then
  begin
    Action1 := MessageBox(#3'Input is not valid'+#13#13+#3+'Try again ?',
      nil,
      mfYesNoCancel);

    if Action1 = cmYes then
    begin
      DisplayName := 'DisplayName';
      ProgToRun   := 'ProgToRun';
      RBName      := 'RBName';
      Description1 := '';
      Description2 := '';
      Description3 := '';
      Goto ReStart;
    end
    else goto Finish;
  end;

  { NoBlanks(DisplayName );
  NoBlanks(ProgToRun );
  NoBlanks(RBName );
  NoBlanks(Description1);
  NoBlanks(Description2);
  NoBlanks(Description3);
  }
  if ((Description1 = '')or(Description2 = '')or(Description3 = '')) then
  begin
    if Description1 = '' then Description1 := DisplayName;
    if Description2 = '' then Description2 := ' ';
    if Description3 = '' then Description3 := ' ';
    MessageBox(#3'Input not complete, Information'+#13#13+#3+
      'saved, goto "Edit" to change', nil, mfInformation + mfOkButton);
  end;

  { Look for maximum code in list then increment by one }
  NewCode := 0;
  for i:= 0 to ExpertList.Count-1 do
  begin
    RecCode := PExpertObject(ExpertList.At(i))^PCode^;
    if RecCode > NewCode then NewCode := RecCode;
  end;
  NewCode:=NewCode+1;

  { Save application data }
  PNewExpertObject := New(PExpertObject,Init(DisplayName,ProgToRun,
      RBName,Description1,Description2,
      Description3,NewCode));

```

```

  ExpertList^.Insert(PNewExpertObject);

  ExpertStream.Init(AppStreamName,stCreate,1024);
  ExpertStream.Put(ExpertList);
  ExpertStream.Done;
  Dispose(ExpertList,Done);
end; { Control1 }

Action2 := MessageBox(#3' Add Another ExpertSystem RuleBase? ', nil,
  mfYesNoCancel);

until (Action2 = cmNo) or (Action2 = cmCancel);
Finish:

end; { AddExpertRBases }

(*****
(* Edit Applications *)
*****)
procedure TDBaseDialog.EditExpertRBases;
var
  Collection1      : PDataCollection4;
  Name2,Descrip1_2,Descrip2_2 : String;
  Descrip3_2,Msg   : String;
  i,EditAppNo,ErrorNo : Integer;
  Pld              : PListADialog;
  Option,Control1,Control2 : Word;
  DisplayName,ProgToRun,RBName : String;
  Description1,Description2,Description3 : String;
  TD1               : PInputDialog1;
  Code2,CodeReturned : Integer;
  FoundApp          : PExpertObject;
  TempName,TempProgName,TempRuleBaseName,
  TempDescrip1,TempDescrip2,TempDescrip3 : PString;

  function CodeMatch(E0:PExpertObject):Boolean;far;
  begin
    CodeMatch := CodeReturned = E0.PCode^;
  end;

begin
  Option := cmYes;
  repeat
    LoadStream;
    { Read in data for each App as a string, compile string and insert }
    Collection1 := New(PDataCollection4,Init(ExpertList.Count-1, 5));
    Collection1^.Duplicates := False;

    for i:= 0 to ExpertList.Count-1 do
    begin
      Name2      := PExpertObject(ExpertList.At(i))^Name^;
      Descrip1_2 := PExpertObject(ExpertList.At(i))^Descrip1^;
      Descrip2_2 := PExpertObject(ExpertList.At(i))^Descrip2^;
      Descrip3_2 := PExpertObject(ExpertList.At(i))^Descrip3^;
      Code2      := PExpertObject(ExpertList.At(i))^PCode^;

      Collection1^.Insert(New(PRecD4,Init(Name2,Descrip1_2,Descrip2_2,
        Descrip3_2,Code2)));
    end;

```

```

( Display ListADialog for Orchard Row Selection )
Pld := New(PListADialog,Init(Collection1));
Pld^.HelpCtx := hcEditRulebase;
Control1 := Desktop^.ExecView(Pld);
if Control1 <> cmCancel then
begin
    Pld^.GetData(TransWCode);
    CodeReturned := TransWCode.RecCode;
end;
Dispose(Pld, Done);

if Control1 <> cmCancel then
begin
    ( Search for Application with the code returned )

    FoundApp := ExpertList^.FirstThat(@CodeMatch);
    EditAppNo := ExpertList^.IndexOf(FoundApp);

    ( Correct Application corresponding to corect code found )
    with TDinfo1 do
    begin
        DisplayName := PExpertObject(ExpertList^.At(EditAppNo))^.Name^;
        ProgToRun := PExpertObject(ExpertList^.At(EditAppNo))^.ProgName^;
        RBName := PExpertObject(ExpertList^.At(EditAppNo))^.RuleBaseName^;

        Description1 := PExpertObject(ExpertList^.At(EditAppNo))^.Descrip1^;
        Description2 := PExpertObject(ExpertList^.At(EditAppNo))^.Descrip2^;
        Description3 := PExpertObject(ExpertList^.At(EditAppNo))^.Descrip3^;
    end; ( TDinfo1 )

    ( Get information from InputDialog1 )

    TD1:= New(PInputDialog1,Init);
    TD1^.HelpCtx := hcEditRulebaseData;
    TD1^.SetData(TDinfo1);
    Control2:=Desktop^.ExecView(TD1);
    if Control2 <> cmCancel then TD1^.GetData(TDinfo1);
    Dispose(TD1,Done);

    ( Test and Insert Information )
    if Control2 <> cmCancel then
    begin
        DisplayName :=TDinfo1.DisplayName ;
        ProgToRun :=TDinfo1.ProgToRun ;
        RBName :=TDinfo1.RBName ;
        Description1 :=TDinfo1.Description1 ;
        Description2 :=TDinfo1.Description2 ;
        Description3 :=TDinfo1.Description3 ;

        TempName := NewStr(DisplayName);
        TempProgName := NewStr(ProgToRun);
        TempRuleBaseName := NewStr(RBName);
        TempDescrip1 := NewStr(Description1);
        TempDescrip2 := NewStr(Description2);
        TempDescrip3 := NewStr(Description3);

        PExpertObject(ExpertList^.At(EditAppNo))^.Name := TempName;
        PExpertObject(ExpertList^.At(EditAppNo))^.ProgName := TempProgName;

```

```

PExpertObject(ExpertList^.At(EditAppNo))^.RuleBaseName := TempRuleBase
Name;
PExpertObject(ExpertList^.At(EditAppNo))^.Descrip1 := TempDescrip1;
PExpertObject(ExpertList^.At(EditAppNo))^.Descrip2 := TempDescrip2;
PExpertObject(ExpertList^.At(EditAppNo))^.Descrip3 := TempDescrip3;

( UpDate Stream )
ExpertStream.Init(AppStreamName,stCreate,1024);
ExpertStream.Put(ExpertList);
ExpertStream.Done;

DisposeStr(TempName);
DisposeStr(TempProgName);
DisposeStr(TempRuleBaseName);
DisposeStr(TempDescrip1);
DisposeStr(TempDescrip2);
DisposeStr(TempDescrip3);

end; ( Control2 )
end; ( Control1 )

Msg := #3 + ' Edit another ExpertSystem Record ? ' ;
Option := MessageBox(Msg, nil,mfYesNoCancel);

until (Option = cmNo) or (Option = cmCancel);

end; ( EditExpertRBases )

(*****
(* Delete Applications
*****
)
procedure TDBaseDialog.DeleteExpertRBases;
label
    Skip;
var
    Collection1 : PDataCollection4;
    Name2,Descrip1_2,Descrip2_2 : String;
    Descrip3_2,Msg : String;
    i,EditAppNo,ErrorNo,FormatNo : Integer;
    Pld : PListADialog;
    Action1,Action2,Control1,Control2 : Word;
    DisplayName,ProgToRun,RBName : String;
    Description1,Description2,Description3 : String;
    TD1 : PInputDialog1;
    Code2,CodeReturned : Integer;
    FoundApp : PExpertObject;

    function CodeMatch(App:PExpertObject):Boolean;far;
    begin
        CodeMatch := Codereturned = App^.PCode^;
    end;

begin
    repeat
        LoadStream;
        ( Read in data for each App as a string, compile string and insert )
        Collection1 := New(PDataCollection4,Init(ExpertList^.Count-1, 5));
        Collection1^.Duplicates := True;

        for i:= 0 to ExpertList^.Count-1 do

```

```

begin
  Name2      := PExpertObject(ExpertList^.At(i))^Name^;
  Descrip1_2 := PExpertObject(ExpertList^.At(i))^Descrip1^;
  Descrip2_2 := PExpertObject(ExpertList^.At(i))^Descrip2^;
  Descrip3_2 := PExpertObject(ExpertList^.At(i))^Descrip3^;
  Code2      := PExpertObject(ExpertList^.At(i))^PCode^;

  Collection1^.Insert(New(PRecD4, Init(Name2, Descrip1_2, Descrip2_2,
                                     Descrip3_2, Code2)));

end;

( Display ListADialog for Orchard Row Selection )

Pld := New(PListADialog, Init(Collection1));
Pld^.HelpCtx := hcDeleteRulebase;
Control1 := Desktop^.ExecView(Pld);
if Control1 <> cmCancel then
  begin
    Pld^.GetData(TransWCode);
    CodeReturned := TransWCode.RecCode;
  end;
Dispose(Pld, Done);

if Control1 <> cmCancel then
begin
  ( Search for Application with the code returned )

  FoundApp := ExpertList^.FirstThat(@CodeMatch);
  EditAppNo := ExpertList^.IndexOf(FoundApp);

  ( Correct Application corresponding to corect code found )
  with TDinfo1 do
  begin
    DisplayName := PExpertObject(ExpertList^.At(EditAppNo))^Name^;
    ProgToRun    := PExpertObject(ExpertList^.At(EditAppNo))^ProgName^;
    RBName       := PExpertObject(ExpertList^.At(EditAppNo))^RuleBaseName^;

    Description1 := PExpertObject(ExpertList^.At(EditAppNo))^Descrip1^;
    Description2 := PExpertObject(ExpertList^.At(EditAppNo))^Descrip2^;
    Description3 := PExpertObject(ExpertList^.At(EditAppNo))^Descrip3^;
  end; ( TDinfo1 )

  ( Confirm Delete by Showing Info )

  TD1 := New(PInputDialog1, Init);
  TD1^.HelpCtx := hcDeleteRulebaseData;
  TD1^.SetData(TDinfo1);
  Control2 := Desktop^.ExecView(TD1);
  if Control2 <> cmCancel then TD1^.GetData(TDinfo1);
  Dispose(TD1, Done);

  ( Test and Delete Information )
  if Control2 <> cmCancel then
  begin
    Msg := #3+'Delete this Rulebase record'+#13+#13+#3+'OK';
    Action1 := MessageBox(Msg, nil, mfYesNoCancel);
    if (Action1 = cmNo) or (Action2 = cmCancel) then goto skip;

    ExpertList^.Free(FoundApp);
    ExpertList^.Pack;
  end;

```

```

    ExpertStream.Init(AppStreamName, stCreate, 1024);
    ExpertStream.Put(ExpertList);
    ExpertStream.Done;
    Dispose(ExpertList, Done);
  skip;
end; ( Control2 )
end; ( Control1 )

Msg := #3+'Delete Another ExpertSystem Record?';
Action2 := MessageBox(Msg, nil, mfYesNoCancel);
until (Action2 = cmNo) or (Action2 = cmCancel);

end; ( DeleteExpertRBases )

(*****
(*                               Noblanks                               *)
*****
)
Procedure TDbaseDialog.Noblanks(var S:String);
var j : integer;
begin
  j:=1;
  while (j <=length(s)) do
  begin
    if s[j] = ' ' then System.Delete(s, j, 1)
    else inc(j);
  end;
end; ( Noblanks )

(*****
(*                               General Procedures                       *)
*****
)

end. ( Unit )

```



```

{*****}
{
  Utilities for managing Expert Objects
}
{
  Filename : General1.pas      Author K G Mercer
  Date      : September 1993
}
{
  Objects : TInputDialog1
           : TInfoPane
           : TInfoPane2
           : TViewPane1
}
{*****}
{$X+}
unit General1;

interface

uses
  Dos, Objects, Memory, Drivers, Views, Dialogs, StdDlg, CmdList,
  Crt, MsgBox, Menus, ViewWinF, StrmUnit, App;

type
  { TCreateInputDialog1 is a dialog called from TManage.CreateOrchard }
  { to input row data }

  PInputDialog1 = ^TInputDialog1;
  TInputDialog1 = object(TDialog)
    constructor init;
    procedure HandleEvent(var Event:TEvent);virtual;
    destructor done;virtual;
  end;

  { TInfoPane is a TView that displays the information      }
  { about the currently selected orchard in TManage.ReviewOrchard }

  PInfoPane = ^TInfoPane;
  TInfoPane = object(TView)
    Info : PString;
    constructor Init(var Bounds:TRect;I : String);
    destructor Done; virtual;
    procedure Draw; virtual;
    function GetPalette: PPalette; virtual;
    procedure HandleEvent(var Event: TEvent); virtual;
  end;

  { TInfoPane2 is a TView that displays the information      }
  { about the currently selected orchard responding to broadcast }
  { message 'cmUpdatePane2' from ListHDialog (ListHDlg.pas) }

  PInfoPane2 = ^TInfoPane2;
  TInfoPane2 = object(TInfoPane)
    procedure HandleEvent(var Event: TEvent); virtual;
  end;

  PViewPane1 = ^TViewPane1;
  TViewPane1 = object(TDialog)
    Constructor Init(Descrip:String);
  end;

```

implementation

```

{*****}
{ ***** TViewPane1 ***** }
{*****}

constructor TViewPane1.Init(Descrip:String);
var
  R      : TRect;
  C      : Word;
  N,s    : string[30];
  ButtonX1,ButtonX2,ButtonWd : integer;
  i,y    : Integer;

begin
  N:= 'Description';
  R.Assign(1,1, 50, 8);
  R.Move(15,4);
  TDialog.Init(R,N);

  { InfoPane }
  R.Assign(2, 2, 47, 3);
  Insert(New(PInfoPane, Init(R,Descrip)));

  y:=4;
  { Buttons }
  ButtonX1 := 10;
  ButtonX2 := 30;
  ButtonWd := 11;
  R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~H~elp', cmHelp, bfNormal)));

  R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~O~k', cmOK, bfDefault)));
end;

{*****}
{ ***** TInputDialog1 ***** }
{*****}

constructor TInputDialog1.Init;
var
  R      : TRect;
  C      : Word;
  N      : String[20];
  ButtonX1,ButtonX2,ButtonWd : Integer;
  ButtonX3,i,y : Integer;
  Q      : PView;

begin
  N:= 'Input/Edit Rulebase Data';
  R.Assign(1,1, 45, 20);
  R.Move(16,1);
  TDialog.Init(R,N);

  { Labels }
  R.Assign(3, 2, 17,3);
  Insert(New(PLabel, Init(R, 'Display Name', Q)));
  R.Assign(3, 4, 17,5);

```

```

Insert(New(PLabel, Init(R, 'Program Name', Q)));
R.Assign(3, 6, 17, 7);
Insert(New(PLabel, Init(R, 'Rulebase Name', Q)));
R.Assign(3, 8, 40, 9);
Insert(New(PLabel, Init(R, 'Display Description', Q)));
R.Assign(3, 11, 40, 12);
Insert(New(PLabel, Init(R, 'Secondary Description', Q)));

R.Assign(28, 4, 34, 5);
Insert(New(PLabel, Init(R, '.exe', Q)));
R.Assign(28, 6, 34, 7);
Insert(New(PLabel, Init(R, '.rb', Q)));

( Input lines )
R.Assign(18, 2, 42, 3);
Insert(New(PInputLine, Init(R, 35)));
R.Assign(18, 4, 28, 5);
Insert(New(PInputLine, Init(R, 8)));
R.Assign(18, 6, 28, 7);
Insert(New(PInputLine, Init(R, 8)));
R.Assign(4, 9, 40, 10);
Insert(New(PInputLine, Init(R, 60)));
R.Assign(4, 12, 40, 13);
Insert(New(PInputLine, Init(R, 60)));
R.Assign(4, 13, 40, 14);
Insert(New(PInputLine, Init(R, 60)));

y:=16;
( Buttons )
ButtonX1 := 2;
ButtonX2 := 15;
ButtonX3 := 29;
ButtonWd := 12;

R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~C-ancel', cmCancel, bfNormal)));

R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~H-elp', cmHelp, bfNormal)));

R.Assign(ButtonX3, Y, ButtonX3 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~O-k', cmOK, bfDefault)));
SelectNext(False);
end;

destructor TInputDialog1.done;
begin
  TDialog.done;
end;

procedure TInputDialog1.HandleEvent(var Event: TEvent);
var
  i : integer;
begin
  ( Respond to CANCEL button and ESC )
  if (Event.What = evKeyDown) and (Event.KeyCode = kbEsc) then
  begin
    ClearEvent(Event);

```

```

Event.What := evCommand;
Event.Command := cmCancel;
PutEvent(Event);
end;

( Respond to ENTER button )
if (Event.What = evKeyDown) and (Event.KeyCode = kbEnter) then
begin
  ClearEvent(Event);
end;

( Respond to Down Arrow Key )
if ((Event.What = evKeyDown) and (Event.KeyCode = kbDown)) then
begin
  ClearEvent(Event);
  SelectNext(False);
end;

( Respond to Up Arrow Key )
if ((Event.What = evKeyDown) and (Event.KeyCode = kbUp)) then
begin
  ClearEvent(Event);
  SelectNext(True);
end;

TDialog.HandleEvent(Event);
end;

(*****
(*                               TInfoPane                               *)
(*****)

constructor TInfoPane.Init(var Bounds: TRect; I:String);
begin
  TView.Init(Bounds);
  EventMask := EventMask or evBroadcast;
  Info := NewStr(I);
end;

destructor TInfoPane.Done;
begin
  DisposeStr(Info);
  TView.Done;
end;

procedure TInfoPane.Draw;
var
  B : TDrawBuffer;
  Color : Word;
begin
  if length(Info^) > 42 then
  begin
    end;
    Color := GetColor($01);           ( Selecting Color )
    MoveChar(B, ' ', Color, Size.X);  ( Fill buffer with blanks )
    MoveStr(B[1], Info^, Color);      ( Move Info variable into buffer )
    WriteLine(0, 0, Size.X, Length(Info^), B); ( Write out buffer )
  end;
end;

```

```
function TInfoPane.GetPalette: PPalette;
const
  P: String[Length(CInfoPane)] = CInfoPane;
begin
  GetPalette := @P;
end;
```

```
procedure TInfoPane.HandleEvent(var Event: TEvent);
begin
  TView.HandleEvent(Event);
end;
```

```
(*****
(*)          TInfoPane2          (*)
(***)
```

```
procedure TInfoPane2.HandleEvent(var Event: TEvent);
begin
  if (Event.What = evBroadcast) and (Event.Command = cmUpdatePane2) then
  begin
    Info := PString(Event.InfoPtr);
    ClearEvent(Event);
    DrawView;
  end;
```

```
  TInfoPane.HandleEvent(Event);
end;
```

```
end. { Unit }
```

```

(*****)
(
( Utilities for managing Batch Objects )
(
( Filename : General2.pas      Author K G Mercer )
( Date   : September 1993    )
(
( Objects : TEditDialog1
(           : TEditDialog2
(           : TEditDialog3
(           : TEditDialog4
(           : TEditDialog5
(           : TPopInputLine0
(           : TPopInputLine1
(
( Other   : TInfoRec1        - record
(
(*****)
($X+)

```

```
unit General2;
```

```
interface
```

```
uses
```

```

Dos, Objects, Memory, Drivers, Views, Dialogs, StdDlg,
Crt, MsgBox, Menus, ViewWinF, App, Validate,
( ExpertSystem related units )
CmdnList, StrmUnit, PkList1;

```

```
type
```

```
( EditDialog1 used by DBaseDialog in Appman.pas for editing app records )
```

```

PEditDialog1 = ^TEditDialog1;
TEditDialog1 = object(TDialog)
  constructor init;
  procedure HandleEvent(var Event:TEvent);virtual;
  destructor done;virtual;
end;

PEditDialog2 = ^TEditDialog2;
TEditDialog2 = object(TEditDialog1)
  constructor init;
end;

PEditDialog3 = ^TEditDialog3;
TEditDialog3 = object(TEditDialog1)
  constructor init;
end;

PEditDialog4 = ^TEditDialog4;
TEditDialog4 = object(TEditDialog1)
  constructor init;
end;

PEditDialog5 = ^TEditDialog5;
TEditDialog5 = object(TEditDialog1)
  constructor init;
end;

```

```
( PopInputLine objects used in EditDialog1 in AppMan.pas to pop up
relevant picklist )
```

```

PPopInputLine0 = ^TPopInputLine0;
TPopInputLine0 = object(TInputLine)
  constructor Init(var Bounds:TRect; Len:Integer);
end;

PPopInputLine1 = ^TPopInputLine1;
TPopInputLine1 = object(TPopInputLine0)
  procedure HandleEvent(var Event: TEvent); virtual;
end;

```

```
( TInfoRec1 used by thesis2.pas in Analysis Method )
```

```

TInfoRec1 = record
  Name : String[20];
end;

```

```
var
```

```

BatchList : PCollection;
BatchStream : TBufStream;
TransWCode : TransferWithCode;
RezName : String[12];
BatchRez : TResourceFile;

```

```
implementation
```

```

(*****)
( ***** TEditDialog5 ***** )
(*****)

```

```
constructor TEditDialog5.Init;
```

```
var
```

```

R      : TRect;
C      : Word;
N,s    : string[30];
ButtonX1,ButtonX2,ButtonWd : integer;
ButtonX3,i,y : Integer;
Q      : PView;
Field  : PInputLine;

```

```
begin
```

```

N:='Weighting';
R.Assign(1,1, 40, 11);
R.Move(20,4);
TDialog.Init(R,N);

( Labels )
R.Assign(1, 2, 38,3);
Insert(New(PLabel, Init(R, 'Input revised weighting of rulebase', Q)));
R.Assign(1, 4, 13,5);
Insert(New(PLabel, Init(R, 'Weighting', Q)));

```

```

( Input lines )
R.Assign(14, 4, 19, 5);
Field := New(PInputLine, Init(R, 3));
Field^.SetValidator(New(PRangeValidator, Init(1,100)));
Insert(Field);

```

```
R.Assign(19, 4, 22,5);
```

```

Insert(New(PLabel, Init(R, '%', Q)));

y:=7;
{ Buttons }
ButtonX1 := 1;
ButtonX2 := 13;
ButtonX3 := 26;
ButtonWd := 11;
R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~C-ancel', cmCancel, bfDefault)));

R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~H-elp', cmHelp, bfNormal)));

R.Assign(ButtonX3, Y, ButtonX3 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~O-k', cmOK, bfNormal)));
SelectNext(False);
end;

{*****}
{ ***** TEditDialog4 ***** }
{*****}

constructor TEditDialog4.Init;
var
  R      : TRect;
  C      : Word;
  N,s    : string[30];
  ButtonX1,ButtonX2,ButtonWd : integer;
  ButtonX3,i,y : Integer;
  Q      : PView;
  Field  : PInputLine;
begin
  N:='New Batch Name';
  R.Assign(1,1, 40, 11);
  R.Move(25,4);
  TDialog.Init(R,N);

  { Labels }
  R.Assign(3, 2, 38,3);
  Insert(New(PLabel, Init(R, 'Input weighting of rulebase', Q)));
  R.Assign(3, 4, 15,5);
  Insert(New(PLabel, Init(R, 'Weighting', Q)));

  { Input lines }
  R.Assign(16, 4, 21, 5);
  Field := New(PInputLine, Init(R, 3));
  Field.SetValidator(New(PRangeValidator, Init(1,100)));
  Insert(Field);

  R.Assign(23, 4, 25,5);
  Insert(New(PLabel, Init(R, '%', Q)));

  y:=7;
  { Buttons }
  ButtonX1 := 1;
  ButtonX2 := 13;
  ButtonX3 := 26;
  ButtonWd := 11;

```

```

R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~C-ancel', cmCancel, bfDefault)));

R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~H-elp', cmHelp, bfNormal)));

R.Assign(ButtonX3, Y, ButtonX3 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~O-k', cmOK, bfNormal)));
SelectNext(False);
end;

{*****}
{ ***** TEditDialog3 ***** }
{*****}

constructor TEditDialog3.Init;
var
  R      : TRect;
  C      : Word;
  N,s    : string[30];
  ButtonX1,ButtonX2,ButtonWd : integer;
  ButtonX3,i,y : Integer;
  Q      : PView;
  Field  : PInputLine;
begin
  N:='New Batchrun Name';
  R.Assign(1,1, 40, 11);
  R.Move(25,4);
  TDialog.Init(R,N);

  { Labels }
  R.Assign(3, 2, 38,3);
  Insert(New(PLabel, Init(R, 'Input or Select Batchrun Name', Q)));
  R.Assign(3, 4, 22,5);
  Insert(New(PLabel, Init(R, 'Name of Batch Job', Q)));

  { Input lines }
  R.Assign(4, 5, 26, 6);
  Insert(New(PInputLine, Init(R, 20)));

  y:=7;
  { Buttons }
  ButtonX1 := 1;
  ButtonX2 := 13;
  ButtonX3 := 26;
  ButtonWd := 11;
  R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~C-ancel', cmCancel, bfDefault)));

  R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~H-elp', cmHelp, bfNormal)));

  R.Assign(ButtonX3, Y, ButtonX3 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~O-k', cmOK, bfNormal)));
  SelectNext(False);
end;

{*****}
{ ***** TeditDialog2 ***** }
{*****}

```

```
{*****}
```

```
constructor TEditDialog2.Init;
```

```
var
  R      : TRect;
  C      : Word;
  N,s    : string[30];
  ButtonX1,ButtonX2,ButtonWd : integer;
  ButtonX3,i,y : Integer;
  Q      : PView;
  Field  : PInputLine;
```

```
begin
```

```
  N:='Expert System Batch Run';
  R.Assign(1,1, 40, 14);
  R.Move(20,2);
  TDialog.Init(R,N);
```

```
  s:= Chr(23);
```

```
  ( Labels )
  R.Assign(3, 2, 38,3);
  Insert(New(PLabel, Init(R, 'Input or Select Batch Job Name', Q)));
  R.Assign(3, 4, 22,5);
  Insert(New(PLabel, Init(R, 'Name of Batch Job', Q)));
  R.Assign(26, 5, 28,6);
  Insert(New(PLabel, Init(R, s, Q)));
```

```
  ( Input lines )
  R.Assign(4, 5, 26, 6);
  Insert(New(PPopInputLine1, Init(R, 20)));
```

```
  ( Labels )
  R.Assign(3,7,37,8);
  Insert(New(PLabel, Init(R, 'Press right mouse button for', Q)));
  R.Assign(3,8,37,9);
  Insert(New(PLabel, Init(R, 'selection of batch job history', Q)));
```

```
  y:=10;
  ( Buttons )
  ButtonX1 := 1;
  ButtonX2 := 13;
  ButtonX3 := 26;
  ButtonWd := 11;
  R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~C~ancel', cmCancel, bfDefault)));
```

```
  R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~H~elp', cmHelp, bfNormal)));
```

```
  R.Assign(ButtonX3, Y, ButtonX3 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~O~k', cmOK, bfNormal)));
  SelectNext(False);
```

```
end;
```

```
{*****}
{ ***** TEditDialog1 ***** }
{*****}
```

```
constructor TEditDialog1.Init;
```

```
var
  R      : TRect;
  C      : Word;
  N,s    : string[30];
  ButtonX1,ButtonX2,ButtonWd : integer;
  ButtonX3,i,y : Integer;
  Q      : PView;
  Field  : PInputLine;
```

```
begin
```

```
  N:='ExpertSystem Batch Run Records';
  R.Assign(1,1, 40, 12);
  R.Move(20,3);
  TDialog.Init(R,N);
```

```
  s:= Chr(23);
```

```
  ( Labels )
  R.Assign(3, 2, 22,3);
  Insert(New(PLabel, Init(R, 'Name of Batch Job', Q)));
  R.Assign(26, 3, 28,4);
  Insert(New(PLabel, Init(R, s, Q)));
```

```
  ( Input lines )
  R.Assign(4, 3, 26, 4);
  Insert(New(PPopInputLine1, Init(R, 20)));
```

```
  ( Labels )
  R.Assign(3,5,37,6);
  Insert(New(PLabel, Init(R, 'Press right mouse button for', Q)));
  R.Assign(3,6,37,7);
  Insert(New(PLabel, Init(R, 'selection of batch job history', Q)));
```

```
  y:=8;
  ( Buttons )
  ButtonX1 := 1;
  ButtonX2 := 13;
  ButtonX3 := 26;
  ButtonWd := 11;
  R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~C~ancel', cmCancel, bfDefault)));
```

```
  R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~H~elp', cmHelp, bfNormal)));
```

```
  R.Assign(ButtonX3, Y, ButtonX3 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~O~k', cmOK, bfNormal)));
  SelectNext(False);
```

```
end;
```

```
destructor TEditDialog1.done;
begin
  ( if BatchList <> nil then Dispose(BatchList,Done);
  ) TDialog.done;
end;
```

```
{*****}
```

```
procedure TEditDialog1.HandleEvent(var Event: TEvent);
var
```

```

i : integer;

Procedure ShowPickList1;
var
  Collection1      : PDataCollection5;
  Pld              : PPickDialog1;
  ListNumber,a,i   : Integer;
  Control          : Word;
  NameHolder       : Pointer;
  CodeReturned     : Integer;
  FoundApp         : PBatchObject;

  function CodeMatch(Batch:PBatchObject):Boolean;far;
  begin
    CodeMatch := CodeReturned = Batch^.PCode^;
  end;

  procedure CallWrite1(P: PBatchObject); far;
  begin
    Collection1^.Insert(New(PRecD5, Init(P^.BatchName^,P^.PCode^)));
  end;

begin
  ( Read in batch info from ResourceFile )
  RezName := 'Batch.Rez';
  BatchRez.Init(New(PBufStream,Init(RezName,stOpen,1024)));
  if BatchRez.Stream^.Status <> 0 then
    MessageBox(#3'Error loading stream', nil, mfInformation + mfOkButton);

  BatchList := PCollection(BatchRez.Get('KeyNames'));

  Collection1 := New(PDataCollection5,Init(BatchList^.Count -1, 5));

  for i:=0 to BatchList^.Count -1 do
  begin
    CallWrite1(BatchList^.At(i));
  end;

  Pld := New(PPickDialog1,Init(Collection1));
  Control := Desktop^.ExecView(Pld);
  if control <> cmCancel then
  begin
    Pld^.GetData(TransWCode);
    CodeReturned := TransWCode.RecCode;
  end;
  Dispose(Pld, Done);

  ( Search for Application with the code returned )

  FoundApp := BatchList^.FirstThat(@CodeMatch);
  i := BatchList^.IndexOf(FoundApp);

  ( Name is of type PString )
  if Message(Application,evBroadcast,cmUpdatePopILine1,BatchList^.At(i)) = nil th
en
  MessageBox(#3'Message = nil', nil, mfInformation + mfOkButton);
  BatchRez.Done;

end;

```

```

(***** Main Section *****)
begin
  with Event do
    if What=evBroadcast then
    begin
      case Command of
        cmPopWindow1 : begin
                          ClearEvent(Event);
                          ShowPickList1;
                        end;
      end ( Case )
    end; ( What=evBroadcast )

    ( Respond to CANCEL button and ESC )
    if (Event.What = evKeyDown) and (Event.KeyCode = kbEsc) then
    begin
      ClearEvent(Event);
      Event.What := evCommand;
      Event.Command := cmCancel;
      PutEvent(Event);
    end;

    ( Respond to Down Arrow Key )
    if ((Event.What = evKeyDown) and (Event.KeyCode = kbDown)) then
    begin
      ClearEvent(Event);
      SelectNext(False);
    end;

    ( Respond to Up Arrow Key )
    if ((Event.What = evKeyDown) and (Event.KeyCode = kbUp)) then
    begin
      ClearEvent(Event);
      SelectNext(True);
    end;

    TDialog.HandleEvent(Event);

    if Event.What=evCommand then
    begin
      ClearEvent(Event);
    end;
  end;

  (*****
  (* TPopInputLine0 *)
  (*****
  constructor TPopInputLine0.Init(var Bounds: TRect; len:Integer);
  begin
    TInputLine.Init(Bounds,Len);
    EventMask := EventMask or evBroadcast or evMouseDown;
  end;

  (*****
  (* TPopInputLine1 *)
  (*****
  procedure TPopInputLine1.HandleEvent(var Event: TEvent);
  begin

```

```
if (Event.What = evKeyDown) and ( Event.KeyCode = kbEnter) or
  (Event.What = evMouseDown ) and (Event.Buttons = mbRightButton ) then
begin
  ClearEvent(Event);
  ( Broadcast handled by EditDialog1 in Appman.pas )
  if Message(Owner,evBroadcast,cmPopWindow1,nil) = nil then
  begin
    end;
  end;

  if (Event.What = evBroadcast) and (Event.Command = cmUpdatePopILine1) then
  begin
    Data := PRecD5(Event.InfoPtr)^.Name1;
    ClearEvent(Event);
    DrawView;
  end;

  TInputLine.HandleEvent(Event);
end;

end. ( Unit )
```



```
unit HELP;

interface

const
    hcBatchDataObjectSelect = 16;
    hcBatchESSelect         = 18;
    hcDeleteRulebase        = 27;
    hcDeleteRulebaseData    = 28;
    hcEditRulebase          = 25;
    hcEditRulebaseData      = 26;
    hcEditWeight            = 21;
    hcExpertManage          = 23;
    hcInfoPane              = 32;
    hcManage                 = 7;
    hcMenuBatchMan          = 15;
    hcMenuDatabases         = 13;
    hcMenuESysMan           = 22;
    hcMenuProjectAnalysis   = 14;
    hcMenuWindows           = 12;
    hcNewBatchName          = 17;
    hcNewWeight             = 20;
    hcNoContext             = 0;
    hcQuit                  = 35;
    hcResultAnalysis        = 33;
    hcResultSelect          = 34;
    hcRulebaseInput         = 24;
    hcRunBatchRulebases     = 31;
    hcRunRulebase           = 29;
    hcRunRulebaseAndB       = 30;
    hcSAbout                = 6;
    hcSCD                   = 11;
    hcSColor                = 3;
    hcSEdit                 = 9;
    hcSelectRulebase        = 19;
    hcSExit                 = 4;
    hcSIE                   = 10;
    hcSScreenSave           = 5;
    hcSVideoMode            = 2;
    hcSView                 = 8;
    hcSystem                = 1;

implementation

end.
```


When you pull down the Rulebase menu, you see

```
{View Rulebases      :SView}
{Edit Rulebases      :SEdit}
{Integrated Environment :SIE}
{Change Directory     :SCD}
```

```
*****
;
;topic SView
;View Rulebases É
;0000000000000000
```

View one or more rulebases.

```
{Rulebase Menu:Manage}
*****
;
;topic SEdit
;Edit Rulebases É
;0000000000000000
```

Run the editor provided with the system. One of more rulebases can be edited at a time.

```
{Rulebase Menu:Manage}
*****
;
;topic SIE
;SYNAPSE Integrated Environment É
;000000000000000000000000000000
```

This is SYNAPSE's own environment for editing, compiling, running plus using other associated features.

```
{Rulebase Menu:Manage}
*****
;
;topic SCD
;Change Directory É
;0000000000000000
```

This facility enables the user to retrieve or store files in directories other than : \ECSA.

```
{Rulebase Menu:Manage}
*****
;
;***** Windows Management *****
;
;topic MenuWindows
;PÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉ
;Windows
;000000000000
```

From this menu the options for selecting various window modes and manipulations.

See associated topic for more information :

```
{Resizing Windows :Resize}
{Selecting Next Window:Next}
{Tiling Windows:Tile}
```

```
{Cascading Windows:Cascade}
{Previous Windows:Prev}
{Closing Windows:Close}
```

```
*****
;
;topic Resize
;P Resizing Windows É
;00000000000000000000
```

This tool provides a means to resize the window currently highlighted.

```
{Windows Menu:MenuWindows}
```

```
*****
;
;topic Next
;P Select Next Windows É
;00000000000000000000
```

This tool provides a means to select the next window that is not currently highlighted.

```
{Windows Menu:MenuWindows}
```

```
*****
;
;topic Tile
;P Tiling Windows É
;000000000000000000
```

This tool provides the feature of tiling all the windows that are currently displayed on the screen.

```
{Windows Menu:MenuWindows}
```

```
*****
;
;topic Cascade
;P Cascade Windows É
;000000000000000000
```

This tool provides the feature of cascading windows on the screen one behind the other.

```
{Windows Menu:MenuWindows}
```

```
*****
;
;topic Prev
;P Previous Windows É
;000000000000000000
```

This tool provides a means to select the previous window not currently highlighted.

```
{Windows Menu:MenuWindows}
```

```
*****
;
;topic Close
;P Closing Windows É
;000000000000000000
```

This option closes the currently highlighted window.

{Windows Menu:MenuWindows}

```

*****
;*****
;*****      Databases      *****
;*****
;*****

```

```

.topic MenuDatabases
pEEEEEEEEEEEEEEEE
  Databases
  0000000000000000

```

From this menu the options for managing and reviewing databases that are associated with rulebases registered in the utility.

These facilities have not been implemented for the purposes of this system.

```

*****
;*****
;*****      Project Analysis      *****
;*****
;*****

```

```

.topic MenuProjectAnalysis
pEEEEEEEEEEEEEEEE
  Project Analysis
  0000000000000000

```

From this menu the options for managing batchruns and expert system rulebase execution can be selected.

See associated topic for more information :

{Batchrun Management :MenuBatchMan}
{Expert System Rulebase Management:MenuESysMan}

```

*****
;*****
;*****      Batchrun Management      *****
;*****
;*****

```

```

.topic MenuBatchMan
pEEEEEEEEEEEEEEEE
  Batchrun Management
  0000000000000000

```

This module of the program provides the user with tools for setting up, modifying and deleting batchruns.

{Project Analysis Menu}:MenuProjectAnalysis}

```

*****
;*****
;*****      BatchDataObjectSelect
;*****      p Select Batchrun to Modify  E
;*****      000000000000000000000000

```

This menu provides a means to select previously

setup batchruns to modify. If you wish to setup a new batchrun press the New button to input a new batchrun name.

```

*****
;*****
;*****      topic NewBatchName
;*****      p Input New BatchName  E
;*****      000000000000000000000000

```

This is where a new batchrun name is inputted. When the name has been inputted press the OK button.

```

*****
;*****
;*****      topic BatchESSelect
;*****      p Modify Rulebases for the Batchruns  E
;*****      00000000000000000000000000000000

```

This window provides a means to modify batchruns. This can be done by selecting new expert systems rulebases for the batchrun, deleting other rulebases from the batchrun or modifying the weights.

```

*****
;*****
;*****      topic SelectRulebase
;*****      p Select Rulebase  E
;*****      00000000000000000000

```

From this menu select a new rulebase to be included into the batchrun list of rulebases.

```

*****
;*****
;*****      topic NewWeight;
;*****      p Input Weighting of the Rulebase  E
;*****      00000000000000000000000000000000

```

Input the weighting of the rulebase selected.

```

*****
;*****
;*****      topic EditWeight;
;*****      p Edit Weighting of the Rulebase  E
;*****      00000000000000000000000000000000

```

Modify the weighting of the rulebase selected.

```

*****
;*****
;*****      Expert System Management      *****
;*****
;*****

```

```

.topic MenuESysMan
pEEEEEEEEEEEEEEEE
  Expert System Management
  00000000000000000000000000

```

This module of the program provides the user with tools for registering, modifying registration information and deleting rulebases from the register.

{Project Analysis Menu}:MenuProjectAnalysis}

```

*****
;topic ExpertManage
;  Manage Expert Systems É
;  00000000000000000000000000000000

```

This control centre provides a means by which the expert sytem rulebases registration information can be modified and rulebase or batchruns can be selected for execution.

```

*****
;topic RulebaseInput
;  RuleBase Input É
;  00000000000000000000000000000000

```

This input window is used for inputting the new rulebases registration information.

```

*****
;topic EditRuleBase
;  Edit Rulebase É
;  00000000000000000000000000000000

```

From this menu select a rulebase that you want to edit the registration information of.

```

*****
;topic EditRuleBaseData
;  Edit Rulebase Data É
;  00000000000000000000000000000000

```

This input window is used for modifying the rulebases registration information.

```

*****
;topic DeleteRulebase
;  Delete Rulebase É
;  00000000000000000000000000000000

```

From this menu select a rulebase that you want to delete from the register.

```

*****
;topic DeleteRuleBaseData
;  Delete Rulebase Data É
;  00000000000000000000000000000000

```

This is a window for confirming the deletion of the rulebase selected.

```

*****
;topic RunRulebase
;  Run Rulebase É
;  00000000000000000000000000000000

```

From this menu select a rulebase that you wish to execute.

```

*****

```

```

;topic RunRulebaseAndB
;  Run Rulebase for Batchrun É
;  00000000000000000000000000000000

```

From this menu select an optional batchrun to run the rulebase for.

```

*****
;topic RunBatchRulebases
;  Run Batchruns É
;  00000000000000000000000000000000

```

From this menu select a batchrun that you wish to execute.

```

*****
;topic InfoPane
;  Information Pane É
;  00000000000000000000000000000000

```

Information regarding the rulebase selected

```

*****
;***** Result Analysis *****
;*****

```

```

;topic ResultAnalysis
;PEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
;  Result Analysis Menu
;  00000000000000000000000000000000

```

From this menu select a batchrun that you wish to analyse the results of.

See associated topic for more information :

{Batch Analysis:ResultSelect}

```

*****
;topic ResultSelect
;  Select Batchrun to Analyse É
;  00000000000000000000000000000000

```

Select the batchrun to analyse the results of.

```

*****
;***** Quitting *****
;*****
;topic Quit
;PEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
;  Quitting
;  000000000000

```

This menu provides the user with the facility for exiting out of the utility.

```

(*****)
(
( Turbo Vision Demo )
( Copyright (c) 1992 by Borland International )
(
( Unit for managing the help facilities )
(
( Filename : HelpFile.pas )
(
( Objects : THelpTopic )
(           : THelpIndex )
(           : THelpFile )
(           : THelpViewer )
(           : THelpWindow )
(
( Other : TParagraph - record )
(        : TCrossRef - record )
(        : TCrossRefs - array )
(        : TIndexArray - array )
(        : TContextArray - array )
(
(*****)

```

```
unit HelpFile;
```

```
($F+,O+,X+,-R-)
```

```
interface
```

```
uses Objects, Drivers, Views;
```

```

const
  CHelpColor      = #$37#$3F#$3A#$13#$13#$30#$3E#$1E;
  CHelpBlackWhite = #$07#$0F#$07#$70#$70#$07#$0F#$70;
  CHelpMonochrome = #$07#$0F#$07#$70#$70#$07#$0F#$70;
  CHelpViewer     = #6#7#8;
  CHelpWindow     = #128#129#130#131#132#133#134#135;

```

```
type
```

```
( TParagraph )
```

```

  PParagraph = ^TParagraph;
  TParagraph = record
    Next: PParagraph;
    Wrap: Boolean;
    Size: Word;
    Text: record end;
  end;

```

```
( THelpTopic )
```

```

  TCrossRef = record
    Ref: Word;
    Offset: Integer;
    Length: Byte;
  end;

```

```

  PCrossRefs = ^TCrossRefs;
  TCrossRefs = array[1..10000] of TCrossRef;

```

```

  TCrossRefHandler = procedure (var S: TStream; XRefValue: Integer);
  PHelpTopic = ^THelpTopic;
  THelpTopic = object(TObject)
    constructor Init;
    constructor Load(var S: TStream);
    destructor Done; virtual;
    procedure AddCrossRef(Ref: TCrossRef);
    procedure AddParagraph(P: PParagraph);
    procedure GetCrossRef(I: Integer; var Loc: TPoint; var Length: Byte;
      var Ref: Word);
    function GetLine(Line: Integer): String;
    function GetNumCrossRefs: Integer;
    function NumLines: Integer;
    procedure SetCrossRef(I: Integer; var Ref: TCrossRef);
    procedure SetNumCrossRefs(I: Integer);
    procedure SetWidth(AWidth: Integer);
    procedure Store(var S: TStream);
  private
    Paragraphs: PParagraph;
    NumRefs: Integer;
    CrossRefs: PCrossRefs;
    Width: Integer;
    LastOffset: Integer;
    LastLine: Integer;
    LastParagraph: PParagraph;
    function WrapText(var Text; Size: Integer; var Offset: Integer;
      Wrap: Boolean): String;
  end;

```

```
private
```

```

  Paragraphs: PParagraph;
  NumRefs: Integer;
  CrossRefs: PCrossRefs;
  Width: Integer;
  LastOffset: Integer;
  LastLine: Integer;
  LastParagraph: PParagraph;
  function WrapText(var Text; Size: Integer; var Offset: Integer;
    Wrap: Boolean): String;
end;

```

```
( THelpIndex )
```

```

  PIndexArray = ^TIndexArray;
  TIndexArray = array[0..16380] of Longint;

  PContextArray = ^TContextArray;
  TContextArray = array[0..16380] of Word;

```

```

  PHelpIndex = ^THelpIndex;
  THelpIndex = object(TObject)
    constructor Init;
    constructor Load(var S: TStream);
    destructor Done; virtual;
    function Position(I: Word): Longint;
    procedure Add(I: Word; Val: Longint);
    procedure Store(var S: TStream);
  private
    Size: Word;
    Used: Word;
    Contexts: PContextArray;
    Index: PIndexArray;
    function Find(I: Word): Word;
  end;

```

```
( THelpFile )
```

```

  PHelpFile = ^THelpFile;
  THelpFile = object(TObject)
    Stream: PStream;
    Modified: Boolean;

```

```

    constructor Init(S: PStream);
    destructor Done; virtual;
    function GetTopic(I: Word): PHelpTopic;
    function InvalidTopic: PHelpTopic;
    procedure RecordPositionInIndex(I: Integer);
    procedure PutTopic(Topic: PHelpTopic);
private
    Index: PHelpIndex;
    IndexPos: LongInt;
end;

{ THelpViewer }

PHelpViewer = ^THelpViewer;
THelpViewer = object(TScroller)
    HFile: PHelpFile;
    Topic: PHelpTopic;
    Selected: Integer;
    constructor Init(var Bounds: TRect; AHScrollBar,
        AVScrollBar: PScrollBar; AHelpFile: PHelpFile; Context: Word);
    destructor Done; virtual;
    procedure ChangeBounds(var Bounds: TRect); virtual;
    procedure Draw; virtual;
    function GetPalette: PPalette; virtual;
    procedure HandleEvent(var Event: TEvent); virtual;
end;

{ THelpWindow }

PHelpWindow = ^THelpWindow;
THelpWindow = object(TWindow)
    constructor Init(HFile: PHelpFile; Context: Word);
    function GetPalette: PPalette; virtual;
end;

const
    RHelpTopic: TStreamRec = (
        ObjType: 10000;
        VmtLink: ofs(TypeOf(THelpTopic)^);
        Load: @THelpTopic.Load;
        Store: @THelpTopic.Store
    );

const
    RHelpIndex: TStreamRec = (
        ObjType: 10001;
        VmtLink: ofs(TypeOf(THelpIndex)^);
        Load: @THelpIndex.Load;
        Store: @THelpIndex.Store
    );

procedure RegisterHelpFile;

procedure NotAssigned(var S: TStream; Value: Integer);

const
    CrossRefHandler: TCrossRefHandler = NotAssigned;

implementation

```

```

{ THelpTopic }

constructor THelpTopic.Init;
begin
    inherited Init;
    LastLine := MaxInt;
end;

constructor THelpTopic.Load(var S: TStream);

procedure ReadParagraphs;
var
    I, Size: Integer;
    PP: ^PParagraph;
begin
    S.Read(I, SizeOf(I));
    PP := @Paragraphs;
    while I > 0 do
    begin
        S.Read(Size, SizeOf(Size));
        GetMem(PP^, SizeOf(PP^) + Size);
        PP^.Size := Size;
        S.Read(PP^.Wrap, SizeOf(Boolean));
        S.Read(PP^.Text, Size);
        PP := @PP^.Next;
        Dec(I);
    end;
    PP^ := nil;
end;

procedure ReadCrossRefs;
begin
    S.Read(NumRefs, SizeOf(Integer));
    GetMem(CrossRefs, SizeOf(TCrossRef) * NumRefs);
    if CrossRefs <> nil then
        S.Read(CrossRefs^, SizeOf(TCrossRef) * NumRefs);
end;

begin
    ReadParagraphs;
    ReadCrossRefs;
    Width := 0;
    LastLine := MaxInt;
end;

destructor THelpTopic.Done;

procedure DisposeParagraphs;
var
    P, T: PParagraph;
begin
    P := Paragraphs;
    while P <> nil do
    begin
        T := P;
        P := P^.Next;
        FreeMem(T, SizeOf(T) + T^.Size);
    end;
end;

```

```

begin
  DisposeParagraphs;
  FreeMem(CrossRefs, SizeOf(TCrossRef) * NumRefs);
  inherited Done
end;

procedure THelpTopic.AddCrossRef(Ref: TCrossRef);
var
  P: PCrossRefs;
begin
  GetMem(P, (NumRefs+1) * SizeOf(TCrossRef));
  if NumRefs > 0 then
  begin
    Move(CrossRefs^, P^, NumRefs * SizeOf(TCrossRef));
    FreeMem(CrossRefs, NumRefs * SizeOf(TCrossRef));
  end;
  CrossRefs := P;
  CrossRefs^[NumRefs] := Ref;
  Inc(NumRefs);
end;

procedure THelpTopic.AddParagraph(P: PParagraph);
var
  PP: ^PParagraph;
begin
  PP := @Paragraphs;
  while PP^ <> nil do
    PP := @PP^.Next;
  PP^ := P;
  P^.Next := nil;
end;

procedure THelpTopic.GetCrossRef(I: Integer; var Loc: TPoint;
  var Length: Byte; var Ref: Word);
var
  OldOffset, CurOffset, Offset, ParaOffset: Integer;
  P: PParagraph;
  Line: Integer;
begin
  ParaOffset := 0;
  CurOffset := 0;
  OldOffset := 0;
  Line := 0;
  Offset := CrossRefs^[I].Offset;
  P := Paragraphs;
  while ParaOffset+CurOffset < Offset do
  begin
    OldOffset := ParaOffset + CurOffset;
    WrapText(P^.Text, P^.Size, CurOffset, P^.Wrap);
    Inc(Line);
    if CurOffset >= P^.Size then
    begin
      Inc(ParaOffset, P^.Size);
      P := P^.Next;
      CurOffset := 0;
    end;
  end;
  Loc.X := Offset - OldOffset - 1;
  Loc.Y := Line;
  Length := CrossRefs^[I].Length;

```

```

  Ref := CrossRefs^[I].Ref;
end;

function THelpTopic.GetLine(Line: Integer): String;
var
  Offset, I: Integer;
  P: PParagraph;
begin
  if LastLine < Line then
  begin
    I := Line;
    Dec(Line, LastLine);
    LastLine := I;
    Offset := LastOffset;
    P := LastParagraph;
  end
  else
  begin
    P := Paragraphs;
    Offset := 0;
    LastLine := Line;
  end;
  GetLine := '';
  while (P <> nil) do
  begin
    while Offset < P^.Size do
    begin
      Dec(Line);
      GetLine := WrapText(P^.Text, P^.Size, Offset, P^.Wrap);
      if Line = 0 then
      begin
        LastOffset := Offset;
        LastParagraph := P;
        Exit;
      end;
    end;
    P := P^.Next;
    Offset := 0;
  end;
  GetLine := '';
end;

function THelpTopic.GetNumCrossRefs: Integer;
begin
  GetNumCrossRefs := NumRefs;
end;

function THelpTopic.NumLines: Integer;
var
  Offset, Lines: Integer;
  P: PParagraph;
begin
  Offset := 0;
  Lines := 0;
  P := Paragraphs;
  while P <> nil do
  begin
    Offset := 0;
    while Offset < P^.Size do
    begin

```



```

    Inc(Lines);
    WrapText(P^.Text, P^.Size, Offset, P^.Wrap);
end;
P := P^.Next;
end;
NumLines := Lines;
end;

procedure THelpTopic.SetCrossRef(I: Integer; var Ref: TCrossRef);
begin
    if I <= NumRefs then CrossRefs^[I] := Ref;
end;

procedure THelpTopic.SetNumCrossRefs(I: Integer);
var
    P: PCrossRefs;
begin
    if NumRefs = I then Exit;
    GetMem(P, I * SizeOf(TCrossRef));
    if NumRefs > 0 then
    begin
        if I > NumRefs then Move(CrossRefs^, P^, NumRefs * SizeOf(TCrossRef))
        else Move(CrossRefs^, P^, I * SizeOf(TCrossRef));
        FreeMem(CrossRefs, NumRefs * SizeOf(TCrossRef));
    end;
    CrossRefs := P;
    NumRefs := I;
end;

procedure THelpTopic.SetWidth(AWidth: Integer);
begin
    Width := AWidth;
end;

procedure THelpTopic.Store(var S: TStream);

procedure WriteParagraphs;
var
    I: Integer;
    P: PParagraph;
begin
    P := Paragraphs;
    I := 0;
    while P <> nil do
    begin
        Inc(I);
        P := P^.Next;
    end;
    S.Write(I, SizeOf(I));
    P := Paragraphs;
    while P <> nil do
    begin
        S.Write(P^.Size, SizeOf(Integer));
        S.Write(P^.Wrap, SizeOf(Boolean));
        S.Write(P^.Text, P^.Size);
        P := P^.Next;
    end;
end;

procedure WriteCrossRefs;

```

```

var
    I: Integer;
begin
    S.Write(NumRefs, SizeOf(Integer));
    if @CrossRefHandler = @NotAssigned then
        S.Write(CrossRefs^, SizeOf(TCrossRef) * NumRefs)
    else
        for I := 1 to NumRefs do
        begin
            CrossRefHandler(S, CrossRefs^[I].Ref);
            S.Write(CrossRefs^[I].Offset, SizeOf(Integer) + SizeOf(Byte));
        end;
    end;

begin
    WriteParagraphs;
    WriteCrossRefs;
end;

function THelpTopic.WrapText(var Text; Size: Integer;
    var Offset: Integer; Wrap: Boolean): String;
type
    PCharArray = ^CArray;
    CArray = array[0..32767] of Char;
var
    Line: String;
    I, P: Integer;

function IsBlank(Ch: Char): Boolean;
begin
    IsBlank := (Ch = ' ') or (Ch = #13) or (Ch = #10);
end;

function Scan(var P; Offset, Size: Integer; C: Char): Integer; assembler;
asm
    CLD
    LES     DI, P
    ADD     DI, &Offset
    MOV     DX, Size
    SUB     DX, &Offset
    OR      DH, DH
    JZ      @a1
    MOV     DX, 256
@a1:      MOV     CX, DX
    MOV     AL, C
    REPNE   SCASB
    SUB     CX, DX
    NEG     CX
    XCHG    AX, CX
end;

procedure TextToLine(var Text; Offset, Length: Integer; var Line: String);
    assembler;
asm
    CLD
    PUSH    DS
    LDS     SI, Text
    ADD     SI, &Offset
    LES     DI, Line
    MOV     AX, Length

```

```

        STOSB
        XCHG  AX,CX
        REP   MOVSB
        POP   DS
end;

begin
  I := Scan(Text, Offset, Size, #13);
  if (I >= Width) and Wrap then
  begin
    I := Offset + Width;
    if I > Size then I := Size
    else
    begin
      while (I > Offset) and not IsBlank(PCArray[@Text]^I) do Dec(I);
      if I = Offset then I := Offset + Width
      else Inc(I);
    end;
    if I = Offset then I := Offset + Width;
    Dec(I, Offset);
  end;
  TextToLine(Text, Offset, I, Line);
  if Line[Length(Line)] = #13 then Dec(Line[0]);
  Inc(Offset, I);
  WrapText := Line;
end;

( THelpIndex )

constructor THelpIndex.Init;
begin
  inherited Init;
  Size := 0;
  Contexts := nil;
  Index := nil;
end;

constructor THelpIndex.Load(var S: TStream);
begin
  S.Read(Used, SizeOf(Used));
  S.Read(Size, SizeOf(Size));
  if Size = 0 then
  begin
    Contexts := nil;
    Index := nil;
  end
  else
  begin
    GetMem(Contexts, SizeOf(Contexts^[0]) * Size);
    S.Read(Contexts^, SizeOf(Contexts^[0]) * Size);
    GetMem(Index, SizeOf(Index^[0]) * Size);
    S.Read(Index^, SizeOf(Index^[0]) * Size);
  end;
end;

destructor THelpIndex.Done;
begin
  FreeMem(Index, SizeOf(Index^[0]) * Size);
  FreeMem(Contexts, SizeOf(Contexts^[0]) * Size);
  inherited Done;
end;

```

```

end;

function THelpIndex.Find(I: Word): Word;
var
  Hi, Lo, Pos: Integer;
begin
  Lo := 0;
  if Used > 0 then
  begin
    Hi := Used - 1;
    while Lo <= Hi do
    begin
      Pos := (Lo + Hi) div 2;
      if I > Contexts^[Pos] then
        Lo := Pos + 1
      else
      begin
        Hi := Pos - 1;
        if I = Contexts^[Pos] then
          Lo := Pos;
        end;
      end;
    end;
    Find := Lo;
  end;
end;

function THelpIndex.Position(I: Word): Longint;
begin
  Position := Index^[Find(I)];
end;

procedure THelpIndex.Add(I: Word; Val: Longint);
const
  Delta = 10;
var
  P: PIndexArray;
  NewSize: Integer;
  Pos: Integer;

  function Grow(P: Pointer; OldSize, NewSize, ElemSize: Integer): Pointer;
  var
    NewP: PByteArray;
  begin
    GetMem(NewP, NewSize * ElemSize);
    if NewP <> nil then
    begin
      if P <> nil then
        Move(P^, NewP^, OldSize * ElemSize);
      FillChar(NewP^[OldSize * ElemSize], (NewSize - OldSize) * ElemSize, $FF);
    end;
    if OldSize > 0 then FreeMem(P, OldSize * ElemSize);
    Grow := NewP;
  end;

begin
  Pos := Find(I);
  if (Contexts = nil) or (Contexts^[Pos] <> I) then
  begin
    Inc(Used);
    if Used >= Size then

```

```

begin
  NewSize := (Used + Delta) div Delta * Delta;
  Contexts := Grow(Contexts, Size, NewSize, SizeOf(Contexts^[0]));
  Index := Grow(Index, Size, NewSize, SizeOf(Index^[0]));
  Size := NewSize;
end;
if Pos < Used then
begin
  Move(Contexts^[Pos], Contexts^[Pos + 1], (Used - Pos - 1) *
    SizeOf(Contexts^[0]));
  Move(Index^[Pos], Index^[Pos + 1], (Used - Pos - 1) *
    SizeOf(Index^[0]));
end;
end;
Contexts^[Pos] := I;
Index^[Pos] := Val;
end;

procedure THelpIndex.Store(var S: TStream);
begin
  S.Write(Used, SizeOf(Used));
  S.Write(Size, SizeOf(Size));
  S.Write(Contexts^, SizeOf(Contexts^[0]) * Size);
  S.Write(Index^, SizeOf(Index^[0]) * Size);
end;

( THelpFile )

const
  MagicHeader = $46484246; ('FBHF')

constructor THelpFile.Init(S: PStream);
var
  Magic: Longint;
begin
  Magic := 0;
  S.Seek(0);
  if S.GetSize > SizeOf(Magic) then
    S.Read(Magic, SizeOf(Magic));
  if Magic <> MagicHeader then
    begin
      IndexPos := 12;
      S.Seek(IndexPos);
      Index := New(PHelpIndex, Init);
      Modified := True;
    end
  else
    begin
      S.Seek(8);
      S.Read(IndexPos, SizeOf(IndexPos));
      S.Seek(IndexPos);
      Index := PHelpIndex(S.Get);
      Modified := False;
    end;
  Stream := S;
end;

destructor THelpFile.Done;
var
  Magic, Size: Longint;

```

```

begin
  if Modified then
    begin
      Stream.Seek(IndexPos);
      Stream.Put(Index);
      Stream.Seek(0);
      Magic := MagicHeader;
      Size := Stream.GetSize - 8;
      Stream.Write(Magic, SizeOf(Magic));
      Stream.Write(Size, SizeOf(Size));
      Stream.Write(IndexPos, SizeOf(IndexPos));
    end;
  Dispose(Stream, Done);
  Dispose(Index, Done);
end;

function THelpFile.GetTopic(I: Word): PHelpTopic;
var
  Pos: Longint;
begin
  Pos := Index.Position(I);
  if Pos > 0 then
    begin
      Stream.Seek(Pos);
      GetTopic := PHelpTopic(Stream.Get);
    end
  else
    GetTopic := InvalidTopic;
  end;
end;

function THelpFile.InvalidTopic: PHelpTopic;
var
  Topic: PHelpTopic;
  Para: PParagraph;
const
  InvalidStr = #13' No help available in this context.';
  InvalidText: array[1..Length(InvalidStr)] of Char = InvalidStr;
begin
  Topic := New(PHelpTopic, Init);
  GetMem(Para, SizeOf(Para) + SizeOf(InvalidText));
  Para.Size := SizeOf(InvalidText);
  Para.Wrap := False;
  Para.Next := nil;
  Move(InvalidText, Para.Text, SizeOf(InvalidText));
  Topic.AddParagraph(Para);
  InvalidTopic := Topic;
end;

procedure THelpFile.RecordPositionInIndex(I: Integer);
begin
  Index.Add(I, IndexPos);
  Modified := True;
end;

procedure THelpFile.PutTopic(Topic: PHelpTopic);
begin
  Stream.Seek(IndexPos);
  Stream.Put(Topic);
  IndexPos := Stream.GetPos;
  Modified := True;
end;

```

```
{ THelpViewer }
```

```
constructor THelpViewer.Init(var Bounds: TRect; AHScrollBar,
  AVScrollBar: PScrollBar; AHelpFile: PHelpFile; Context: Word);
begin
```

```
  inherited Init(Bounds, AHScrollBar, AVScrollBar);
  Options := Options or ofSelectable;
  GrowMode := gfGrowHiX + gfGrowHiY;
  HFile := AHelpFile;
  Topic := AHelpFile^.GetTopic(Context);
  Topic^.SetWidth(Size.X);
  SetLimit(78, Topic^.NumLines);
  Selected := 1;
end;
```

```
destructor THelpViewer.Done;
```

```
begin
  inherited Done;
  Dispose(HFile, Done);
  Dispose(Topic, Done);
end;
```

```
procedure THelpViewer.ChangeBounds(var Bounds: TRect);
```

```
begin
  inherited ChangeBounds(Bounds);
  Topic^.SetWidth(Size.X);
  SetLimit(Limit.X, Topic^.NumLines);
end;
```

```
procedure THelpViewer.Draw;
```

```
var
  B: TDrawBuffer;
  Line: String;
  I, J, L: Integer;
  KeyCount: Integer;
  Normal, Keyword, SelKeyword, C: Byte;
  KeyPoint: TPoint;
  KeyLength: Byte;
  KeyRef: Word;
begin
  Normal := GetColor(1);
  Keyword := GetColor(2);
  SelKeyword := GetColor(3);
  KeyCount := 0;
  KeyPoint.X := 0;
  KeyPoint.Y := 0;
  Topic^.SetWidth(Size.X);
  if Topic^.GetNumCrossRefs > 0 then
    repeat
      Inc(KeyCount);
      Topic^.GetCrossRef(KeyCount, KeyPoint, KeyLength, KeyRef);
    until (KeyCount >= Topic^.GetNumCrossRefs) or (KeyPoint.Y > Delta.Y);
  for I := 1 to Size.Y do
    begin
      MoveChar(B, ' ', Normal, Size.X);
      Line := Topic^.GetLine(I + Delta.Y);
      MoveStr(B, Copy(Line, Delta.X+1, Size.X), Normal);
      while I + Delta.Y = KeyPoint.Y do
        begin
```

```
      L := KeyLength;
      if KeyPoint.X < Delta.X then
        begin
          Dec(L, Delta.X - KeyPoint.X);
          KeyPoint.X := Delta.X;
        end;
      if KeyCount = Selected then C := SelKeyword
      else C := Keyword;
      for J := 0 to L-1 do
        WordRec(B[KeyPoint.X - Delta.X + J]).Hi := C;
      Inc(KeyCount);
      if KeyCount <= Topic^.GetNumCrossRefs then
        Topic^.GetCrossRef(KeyCount, KeyPoint, KeyLength, KeyRef)
      else KeyPoint.Y := 0;
      end;
      WriteLine(0, I-1, Size.X, 1, B);
    end;
  end;
```

```
function THelpViewer.GetPalette: PPalette;
```

```
const
  P: String[Length(CHelpViewer)] = CHelpViewer;
begin
  GetPalette := @P;
end;
```

```
procedure THelpViewer.HandleEvent(var Event: TEvent);
```

```
var
  KeyPoint, Mouse: TPoint;
  KeyLength: Byte;
  KeyRef: Word;
  KeyCount: Integer;
```

```
procedure MakeSelectVisible;
```

```
var
  D: TPoint;
begin
  Topic^.GetCrossRef(Selected, KeyPoint, KeyLength, KeyRef);
  D := Delta;
  if KeyPoint.X < D.X then D.X := KeyPoint.X
  else if KeyPoint.X + KeyLength > D.X + Size.X then
    D.X := KeyPoint.X + KeyLength - Size.X + 1;
  if KeyPoint.Y <= D.Y then D.Y := KeyPoint.Y - 1;
  if KeyPoint.Y > D.Y + Size.Y then D.Y := KeyPoint.Y - Size.Y;
  if (D.X <> Delta.X) or (D.Y <> Delta.Y) then ScrollTo(D.X, D.Y);
end;
```

```
procedure SwitchToTopic(KeyRef: Integer);
```

```
begin
  if Topic <> nil then Dispose(Topic, Done);
  Topic := HFile^.GetTopic(KeyRef);
  Topic^.SetWidth(Size.X);
  ScrollTo(0, 0);
  SetLimit(Limit.X, Topic^.NumLines);
  Selected := 1;
  DrawView;
end;
```

```
begin
  inherited HandleEvent(Event);
```

```

case Event.What of
  evKeyDown:
    begin
      case Event.KeyCode of
        kbUp:
          if Topic^.GetNumCrossRefs > 0 then
            begin
              Selected := Selected - 1;
              if Selected < 1 then Selected := Topic^.GetNumCrossRefs;
              MakeSelectVisible;
            end;
          kbDown:
            if Topic^.GetNumCrossRefs > 0 then
              begin
                Inc(Selected);
                if Selected > Topic^.GetNumCrossRefs then Selected := 1;
                MakeSelectVisible;
              end;
          kbTab:
            if Topic^.GetNumCrossRefs > 0 then
              begin
                Inc(Selected);
                if Selected > Topic^.GetNumCrossRefs then Selected := 1;
                MakeSelectVisible;
              end;
          kbShiftTab:
            if Topic^.GetNumCrossRefs > 0 then
              begin
                Dec(Selected);
                if Selected = 0 then Selected := Topic^.GetNumCrossRefs;
                MakeSelectVisible;
              end;
          kbEnter:
            if Selected <= Topic^.GetNumCrossRefs then
              begin
                Topic^.GetCrossRef(Selected, KeyPoint, KeyLength, KeyRef);
                SwitchToTopic(KeyRef);
              end;
          kbEsc:
            begin
              Event.What := evCommand;
              Event.Command := cmClose;
              PutEvent(Event);
            end;
        else
          Exit;
        end;
      DrawView;
      ClearEvent(Event);
    end;
  evMouseDown:
    begin
      MakeLocal(Event.Where, Mouse);
      Inc(Mouse.X, Delta.X); Inc(Mouse.Y, Delta.Y);
      KeyCount := 0;
      repeat
        Inc(KeyCount);
        if KeyCount > Topic^.GetNumCrossRefs then Exit;
        Topic^.GetCrossRef(KeyCount, KeyPoint, KeyLength, KeyRef);
      until (KeyPoint.Y = Mouse.Y+1) and (Mouse.X >= KeyPoint.X) and

```

```

      (Mouse.X < KeyPoint.X + KeyLength);
      Selected := KeyCount;
      DrawView;
      if Event.Double then SwitchToTopic(KeyRef);
      ClearEvent(Event);
    end;
  evCommand:
    if (Event.Command = cmClose) and (Owner^.State and sfModal <> 0) then
      begin
        EndModal(cmClose);
        ClearEvent(Event);
      end;
    end;
  end;
end;

( THelpWindow )

constructor THelpWindow.Init(HFile: PHelpFile; Context: Word);
var
  R: TRect;
begin
  R.Assign(0, 0, 50, 18);
  TWindow.Init(R, 'Help', wnNoNumber);
  Options := Options or ofCentered;
  R.Grow(-2, -1);
  Insert(New(PHelpViewer, Init(R,
    StandardScrollBar(sbHorizontal + sbHandleKeyboard),
    StandardScrollBar(sbVertical + sbHandleKeyboard), HFile, Context)));
end;

function THelpWindow.GetPalette: PPalette;
const
  P: String[Length(CHelpWindow)] = CHelpWindow;
begin
  GetPalette := @P;
end;

procedure RegisterHelpFile;
begin
  RegisterType(RHelpTopic);
  RegisterType(RHelpIndex);
end;

procedure NotAssigned(var S: TStream; Value: Integer);
begin
  end;
end.

```

```

{*****}
{
{   Unit for operating the overall system   }
{
{   Filename : Control.pas   Author K G Mercer   }
{   Date      : September 1993   }
{*****}
{$M 4000,0,0}
program Install;

Uses Dos,Crt,WinDos,Strings;
label
    Restart;

var
    x,y      : Integer;
    Drive    : Byte;
    DriveLetter : Char;
    Ques     : Char;
    OK       : Boolean;
    Size     : Byte;
    Dir      : PChar;
    S        : String;

procedure box(x1,y1,x2,y2:integer;title:string;color1,color2:byte);
var i:integer;
begin
    textbackground(black);
    window(x1+2,y1+1,x2+2,y2+1);
    clrscr;
    textbackground(color1);
    textcolor(color2);
    window(x1,y1,x2,y2);
    clrscr;
    gotoxy(2,1);
    write(chr(218));
    for i:= 1 to (x2-x1-3) do
        write(chr(196));
    write(chr(191));
    for i := 1 to (y2-y1-1) do
        begin
            gotoxy(2,i+1);
            write(chr(179));
            gotoxy(x2-x1,i+1);
            write(chr(179));
        end;
    gotoxy(2,y2-y1+1);
    write(chr(192));
    for i := 1 to (x2-x1-3) do
        write(chr(196));
    write(chr(217));
    if not(title='') then
        begin
            gotoxy((round((x2-x1)/2-(length(title)/2))+1),1);
            write(title);
        end;
    end;
end;

Procedure NewScreen;

```

```

begin
    window(1,1,80,25);
    textbackground(black);
    textcolor(black);
    clrscr;
    textcolor(cyan);
    clrscr;
    for y:= 1 to 25 do
        begin
            for x:= 1 to 80 do
                begin
                    gotoxy(x,y);
                    write(chr(176));
                end;
            end;
        end;
end;

Begin
{*****}
{                               Screen One                               }
{*****}
Restart:
NewScreen;
box(10,2,70,4,'',blue,yellow);
gotoxy(20,2);
writeln('Installation Program');

box(15,7,65,16,'',blue,lightcyan);
gotoxy(5,2);
writeln('Input drive destination :');
gotoxy(33,2);
writeln('(C/D/E etc.)');
gotoxy(34,3);
writeln('Q - Quit');
gotoxy(31,2);

OK:=false;
repeat
begin
    DriveLetter := ReadKey;
    Case Upcase(DriveLetter) of
        'A','B','C','D','E','F','G','H','Q' : OK := true;
    end;
end;
until OK;
gotoxy(31,2);
write(UpCase(DriveLetter));
if UpCase(DriveLetter) = 'Q' then Exit;
DriveLetter:=UpCase(DriveLetter);

Case Upcase(DriveLetter) of
    'A' : Drive := 1;
    'B' : Drive := 2;
    'C' : Drive := 3;
    'D' : Drive := 4;
    'E' : Drive := 5;
    'F' : Drive := 6;
    'G' : Drive := 7;
    'H' : Drive := 8;

```

```

end;
if DiskSize(Drive) = -1 then
begin
  textcolor(red);
  gotoxy(5,5);
  writeln('Invalid Drive !');
  delay(4000);
  Goto Restart;
end
else
begin
  textcolor(lightgreen);
  gotoxy(5,5);
  writeln('Selected Drive Size : ',DiskSize(Drive));
  gotoxy(41,5);
  writeln('Bytes');
  gotoxy(5,6);
  writeln('Available Disk Space : ',DiskFree(Drive));
  gotoxy(41,6);
  write('Bytes');
  textcolor(lightgreen);
  gotoxy(5,8);
  writeln('Creating Directory : ',DriveLetter,':\ESEA');
end;

box(10,19,70,22,'',blue,lightcyan);
gotoxy(10,2);
write('Do you wish to continue installation ? (y/n)');
OK:=false;
repeat
begin
  Ques := ReadKey;
  Case Uppcase(Ques) of
    'Y','N' : OK := true;
  end;
end;
until OK;
if UpCase(Ques)='N' then
begin
  textbackground(black);
  window(1,1,80,25);
  clrscr;
  delay(2000);
  exit;
end;
s:=DriveLetter+':\ESEA';
StrPCopy(Dir,s);
CreateDir(Dir);

```

```

{*****}
{      Screen Two      }
{*****}
NewScreen;
box(10,5,70,10,'',blue,yellow);
gotoxy(20,3);
writeln('Copying Expert System Files');
delay(1000);
textbackground(blue);

```

```

textcolor(cyan);
window(1,1,80,25);
clrscr;
Exec(GetEnv('COMSPEC'), '/C pkunzip esystem.zip '+s);

{*****}
{      Screen Three      }
{*****}
NewScreen;
box(10,5,70,10,'',blue,yellow);
gotoxy(20,3);
writeln('Copying Rule Base Files');
delay(1000);
textbackground(blue);
textcolor(cyan);
window(1,1,80,25);
clrscr;
Exec(GetEnv('COMSPEC'), '/C pkunzip rbases.zip '+s);

{*****}
{      Screen Four      }
{*****}
NewScreen;
box(10,5,70,10,'',blue,yellow);
gotoxy(20,3);
writeln('Copying Utility Files');
delay(1000);
textbackground(blue);
textcolor(cyan);
window(1,1,80,25);
clrscr;
Exec(GetEnv('COMSPEC'), '/C pkunzip progs.zip '+s);

textbackground(black);
window(1,1,80,25);
clrscr;
delay(2000);
End.

```

```

(*****)
(
(   Filename : ListADlg.pas   Author K G Mercer )
(   Date      : September 1993 )
(
(   Objects   : TRecD4
(               : TDataCollection4
(               : TPickListKeyBox
(               : TListADialog
(               : TListA2Dialog
(               : PRBaseInfoPane
(
(   Other      : PInteger      - integer ptr
(               : TransferWithCode - record
(               : TRecD23      - record
(
(*****)

```

```
unit ListADlg;
```

```
($F+,O+,X+,S-,D+)
```

```
interface
```

```
uses
```

```
  Dos, Objects, Memory, Drivers, Views, Dialogs, StdDlg, CmdnList,
  General1;
```

```
type
```

```
  PInteger = ^Integer;
```

```
  PRecD4 = ^TRecD4;
```

```
  TRecD4 = object(TObject)
```

```
    Name1 : PString;
```

```
    Descrip1 : PString;
```

```
    Descrip2 : PString;
```

```
    Descrip3 : PString;
```

```
    PCode : PInteger;
```

```
    constructor Init(Name,Des1,Des2,Des3: String; NewCode : Integer);
```

```
    destructor Done; virtual;
```

```
  end;
```

```
  PDataCollection4 = ^TDataCollection4;
```

```
  TDataCollection4 = object(TSortedCollection)
```

```
    function KeyOf(Item: Pointer): Pointer; virtual;
```

```
    function Compare(Key1, Key2: Pointer): Integer; virtual;
```

```
  end;
```

```
  PPickListKeyBox = ^TPickListKeyBox;
```

```
  TPickListKeyBox = object(TSortedListBox)
```

```
    function GetText(Item: Integer; maxlength:integer): String; virtual;
```

```
    procedure GetData(var Rec);virtual;
```

```
    function DataSize:Word;virtual;
```

```
    procedure FocusItem(Item: Integer);virtual;
```

```
    procedure SendFirst;
```

```
  end;
```

```
  PListADialog = ^TListADialog;
```

```
  TListADialog = object(TDialog)
```

```
    DataCollection4 : PDataCollection4;
```

```

List      : PPickListKeyBox;
IsValid   : Boolean;
Modified  : Boolean;
constructor Init(Collection:PDataCollection4);
destructor Done; virtual;
procedure Close; virtual;
procedure HandleEvent(var Event: TEvent); virtual;
end;

```

```
PListA2Dialog = ^TListA2Dialog;
```

```
TListA2Dialog = object(TListADialog)
```

```
  constructor Init(Collection:PDataCollection4);
```

```
end;
```

```

( TRBaseInfoPane is a TView that displays the information
( about the currently selected rulebase responding to broadcast )
( message 'cmFileFocussed' from PickListKeyBox )

```

```
PRBaseInfoPane = ^TRBaseInfoPane;
```

```
TRBaseInfoPane = object(TView)
```

```
  Info,Info2 : String;
```

```
  constructor Init(var Bounds:TRect);
```

```
  procedure Draw; virtual;
```

```
  function GetPalette: PPalette; virtual;
```

```
  procedure HandleEvent(var Event: TEvent); virtual;
```

```
  destructor Done; virtual;
```

```
end;
```

```
TransferWithCode = record
```

```
  Listname : PDataCollection4;
```

```
  Listfocused : Integer;
```

```
  RecCode : Integer;
```

```
end;
```

```
TRecD23 = record
```

```
  Name : String;
```

```
end;
```

```
function FileExists(Name: PathStr): Boolean;
```

```
const
```

```
  CInfoPane = #30;
```

```
(*****)
```

```
implementation
```

```
(*****)
```

```
uses App, MsgBox;
```

```
function FileExists(Name: PathStr): Boolean;
```

```
var
```

```
  SR: SearchRec;
```

```
begin
```

```
  FindFirst(Name, 0, SR);
```

```
  FileExists := DosError = 0;
```

```
end;
```

```
(*****)
```

```
(* TListA2Dialog *)
```

```
(*****)
```



```

{ TListA2Dialog }
(*****)
(*          Init          *)
(*****)
constructor TListA2Dialog.Init(Collection:PDataCollection4);
const
  FormX = 16;      { form x position }
  FormY = 2;       { form y position }
  FormWd = 46;     { form width }
  FormHt = 19;     { form length }
  ListX = 1;
  ListY = 5;
  ListHt = 6;
  ListWd = 43;

var
  R      : TRect;
  SB     : PScrollBar;
  Y      : Integer;
  D      : DirStr;
  N      : String;
  E      : ExtStr;
  TitleStr : String[50];
  Bruce  : PView;
  ButtonX1,ButtonX2,ButtonWd : Integer;
  ButtonX3 : Integer;

begin
  R.Assign(FormX, FormY, FormX + FormWd, FormY + FormHt);
  N:='Expert System Selection';
  TDialog.Init(R, N);

  { Read data collection into memory }

  DataCollection4 := Collection;
  if DataCollection4 <> nil then
  begin
    { Loaded successfully: build ListADialog dialog }

    { Scrollbar }
    R.Assign(ListX + ListWd, ListY, ListX + ListWd + 1, ListY + ListHt);
    SB := New(PScrollBar, Init(R));
    Insert(SB);

    { List box }
    R.Assign(ListX, ListY, ListX + ListWd, ListY + ListHt);
    List := New(PPickListKeyBox, Init(R, 1, SB));
    List^.NewList(DataCollection4);
    Insert(List);

    { Label }
    R.Assign(1,2,40,3);
    Insert(New(PLabel, Init(R,'Select Expert System Rulebase to Run', Bruce)));

    TitleStr:='RuleBase Name';
    R.Assign(1,4,20,5);
    Insert(New(PLabel, Init(R, TitleStr, Bruce)));

    { Label2 }

```

```

  R.Assign(1,12,25,13);
  Insert(New(PLabel, Init(R, 'Rulebase Description', Bruce)));

  { RBaseInfoPane }
  R.Assign(2,13,44,15);
  Insert(New(PRBaseInfoPane, Init(R)));

  { Buttons }
  Y:=16;
  ButtonX1 := 2;
  ButtonX2 := 16;
  ButtonX3 := 31;
  ButtonWd := 12;
  R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~C-ancel', cmCancel, bfNormal)));

  R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~H-elp', cmHelp, bfNormal)));

  R.Assign(ButtonX3, Y, ButtonX3 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~S-elect', cmOK, bfDefault)));

  SelectNext(False);      { Select first field }
  IsValid := True;
  List^.SendFirst;

end;
end;

(*****)
(*          TRecD4          *)
(*****)
constructor TRecD4.Init(Name,Des1,Des2,Des3 : String; NewCode : Integer);
begin
  Name1 := NewStr(Name);
  Descrip1 := NewStr(Des1);
  Descrip2 := NewStr(Des2);
  Descrip3 := NewStr(Des3);
  New(PCode);
  PCode^ := NewCode;
end;

destructor TRecD4.Done;
begin
  DisposeStr(Name1);
  DisposeStr(Descrip1);
  DisposeStr(Descrip2);
  DisposeStr(Descrip3);
  Dispose(PCode);
end;

(*****)
(*          TDataCollection4          *)
(*****)

function TDataCollection4.KeyOf(Item: Pointer): Pointer;
begin
  KeyOf := PRecD4(Item)^.Name1;
end;

```

```

function TDataCollection4.Compare(Key1, Key2: Pointer): Integer;
begin
  if PString(Key1)^ = PString(Key2)^ then
    Compare := 0
  else if PString(Key1)^ < PString(Key2)^ then
    Compare := -1
  else
    Compare := 1;
end;

(*****
*)
*)
TPickListKeyBox
*)
( TPickListKeyBox )

procedure TPickListKeyBox.SendFirst;
begin
  Message(Owner, evBroadcast, cmFileDescrip, List^.At(0));
end;

procedure TPickListKeyBox.FocusItem(Item: Integer);
begin
  TSortedListBox.FocusItem(Item);
  if Message(Owner, evBroadcast, cmFileDescrip, List^.At(Item)) = nil
  then begin
    end;
end;

function TPickListKeyBox.DataSize:Word;
begin
  DataSize:= SizeOf(List)+SizeOf(Word)+SizeOf(Integer);
end;

procedure TPickListKeyBox.GetData(var Rec);
type
  ZapRec = record
    listname : PCollection;
    listfocused : integer;
    RecCode : integer;
  end;
var
  R : ZapRec Absolute Rec;
begin
  R.ListName := List;
  R.ListFocused := Focused;
  R.RecCode := PRecD4(List^.At(Focused))^ .PCode^;
end;

function TPickListKeyBox.GetText(Item: Integer;maxlength:integer): String;
var
  S: String;
begin
  Maxlength:=35;
  GetText := PString(PRecD4(List^.At(Item))^ .Name1)^;
end;

(*****
*)
*)
TListADialog
*)
(*****

```

```

( TListADialog )
(*****
*)
*)
Init
*)
(*****
constructor TListADialog.Init(Collection:PDataCollection4);
const
  FormX = 16;      ( form x position )
  FormY = 3;       ( form y position )
  FormWd = 46;     ( form width )
  FormHt = 17;     ( form length )
  ListX = 1;
  ListY = 3;
  ListHt = 6;
  ListWd = 43;

var
  R : TRect;
  SB : PScrollBar;
  Y : Integer;
  D : DirStr;
  N : String;
  E : ExtStr;
  TitleStr : String[50];
  Bruce : PView;
  ButtonX1,ButtonX2,ButtonWd : Integer;
  ButtonX3 : Integer;

begin
  R.Assign(FormX, FormY, FormX + FormWd, FormY + FormHt);
  N:='Expert System Selection';
  TDialog.Init(R, N);

  ( Read data collection into memory )

  DataCollection4 := Collection;
  if DataCollection4 <> nil then
  begin
    ( Loaded successfully: build ListADialog dialog )

    ( Scrollbar )
    R.Assign(ListX + ListWd, ListY, ListX + ListWd + 1, ListY + ListHt);
    SB := New(PScrollBar, Init(R));
    Insert(SB);

    ( List box )
    R.Assign(ListX, ListY, ListX + ListWd, ListY + ListHt);
    List := New(PPickListKeyBox, Init(R, 1, SB));
    List^.NewList(DataCollection4);
    Insert(List);

    ( Label1 )
    TitleStr:='RuleBase Name';
    R.Assign(1,2,20,3);
    Insert(New(PLabel, Init(R, TitleStr, Bruce)));

    ( Label2 )
    R.Assign(1,10,25,11);
    Insert(New(PLabel, Init(R, 'Rulebase Description', Bruce)));

```

```

( RBaseInfoPane )
R.Assign(2,11,44,13);
Insert(New(PRBaseInfoPane, Init(R)));

( Buttons )
y:=14;
ButtonX1 := 2;
ButtonX2 := 16;
ButtonX3 := 31;
ButtonWd := 12;
R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~C-ancel', cmCancel, bfNormal)));

R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~H-elp', cmHelp, bfNormal)));

R.Assign(ButtonX3, Y, ButtonX3 + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~S-elect', cmOK, bfDefault)));

SelectNext(False);      ( Select first field )
IsValid := True;
List^.SendFirst;

end;
end;

(*****
(*)          Handle Event          (*)
*****)
procedure TListADialog.HandleEvent(var Event: TEvent);

var
    DataWanted : Pointer;

begin
    ( Respond to CANCEL button and ESC )
    if (Event.What = evKeyDown) and (Event.KeyCode = kbEsc) then
    begin
        ClearEvent(Event);
        Event.What := evCommand;
        Event.Command := cmCancel;
        PutEvent(Event);
    end;

    TDialog.HandleEvent(Event);

    with Event do
    if What=evCommand then
    begin
        case Command of
            cmSelect :   begin
                            ClearEvent(Event);
                            Event.What := evCommand;
                            Event.Command := cmOK;
                            PutEvent(Event);
                        end;
        end; ( Case )
        ClearEvent(Event);
    end; ( if )

```

```

with Event do
if What=evBroadcast then
begin
    case Command of
        cmListItemSelected: ( Respond to broadcast from TSortedListBox )
        begin
            ClearEvent(Event);
            Event.What := evCommand;
            Event.Command := cmOK;
            PutEvent(Event);
        end;
    end;
end;

(*****
(*)          Done          (*)
*****)
destructor TListADialog.Done;
begin
    if List <> nil then Dispose(List, Done);
    if DataCollection4 <> nil then Dispose(DataCollection4, Done);
    TDialog.Done;
end;

(*****
(*)          Close          (*)
*****)
procedure TListADialog.Close;
begin
    ( TDialog.Close calls Valid and then Free. Before calling
      Free (which calls Done), tell all attached forms to close.
    )
    if Valid(cmClose) then
    begin
        ( Stop desktop video update in case there are scores of attached forms )
        Desktop^.Lock;
        Message(Desktop, evBroadcast, cmCloseForm, @Self);
        Desktop^.Unlock;
        Free;
    end;
end;

(*****
(*)          TRBaseInfoPane          (*)
*****)
constructor TRBaseInfoPane.Init(var Bounds:TRect);
begin
    TView.Init(Bounds);
    EventMask := EventMask or evBroadcast;
end;

procedure TRBaseInfoPane.Draw;
var
    B,C : TDrawBuffer;
    Color : Word;
    i : Integer;
    Found : Boolean;
    Index : Integer;

```

```

procedure SearchStrBack(S:String;var Index:Integer);
begin
  i:= Index;
  Found := False;
  repeat
    i:=i-1;
    if S[i] = ' ' then Found:=true;
  until (Found=True) or (i=1);
  Index :=i+1;
end;

begin
  if length(Info) >= 42 then
  begin
    Index:=42;
    SearchStrBack(Info,Index);
    Info2 := Copy(Info,Index,length(Info));
    Delete(Info,Index,length(Info));
  end
  else Info2:=' ';

  Color := GetColor($01);           { Selecting Color }
  MoveChar(B, ' ', Color, Size.X);   { Fill buffer with blanks }
  MoveStr(B[1], Info, Color);         { Move Info variable into buffer }
  Writeln(0, 0, Size.X, Length(Info), B); { Write out buffer }

  MoveChar(C, ' ', Color, Size.X);   { Fill buffer with blanks }
  MoveStr(C[1], Info2, Color);        { Move Info variable into buffer }
  Writeln(0, 1, Size.X, Length(Info2), C); { Write out buffer }
end;

function TRBaseInfoPane.GetPalette: PPalette;
const
  P: String[Length(CInfoPane)] = CInfoPane;
begin
  GetPalette := @P;
end;

procedure TRBaseInfoPane.HandleEvent(var Event: TEvent);
begin
  TView.HandleEvent(Event);

  if (Event.What = evBroadcast) and (Event.Command = cmFileDescrip) then
  begin
    Info := PRecD4(Event.InfoPtr)^.Descrip1^;
    DrawView;
  end;
end;

Destructor TRBaseInfoPane.Done;
begin
  TView.Done;
end;

end. ( Unit )

```

```

{*****}
{
  Filename : ListDDL.gpas   Author K'G Mercer
  Date    : September 1993
}
{
  Objects : TRecD3
           : TDataCollection3
           : TPickListKeyBox
           : TListDDialog
           : TListD2Dialog
}
{
  Other   : PInteger      - integer ptr
           : TransferWithCode - record
           : TRecD23       - record
}
{*****}

```

```
unit ListDDLg;
```

```
($F+,O+,X+,S-,D+)
```

```
interface
```

```
uses
```

```
Dos, Objects, Memory, Drivers, Views, Dialogs, StdDlg, CmdnList;
```

```
type
```

```
PInteger = ^Integer;
```

```
PRecD3 = ^TRecD3;
```

```
TRecD3 = object(TObject)
```

```
  Name1 : PString;
```

```
  PCode : PInteger;
```

```
  constructor Init(Name : String ; NewCode : Integer);
```

```
  destructor Done; virtual;
```

```
end;
```

```
PDataCollection3 = ^TDataCollection3;
```

```
TDataCollection3 = object(TSortedCollection)
```

```
  function KeyOf(Item: Pointer): Pointer; virtual;
```

```
  function Compare(Key1, Key2: Pointer): Integer; virtual;
```

```
end;
```

```
PPickListKeyBox = ^TPickListKeyBox;
```

```
TPickListKeyBox = object(TSortedListBox)
```

```
  function GetText(Item: Integer; maxlength:integer): String; virtual;
```

```
  procedure GetData(var Rec);virtual;
```

```
  function DataSize:Word;virtual;
```

```
end;
```

```
PListDDialog = ^TListDDialog;
```

```
TListDDialog = object(TDialog)
```

```
  DataCollection3 : PDataCollection3;
```

```
  List            : PPickListKeyBox;
```

```
  IsValid         : Boolean;
```

```
  Modified        : Boolean;
```

```
  constructor Init(Collection:PDataCollection3);
```

```
  destructor Done; virtual;
```

```
  procedure Close; virtual;
```

```
  procedure HandleEvent(var Event: TEvent); virtual;
```

```
end;
```

```
PListD2Dialog = ^TListD2Dialog;
```

```
TListD2Dialog = object(TListDDialog)
```

```
  constructor Init(Collection:PDataCollection3;S:String);
```

```
end;
```

```
TransferWithCode = record
```

```
  Listname      : PDataCollection3;
```

```
  Listfocused   : Integer;
```

```
  RecCode       : Integer;
```

```
end;
```

```
TRecD23 = record
```

```
  Name : String;
```

```
end;
```

```
function FileExists(Name: PathStr): Boolean;
```

```
implementation
```

```
uses App, MsgBox;
```

```
function FileExists(Name: PathStr): Boolean;
```

```
var
```

```
  SR: SearchRec;
```

```
begin
```

```
  FindFirst(Name, 0, SR);
```

```
  FileExists := DosError = 0;
```

```
end;
```

```
{*****}
```

```
(* TListD2Dialog *)
```

```
{*****}
```

```
{ TListD2Dialog }
```

```
{*****}
```

```
(* Init *)
```

```
{*****}
```

```
constructor TListD2Dialog.Init(Collection:PDataCollection3;S:String);
```

```
const
```

```
  FormX = 23;      { form x position }
```

```
  FormY = 5;       { form y position }
```

```
  FormWd = 30;     { form width }
```

```
  FormHt = 14;     { form length }
```

```
  ListX = 1;
```

```
  ListY = 2;
```

```
  ListHt = 6;
```

```
  ListWd = 27;
```

```
var
```

```
  R: TRect;
```

```
  SB: PScrollBar;
```

```
  Y: Integer;
```

```
  D: DirStr;
```

```
  N: String;
```

```
  E: ExtStr;
```

```
  Bruce : PView;
```

```
  ButtonX1,ButtonX2,ButtonWd : Integer;
```

```
begin
```

```

R.Assign(FormX, FormY, FormX + FormWd, FormY + FormHt);
n:='Search List';
TDialog.Init(R, N);

( Read data collection into memory )

DataCollection3 := Collection;
if DataCollection3 <> nil then
begin
    ( Loaded successfully: build ListDDialog dialog )

    ( Scrollbar )
    R.Assign(ListX + ListWd, ListY, ListX + ListWd + 1, ListY + ListHt);
    SB := New(PScrollBar, Init(R));
    Insert(SB);

    ( List box )
    R.Assign(ListX, ListY, ListX + ListWd, ListY + ListHt);
    List := New(PPickListKeyBox, Init(R, 1, SB));
    List^.NewList(DataCollection3);
    Insert(List);

    ( Label - S came from Init var)
    R.Assign(2,9,28,10);
    Insert(New(PLabel, Init(R,S, Bruce)));

    ( Buttons )
    y:=11;
    ButtonX1 := 2;
    ButtonX2 := 15;
    ButtonWd := 12;
    R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
    Insert(New(PButton, Init(R, 'Cancel', cmCancel, bfDefault)));

    R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
    Insert(New(PButton, Init(R, '~S~elect', cmOK, bfNormal)));

    SelectNext(False);      ( Select first field )
    IsValid := True;
end;
end;

{*****}
{*          TRecD3          *}
{*****}
constructor TRecD3.Init(Name : String ; NewCode : Integer);
begin
    Name1 := NewStr(Name);
    New(PCode);
    PCode^ := NewCode;
end;

destructor TRecD3.Done;
begin
    DisposeStr(Name1);
    Dispose(PCode);
end;

```

```

{*****}
{*          TDataCollection3          *}
{*****}

function TDataCollection3.KeyOf(Item: Pointer): Pointer;
begin
    KeyOf := PRecD3(Item)^.Name1;
end;

function TDataCollection3.Compare(Key1, Key2: Pointer): Integer;
begin
    if PString(Key1)^ = PString(Key2)^ then
        Compare := 0
    else if PString(Key1)^ < PString(Key2)^ then
        Compare := -1
    else
        Compare := 1;
end;

{*****}
{*          TPickListKeyBox          *}
{*****}
( TPickListKeyBox )

function TPickListKeyBox.DataSize:Word;
begin
    DataSize:= SizeOf(List)+SizeOf(Word)+SizeOf(Integer);
end;

procedure TPickListKeyBox.GetData(var Rec);
type
    ZapRec = record
        listname : PCollection;
        listfocused : integer;
        RecCode : integer;
    end;
var
    R : ZapRec Absolute Rec; ( Assign same address as Rec )
begin
    R.ListName := List;
    R.ListFocused := Focused;
    R.RecCode := PRecD3(List^.At(Focused))^ .PCode^;
end;

function TPickListKeyBox.GetText(Item: Integer;maxlength:integer): String;
var
    S: String;
begin
    GetText := PString(PRecD3(List^.At(Item))^ .Name1)^;
end;

{*****}
{*          TListDDialog          *}
{*****}
( TListDDialog )
{*****}
{*          Init          *}
{*****}
constructor TListDDialog.Init(Collection:PDataCollection3);

```

```

const
  FormX = 13;      ( form x position )
  FormY = 4;       ( form y position )
  FormWd = 53;     ( form width )
  FormHt = 16;     ( form length )
  ListX = 1;
  ListY = 4;
  ListHt = 6;
  ListWd = 50;

var
  R: TRect;
  SB: PScrollBar;
  Y: Integer;
  D: DirStr;
  N: String;
  E: ExtStr;
  Bruce : PView;
  ButtonX1,ButtonX2,ButtonWd : Integer;
  ButtonX3,ButtonX4,ButtonWd2: Integer;
  Header: string;

begin
  R.Assign(FormX, FormY, FormX + FormWd, FormY + FormHt);
  n:='Orchard Row Information';
  TDialog.Init(R, N);

  ( Read data collection into memory )

  DataCollection3 := Collection;
  if DataCollection3 <> nil then
  begin
    ( Loaded successfully: build ListDDialog dialog )

    ( Scrollbar )
    R.Assign(ListX + ListWd, ListY, ListX + ListWd + 1, ListY + ListHt);
    SB := New(PScrollBar, Init(R));
    Insert(SB);

    ( List box )
    R.Assign(ListX, ListY, ListX + ListWd, ListY + ListHt);
    List := New(PPickListKeyBox, Init(R, 1, SB));
    List^.NewList(DataCollection3);
    Insert(List);

    ( Label )
    R.Assign(1,2,52,3);
    Header:='Row Row      Crop      Plant Row  Plant No';
    Insert(New(PLabel, Init(R,Header, Bruce)));
    R.Assign(1,3,52,4);
    Header:='St. End      Date Width Spac  Plants';
    Insert(New(PLabel, Init(R,Header, Bruce)));

    ( Label )
    R.Assign(2,11,22,12);
    Insert(New(PLabel, Init(R, 'Select Row to Edit', Bruce)));

    ( Buttons )
    y:=13;

```

```

  ButtonX1 := 1;
  ButtonX2 := 13;
  ButtonX3 := 25;
  ButtonX4 := 39;
  ButtonWd := 12;
  ButtonWd2:= 14;
  R.Assign(ButtonX1, Y, ButtonX1 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~C~ancel', cmCancel, bfNormal)));

  R.Assign(ButtonX2, Y, ButtonX2 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~S~elect', cmOK, bfDefault)));

  R.Assign(ButtonX3, Y, ButtonX3 + ButtonWd2, Y + 2);
  Insert(New(PButton, Init(R, '~D~elete Row', cmDeleteRow, bfNormal)));

  R.Assign(ButtonX4, Y, ButtonX4 + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~A~dd Row', cmAddRow, bfNormal)));

  SelectNext(False);      ( Select first field )
  IsValid := True;
end;
end;

(*****
*)          Handle Event          (*)
*****)
procedure TListDDialog.HandleEvent(var Event: TEvent);

var
  DataWanted : Pointer;

begin
  ( Respond to CANCEL button and ESC )
  if (Event.What = evKeyDown) and (Event.KeyCode = kbEsc) then
  begin
    ClearEvent(Event);
    Event.What := evCommand;
    Event.Command := cmCancel;
    PutEvent(Event);
  end;

  TDialog.HandleEvent(Event);

  with Event do
  if What=evCommand then
  begin
    case Command of
      cmSelect :   begin
                      ClearEvent(Event);
                      Event.What := evCommand;
                      Event.Command := cmOK;
                      PutEvent(Event);
                    end;
      cmDeleteRow : begin
                      ClearEvent(Event);
                      ( Send message back to DisplayDialog for type row opera
tion )
                      ( if Message(Application,evBroadcast,cmDeleteRowBrdcst,ni
l)

```

```

        = nil then
        MessageBox(#3'Message = nil', nil,
        mfInformation + mfOkButton);
        Event.What := evCommand;
        Event.Command := cmOK;
        PutEvent(Event);
    } end;
cmAddRow : begin
    ClearEvent(Event);
    { Send message back to DisplayDialog for type row opera
tion }
    { if Message(Application,evBroadcast,cmAddRowBrdCst,nil)
    = nil then
    MessageBox(#3'Message = nil', nil,
    mfInformation + mfOkButton);
    Event.What := evCommand;
    Event.Command := cmOK;
    PutEvent(Event);
    } end;
end; { Case }
ClearEvent(Event);
end; { if }

with Event do
if What=evBroadcast then
begin
case Command of
cmListItemSelected: { Respond to broadcast from TSortedListBox }
begin
    ClearEvent(Event);
    Event.What := evCommand;
    Event.Command := cmOK;
    PutEvent(Event);
end;
end;
end;

(*****
*)           Done           *)
(*****
destructor TListDDialog.Done;
begin
    if List <> nil then Dispose(List, Done);
    if DataCollection3 <> nil then Dispose(DataCollection3, Done);
    TDialog.Done;
end;

(*****
*)           Close          *)
(*****
procedure TListDDialog.Close;
begin
    { TDialog.Close calls Valid and then Free. Before calling
    Free (which calls Done), tell all attached forms to close.
    }
    if Valid(cmClose) then
    begin
        { Stop desktop video update in case there are scores of attached forms }
        Desktop.Lock;

```

```

        Message(Desktop, evBroadcast, cmCloseForm, @Self);
        Desktop.Unlock;
        Free;
    end;
end;

end. { Unit }

```



```

(*****
(
( Turbo Pascal 6.0
( Turbo Vision Forms Demo
( Copyright (c) 1990 by Borland International
(
(*****
)
)
)

unit ListDlg;

{$F+,O+,X+,S-,D+}

interface

uses
  Dos, Objects, Memory, Drivers, Views, Dialogs, StdDlg,
  DataColl, CmdnList;

type
  PListKeyBox = ^TListKeyBox;
  TListKeyBox = object(TSortedListBox)
    function GetText(Item: Integer; MaxLen: Integer): String; virtual;
  end;

  PListDialog = ^TListDialog;
  TListDialog = object(TDialog)
    DataCollection: PDataCollection;
    FileName: PString;
    FormDataFile: PResourceFile;
    IsValid: Boolean;
    List: PListKeyBox;
    Modified: Boolean;
    constructor Init(RezName: PathStr);
    destructor Done; virtual;
    procedure Close; virtual;
    procedure HandleEvent(var Event: TEvent); virtual;
    function OpenDataFile(Name: PathStr;
      var DataFile: PResourceFile; Mode: Word): Boolean;
    function SaveList: Boolean;
    function SaveForm(F: PDialog): Boolean;
    procedure StackOnPrev(F: PDialog);
    function Valid(Command: Word): Boolean; virtual;
  end;

function FileExists(Name: PathStr): Boolean;

implementation

uses App, Forms, MsgBox;

function FileExists(Name: PathStr): Boolean;
var
  SR: SearchRec;
begin
  FindFirst(Name, 0, SR);
  FileExists := DosError = 0;
end;

( TListKeyBox )
function TListKeyBox.GetText(Item: Integer; MaxLen: Integer): String;

```

```

var
  S: String;
begin
  with PDataCollection(List)^ do
  begin
    case KeyType of
      StringKey: GetText := TSortedListBox.GetText(Item, MaxLen);
      LongIntKey:
        begin
          Str(LongInt(KeyOf(At(Item)))^:MaxLen - 3, S);
          GetText := Copy(S, 1, MaxLen);
        end;
    end;
  end;
end;

( TListDialog )
constructor TListDialog.Init(RezName: PathStr);
const
  ButtonCt = 4;
  FormX = 2;
  FormY = 2;
  FormWd = 30;
  FormHt = 13;
  ListX = 2;
  ListY = 3;
  DefaultListWd = 12;
  ListHt = ButtonCt * 2;
  ButtonWd = 12;
  ButtonY = ListY;
var
  R: TRect;
  SB: PScrollBar;
  Y: Integer;
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  F: PForm;
  ListWd: Word;
  ButtonX: Word;
begin
  FSplit(FExpand(RezName), D, N, E);
  R.Assign(FormX, FormY, FormX + FormWd, FormY + FormHt);
  TDialog.Init(R, N);

  FileName := NewStr(D + N + E);
  ( Read data off resource stream )
  if OpenDataFile(FileName^, FormDataFile, stOpen) then
  begin
    ( Get horizontal size of key field )
    F := PForm(FormDataFile^.Get('FormDialog'));
    if F = nil then
    begin
      MsgBox('Error accessing file data.', nil, mfError or mfOkButton);
      Exit;
    end;

    ( Base listbox width on key width. Grow entire dialog if required )
    if F^.KeyWidth > DefaultListWd then

```

```

begin
  ListWd := F^.KeyWidth;
  GrowTo(FormWd + ListWd - DefaultListWd, FormHt);
end
else ListWd := DefaultListWd;

( Move to upper right corner of desktop )
Desktop^.GetExtent(R);           ( Desktop coordinates )
MoveTo(R.B.X - Size.X, 1);

Dispose(F, Done);

( Read data collection into memory )
DataCollection := PDataCollection(FormDataFile^.Get('FormData'));
if DataCollection <> nil then
begin
  ( Loaded successfully: build ListDialog dialog )

  ( Scrollbar )
  R.Assign(ListX + ListWd, ListY, ListX + ListWd + 1, ListY + ListHt);
  SB := New(PScrollBar, Init(R));
  Insert(SB);

  ( List box )
  R.Assign(ListX, ListY, ListX + ListWd, ListY + ListHt);
  List := New(PListKeyBox, Init(R, 1, SB));
  List^.NewList(DataCollection);
  Insert(List);

  ( Label )
  R.Assign(ListX, ListY - 1, ListX + 10, ListY);
  Insert(New(PLabel, Init(R, '~K-ey's', List)));

  ( Buttons )
  ButtonX := ListX + ListWd + 2;
  Y := ButtonY;
  R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~E-dit', cmFormEdit, bfDefault)));

  Inc(Y, 2);
  R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~N-ew', cmFormNew, bfNormal)));

  Inc(Y, 2);
  R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~D-elete', cmFormDel, bfNormal)));

  Inc(Y, 2);
  R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
  Insert(New(PButton, Init(R, '~S-ave', cmListSave, bfNormal)));

  SelectNext(False);           ( Select first field )
  IsValid := True;
end;
end;
end;

destructor TListDialog.Done;
begin
  if List <> nil then Dispose(List, Done);

```

```

  if DataCollection <> nil then Dispose(DataCollection, Done);
  if FormDataFile <> nil then Dispose(FormDataFile, Done);
  if FileName <> nil then DisposeStr(FileName);
  TDialog.Done;
end;

procedure TListDialog.Close;
begin
  ( TDialog.Close calls Valid and then Free. Before calling
    Free (which calls Done), tell all attached forms to close.
  )
  if Valid(cmClose) then
  begin
    ( Stop desktop video update in case there are scores of attached forms )
    Desktop^.Lock;
    Message(Desktop, evBroadcast, cmCloseForm, @Self);
    Desktop^.Unlock;
    Free;
  end;
end;

procedure TListDialog.HandleEvent(var Event: TEvent);

function EditingForm: PForm;
( Return pointer to the form that is editing the current selection )
begin
  EditingForm := Message(Desktop, evBroadcast,
    cmEditingForm, DataCollection^.At(List^.Focused));
end;

procedure FormOpen(NewForm: Boolean);
var
  F: PForm;
begin
  if not NewForm then
  begin
    ( Empty collection? )
    if DataCollection^.Count = 0 then Exit;

    ( If selection is being edited, then bring its form to top )
    F := EditingForm;
    if F <> nil then
    begin
      F^.Select;
      Exit;
    end;
  end;

  ( Selection is not being edited: open new form from the resource file )
  F := PForm(FormDataFile^.Get('FormDialog'));
  if F = nil then
    MessageBox('Error opening form.', nil, mfError or mfOkButton)
  else
  begin
    with F^ do
    begin
      ListDialog := @Self;           ( Form points back to List )
      if NewForm then
        PrevData := nil             ( Adding new form )
      else

```

```

begin
  { Edit data from collection }
  PrevData := DataCollection^.At(List^.Focused);
  SetData(PrevData^);
end;
end;
if Application^.ValidView(F) <> nil then
begin
  StackOnPrev(F);
  if NewForm then Desktop^.Insert(F)    { Insert & select }
  else Desktop^.InsertBefore(F, Next); { Insert but keep focus }
end;
end;
end;
end;

procedure DeleteSelection;
var
  F: TForm;
begin
  { Empty collection? }
  if DataCollection^.Count = 0 then Exit;

  { Don't allow delete of data already being edited }
  F := EditingForm;
  if F <> nil then
  begin
    F^.Select;
    MessageBox('Data is already being edited. Close form before deleting.',
      nil, mfWarning or mfOkButton);
    Exit;
  end;
end;

{ Confirm delete }
if MessageBox('Are you sure you want to delete this item?', nil,
  mfWarning or mfYesNoCancel) = cmYes then
begin
  DataCollection^.AtFree(List^.Focused);
  List^.SetRange(DataCollection^.Count);
  List^.DrawView;
  Modified := True;
end;
end;

begin
  with Event do
  if (What = evKeyDown) and (KeyCode = kbEsc) then
  begin
    What := Command;
    Command := cmClose;
  end;

  TDialog.HandleEvent(Event);

  case Event.What of
    evCommand:
    begin
      case Event.Command of
        cmFormEdit: FormOpen(False);
        cmFormNew : FormOpen(True);
        cmFormDel  : DeleteSelection;

```

```

      cmListSave: if Modified then SaveList;
    else
      Exit;
  end;
  ClearEvent(Event);
end;
evKeyDown:
begin
  case Event.KeyCode of
    kbIns: FormOpen(True);
  else
    Exit;
  end;
  ClearEvent(Event);
end;
evBroadcast:
begin
  case Event.Command of
    { Respond to broadcast from TSortedListBox }
    cmListItemSelected: FormOpen(False);

    { Keep file from being edited simultaneously by 2 lists }
    cmEditingFile: if FileName^ = PString(Event.InfoPtr)^ then
      ClearEvent(Event);

    { Respond to search for topmost list dialog }
    cmTopList: ClearEvent(Event);
  end;
end;
end;
end;

function TListDialog.OpenDataFile(Name: PathStr;
  var DataFile: PResourceFile; Mode: Word): Boolean;
var
  S: PStream;
begin
  S := New(PBufStream, Init(Name, Mode, 1024));
  DataFile := New(PResourceFile, Init(S));
  if S^.Status <> stOk then
  begin
    Dispose(DataFile, Done);
    DataFile := nil;
    OpenDataFile := False;
  end
  else OpenDataFile := True;
end;

function TListDialog.SaveList: Boolean;
var
  S: PStream;
  NewDataFile: PResourceFile;
  Form: TForm;
  D: DirStr;
  N: NameStr;
  E: ExtStr;
  F: File;
begin
  { Empty collection? Unedited? }
  if (DataCollection^.Count = 0) or not Modified then

```

```

begin
  SaveList := True;
  Exit;
end;

SaveList := False;
{ Read form definition out of original form file }
Form := PForm(FormDataFile^.Get('FormDialog'));
if Form = nil then
  MessageBox('Cannot find original file. Data not saved.',
    nil, mfError or mfOkButton)
else
begin
  { Create new data file }
  FSplit(FileName^, D, N, E);
  if not OpenDataFile(D + N + '.$$$', NewDataFile, stCreate) then
    MessageBox('Cannot create file. Data not saved.',
      nil, mfError or mfOkButton)
  else
  begin
    { Create new from form and collection in memory }
    NewDataFile^.Put(Form, 'FormDialog');
    NewDataFile^.Put(DataCollection, 'FormData');
    NewDataFile^.Flush;
    Dispose(NewDataFile, Done);

    { Close original file, rename to .BAK }
    Dispose(FormDataFile, Done);
    FormDataFile := nil;
    ($I-)
    if FileExists(D + N + '.BAK') then
      begin
        Assign(F, D + N + '.BAK');
        Erase(F);
      end;
    Assign(F, FileName^);
    Rename(F, D + N + '.BAK');
    ($I+)

    { Error trying to erase old .BAK or rename original to .BAK? }
    if IOResult <> 0 then
      begin
        MessageBox('Cannot create .BAK file. Data not saved.',
          nil, mfError or mfOkButton);

        { Try to re-open original. New data will still be in memory }
        if not OpenDataFile(FileName^, FormDataFile, stOpen) then
          begin
            MessageBox('Cannot re-open original file.',
              nil, mfError or mfOkButton);
            Free; { Cannot proceed. Free data and close window }
          end;
        end
      else
      begin
        { Rename temp file to original file and re-open }
        Assign(F, D + N + '.$$$');
        Rename(F, FileName^);
        OpenDataFile(FileName^, FormDataFile, stOpen);
      end
    end
  end
end

```

```

  Modified := False;
  SaveList := True;
end;
end;
Dispose(Form, Done);
end;
end;

function TListDialog.SaveForm(F: PDialog): Boolean;
var
  i: Integer;
  P: Pointer;
begin
  SaveForm := False;
  with PForm(F)^, DataCollection^ do
  begin
    { Validate data before updating collection }
    if not F^.Valid(cmFormSave) then Exit;

    { Extract data from form. Don't use safety pool. }
    P := MemAlloc(ItemSize);
    if P = nil then
      begin
        Application^.OutOfMemory;
        Exit;
      end;

    GetData(P^);
    { If no duplicates, make sure not attempting to add duplicate key }
    if not Duplicates and Search(KeyOf(P), i) then
      if (PrevData = nil) or (PrevData <> At(i)) then
        begin
          FreeMem(P, ItemSize);
          MessageBox('Duplicate keys are not allowed in this database.' +
            ' Delete duplicate record before saving this form.', nil,
            mfError or mfOkButton);
          Exit;
        end;

    { Free previous data? }
    if (PrevData <> nil) then Free(PrevData);

    { TDataCollection.Insert may fail because it doesn't use
      the safety pool. Check status field after insert and cleanup
      if necessary.
    }
    Insert(P);
    if Status <> 0 then
      begin
        FreeMem(P, ItemSize);
        Application^.OutOfMemory;
        Exit;
      end;

    { Success: store off original data pointer }
    PrevData := P;

    { Redraw list }
    List^.SetRange(Count);
    List^.DrawView;
  end
end

```

```

    Modified := True;
    SaveForm := True;
end;
end;

procedure TListDialog.StackOnPrev(F: PDialog);
var
    TopForm: PForm;
    R: TRect;
begin
    { Stack on top topmost form or on top list if first form }
    TopForm := Message(Owner, evBroadcast, cmTopForm, @Self);
    if (TopForm <> nil) then
        { Stack on top previous topmost form }
        with TopForm^.Origin do
            F^.MoveTo(X + 1, Y + 1)
        else
            begin
                { Stack right or left of ListDialog }
                if Origin.X > F^.Size.X then F^.MoveTo(0, Origin.Y)
                else F^.MoveTo(Origin.X + Size.X + 1, Origin.Y);
            end;

    { Visible on desktop? Make sure at least half of form is visible }
    Owner^.GetExtent(R);
    with F^, F^.Origin do
        { Desktop coordinates }
        { Keep stack on screen }
        begin
            if (X + Size.X div 2 > R.B.X) then F^.MoveTo(0, 1);
            if (Y + Size.Y div 2 > R.B.Y) then F^.MoveTo(X, 1);
        end;
    end;

function TListDialog.Valid(Command: Word): Boolean;
var
    Ok: Boolean;
    Reply: Word;
begin
    Ok := True;
    case Command of
        cmValid:
            begin
                Ok := IsValid;
                if not Ok then
                    MessageBox('Error opening file (%S).',
                        @FileName, mfError or mfOkButton);
            end;
        cmQuit, cmClose:
            begin
                { Any forms open that cannot close? }
                Ok := Message(Desktop, evBroadcast, cmCanCloseForm, @Self) = nil;

                { Any data modified? }
                if Ok and Modified then
                    begin
                        Select;
                        Reply := MessageBox('Database has been modified. Save? ', nil,
                            mfYesNoCancel);
                        case Reply of
                            cmYes: Ok := SaveList;

```

```

        cmNo: Modified := False;
        { abandon changes }
    else
        Ok := False;
        { cancel close request }
    end;
end;
end;
end;
end;
if Ok then Valid := TDialog.Valid(Command)
else Valid := False;
end;

end.

```

```

(*****)
(
(   Filename : ListDdlg.pas   Author K G Mercer )
(   Date      : September 1993 )
( )
(   Objects   : TRecD1 )
(               : TDataCollection1 )
(               : TListKeyBox )
(               : TListFDlg )
( )
(   Other     : PInteger      - integer ptr )
(               : TransferWithCode - record )
(               : TRecD12      - record )
( )
(*****)

```

```
unit ListFDlg;
```

```
($F+,O+,X+,S-,D+)
```

```
interface
```

```
uses
```

```
  Dos, Objects, Memory, Drivers, Views, Dialogs, StdDlg, CmdnList,
  General2, Help;
```

```
type
```

```
  PInteger = ^Integer;
```

```

PRecD1 = ^TRecD1;
TRecD1 = object(TObject)
  BigName : PString;
  PCode   : PInteger;
  constructor Init(Name : String ; NewCode : Integer);
  destructor Done; virtual;
end;

```

```

PDataCollection1 = ^TDataCollection1;
TDataCollection1 = object(TSortedCollection)
  function KeyOf(Item: Pointer): Pointer; virtual;
  function Compare(Key1, Key2: Pointer): Integer; virtual;
end;

```

```

PListKeyBox = ^TListKeyBox;
TListKeyBox = object(TSortedListBox)
  function GetText(Item: Integer; maxlength: integer): String; virtual;
  procedure GetData(var Rec); virtual;
  function DataSize: Word; virtual;
end;

```

```

PListFDlg = ^TListFDlg;
TListFDlg = object(TDialog)
  DataCollection1 : PDataCollection1;
  List            : PListKeyBox;
  IsValid         : Boolean;
  Modified        : Boolean;
  constructor Init(Collection: PDataCollection1);
  destructor Done; virtual;
  procedure Close; virtual;
  procedure HandleEvent(var Event: TEvent); virtual;
end;

```

```
end;
```

```

TransferWithCode = record
  Listname      : PDataCollection1;
  Listfocused   : Integer;
  RecCode       : Integer;
end;

```

```

TRecD12 = record
  Name : String;
  Code : String;
end;

```

```
function FileExists(Name: PathStr): Boolean;
```

```

var
  TRec : TRecD12;

```

```
implementation
```

```
uses App, MsgBox;
```

```
function FileExists(Name: PathStr): Boolean;
```

```

var
  SR: SearchRec;
begin
  FindFirst(Name, 0, SR);
  FileExists := DosError = 0;
end;

```

```

(*****)
(*          TRecD1          *)
(*****)
constructor TRecD1.Init(Name : String; NewCode : Integer);
begin
  BigName := NewStr(Name);
  New(PCode);
  PCode^ := NewCode;
end;

```

```

destructor TRecD1.Done;
begin
  DisposeStr(BigName);
  Dispose(PCode);
end;

```

```

(*****)
(*          TDataCollection1          *)
(*****)

```

```

function TDataCollection1.KeyOf(Item: Pointer): Pointer;
begin
  KeyOf := PRecD1(Item)^.BigName;
end;

```

```

function TDataCollection1.Compare(Key1, Key2: Pointer): Integer;
begin
  if PString(Key1)^ = PString(Key2)^ then
    Compare := 0
  else if PString(Key1)^ < PString(Key2)^ then

```

```

    Compare := -1
  else
    Compare := 1;
end;

(*****
*)      TListKeyBox      *
(*****
{ TListKeyBox }

function TListKeyBox.DataSize:Word;
begin
  DataSize:= SizeOf(List)+SizeOf(Word)+SizeOf(Integer);
end;

procedure TListKeyBox.GetData(var Rec);
type
  ZapRec = record
    listname      : PCollection;
    listfocused   : integer;
    RecCode       : integer;
  end;
var
  R : ZapRec Absolute Rec; { Assign same address as Rec }
begin
  R.ListName := List;
  R.ListFocused := Focused;
  R.RecCode := PRecD1(List^.At(Focused))^PCode^;
end;

function TListKeyBox.GetText(Item: Integer;maxlength:integer): String;
var
  S: String;
begin
  GetText := PString(PRecD1(List^.At(Item))^BigName^);
end;

(*****
*)      TListFDialog      *
(*****
{ TListFDialog }
(*****
*)      Init      *
(*****
constructor TListFDialog.Init(Collection:PDataCollection1);
const
  ButtonCt = 4;
  FormX = 18;      { form x position }
  FormY = 2;        { form y position }
  FormWd = 40;      { form width }
  FormHt = 17;      { form length }
  ListX = 2;
  ListY = 3;
  DefaultListWd = 35;
  ListHt = 8;
  ButtonWd = 11;
  ButtonY = ListY+ListHt+1;

var

```

```

  R: TRect;
  SB: PScrollBar;
  Y: Integer;
  D: DirStr;
  N: String;
  E: ExtStr;
  ListWd: Word;
  ButtonX: Word;
begin
  R.Assign(FormX, FormY, FormX + FormWd, FormY + FormHt);
  n:='Batchrun List';
  TDialog.Init(R, N);
  ListWd := DefaultListWd;

  { Read data collection into memory }

  DataCollection1 := Collection;
  if DataCollection1 <> nil then
  begin
    { Loaded successfully: build ListFDialog dialog }

    { Scrollbar }
    R.Assign(ListX + ListWd, ListY, ListX + ListWd + 1, ListY + ListHt);
    SB := New(PScrollBar, Init(R));
    Insert(SB);

    { List box }
    R.Assign(ListX, ListY, ListX + ListWd, ListY + ListHt);
    List := New(PListKeyBox, Init(R, 1, SB));
    List^.NewList(DataCollection1);
    Insert(List);

    { Label }
    R.Assign(ListX, ListY - 1, ListX + 15, ListY);
    Insert(New(PLabel, Init(R, '~B-atchrun Names', List)));

    { Buttons }
    ButtonX := ListX;
    Y := ButtonY;
    R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
    Insert(New(PButton, Init(R, '~H-elp', cmHelp, bfNormal)));

    ButtonX := ListX + 12;
    R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
    Insert(New(PButton, Init(R, '~D-elete', cmDelBatchObject, bfNormal)));

    ButtonX := ListX + 25;
    R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
    Insert(New(PButton, Init(R, '~N-ew', cmAddBatchObject, bfNormal)));

    ButtonX := ListX;
    Y := ButtonY+2;
    R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
    Insert(New(PButton, Init(R, '~C-ancel', cmCancel, bfNormal)));

    ButtonX := ListX + 25;
    R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
    Insert(New(PButton, Init(R, '~S-elect', cmSelect, bfDefault)));
    SelectNext(False);      { Select first field }
  end;
end;

```

```

    IsValid := True;
end;
end;

(*****
(*          Handle Event          *)
*****)
procedure TListFDialog.HandleEvent(var Event: TEvent);
var
    DataWanted : Pointer;

procedure AddBatchObject;
label
    Start, Finished;
var
    TD1          : PEditDialog3;
    PNewRecD1    : PRecD1;
    Control, Option : Word;
    NewName, CheckName : String;
    NewCode, RecCode, i : Integer;

begin
    Start:
    TRec.Name := ' ';
    TD1 := New(PEditDialog3, Init);
    TD1^.HelpCtx := hcNewBatchName;
    TD1^.SetData(TRec);
    Control := Desktop^.ExecView(TD1);
    if Control <> cmCancel then TD1^.GetData(TRec);
    Dispose(TD1, Done);

    if control <> cmCancel then
    begin
        NewName := TRec.Name;
        { Check to see whether duplicate name exists }

        for i:= 0 to DataCollection1^.Count-1 do
        begin
            CheckName := PRecD1(DataCollection1^.At(i)).BigName^;
            if CheckName = NewName then
            begin
                Option := MessageBox(#3+'Duplicate name exists'+#13#13#3+
                    'Do you wish to try a again?', nil, mfYesNoCancel);

                if Option = cmYes then goto start;
                if (Option = cmNo) or (Option = cmCancel) then goto finished;
            end;
        end;

        { Look for maximum code in list then increment by one }
        NewCode := 0;
        for i:= 0 to DataCollection1^.Count-1 do
        begin
            RecCode := PRecD1(DataCollection1^.At(i)).PCode^;
            if RecCode > NewCode then NewCode := RecCode;
        end;
        NewCode:=NewCode+1;

        { Save application data }
        PNewRecD1 := New(PRecD1, Init(NewName, NewCode));
    end;
end;

```

```

    DataCollection1^.Insert(PNewRecD1);
    List^.SetRange(DataCollection1^.Count);
    List^.DrawView;

    { Send information through to TFormApp.Batchfiles (thesis2.pas)
      for insertion }
    if Message(Application, evBroadcast, cmNewBatchObjBrdcst, PNewRecD1) = nil then
    en
        MessageBox(#3'Message = nil', nil, mfInformation + mfOkButton);

    end; { cmCancel }
    Finished:
    end;

procedure DelBatchObject;
var
    Option : Word;
    ShowName : String;
    PDelCode : PInteger;
    DelCode : Integer;

begin
    ShowName := PRecD1(DataCollection1^.At(List^.Focused)).BigName^;
    Option := MessageBox(#3+'Are you sure you wish to delete :'+#13#13#3+
        'project batchrun : '+ShowName, nil, mfYesNoCancel);

    if Option = cmYes then
    begin
        DelCode := PRecD1(DataCollection1^.At(List^.Focused)).PCode^;
        New(PDelCode);
        PDelCode^:= DelCode;

        DataCollection1^.AtFree(List^.Focused);
        List^.SetRange(DataCollection1^.Count);
        List^.DrawView;

        { Send information through to TFormApp.Batchfiles (thesis2.pas)
          for deletion }
        if Message(Application, evBroadcast, cmDelBatchObjBrdcst, PDelCode) = nil then
        n
            MessageBox(#3'Message = nil', nil, mfInformation + mfOkButton);
        end;
    end;

begin
    { Respond to CANCEL button and ESC }
    if (Event.What = evKeyDown) and (Event.KeyCode = kbEsc) then
    begin
        ClearEvent(Event);
        Event.What := evCommand;
        Event.Command := cmCancel;
        PutEvent(Event);
    end;

    TDialog.HandleEvent(Event);

    with Event do
        if What=evCommand then
        begin

```



```
case Command of
  cmAddBatchObject : AddBatchObject;
  cmDelBatchObject : DelBatchObject;
  cmSelect         : begin
                        ClearEvent(Event);
                        Event.What := evCommand;
                        Event.Command := cmOK;
                        PutEvent(Event);
                      end;
end; { Case }
ClearEvent(Event);
end; { if }

with Event do
  if What=evBroadcast then
  begin
    case Command of
      cmListItemSelected: { Respond to broadcast from TSortedListBox }
      begin
        ClearEvent(Event);
        Event.What := evCommand;
        Event.Command := cmOK;
        PutEvent(Event);
      end;
    end; { Case }
  end;
end;

(*****
*)           Done           *)
(*****)
destructor TListFDIALOG.Done;
begin
  if List <> nil then Dispose(List, Done);
  if DataCollection1 <> nil then Dispose(DataCollection1, Done);
  TDialog.Done;
end;

(*****
*)           Close          *)
(*****)
procedure TListFDIALOG.Close;
begin
  { TDialog.Close calls Valid and then Free. Before calling
  Free (which calls Done), tell all attached forms to close.
  }
  if Valid(cmClose) then
  begin
    { Stop desktop video update in case there are scores of attached forms }
    Desktop^.Lock;
    Message(Desktop, evBroadcast, cmCloseForm, @Self);
    Desktop^.Unlock;
    Free;
  end;
end;

end. { Unit }
```

```

{*****}
{
  Filename : ListHdlg.pas   Author K G Mercer }
{   Date   : September 1993 }
{
  Objects : TRecD2         }
{           : TDataCollection2 }
{           : TListKeyBox  }
{           : TListHDialog }
{
  Other   : PInteger       - integer ptr. }
{           : TransferWithCode - record   }
{           : TRecD22       - record     }
{           : TRecD23       - record     }
{*****}

```

```
unit ListHdlg;
```

```
($F+,O+,X+,S-,D+)
```

```
interface
```

```
uses
```

```
Dos, Objects, Memory, Drivers, Views, Dialogs, StdDlg, CmndList, General1,
ListADlg, PKlist1, StrmUnit, Help, General2;
```

```
type
```

```
PInteger = ^Integer;
```

```
PRecD2 = ^TRecD2;
```

```
TRecD2 = object(TObject)
```

```
  BigName : PString;
```

```
  PCode : PInteger;
```

```
  constructor Init(Name : String ; NewCode : Integer);
```

```
  destructor Done; virtual;
```

```
end;
```

```
PDataCollection2 = ^TDataCollection2;
```

```
TDataCollection2 = object(TSortedCollection)
```

```
  function KeyOf(Item: Pointer): Pointer; virtual;
```

```
  function Compare(Key1, Key2: Pointer): Integer; virtual;
```

```
end;
```

```
PListKeyBox = ^TListKeyBox;
```

```
TListKeyBox = object(TSortedListBox)
```

```
  function GetText(Item: Integer; maxlength:integer): String; virtual;
```

```
  procedure GetData(var Rec);virtual;
```

```
  function DataSize:Word;virtual;
```

```
end;
```

```
PListHDialog = ^TListHDialog;
```

```
TListHDialog = object(TDialog)
```

```
  DataCollection2 : PDataCollection2;
```

```
  List : PListKeyBox;
```

```
  IsValid : Boolean;
```

```
  Modified : Boolean;
```

```
  BOKKeyName : PString;
```

```
  constructor Init(Collection:PDataCollection2;BOKName:String;Total:Integer;
```

```
    ListEmpty : Boolean);
```

```

  destructor Done; virtual;
  procedure Close; virtual;
  procedure HandleEvent(var Event: TEvent); virtual;
  procedure FillBlanks(var S:String; StrLength:Integer);
end;
```

```

TransferWithCode = record
  Listname : PDataCollection2;
  Listfocused : Integer;
  RecCode : Integer;
end;
```

```

TRecD22 = record
  Name : String;
  Code : String;
end;
```

```

TRecD23 = record
  Name : String;
end;
```

```
function FileExists(Name: PathStr): Boolean;
```

```
var
```

```
  BatchList2 : PCollection;
```

```
  BatchStream : TBufStream;
```

```
  TransWCode : TransferWithCode;
```

```
  RezName : String[12];
```

```
  BatchRez : TResourceFile;
```

```
  ExpertList : PCollection;
```

```
  ExpertStream : TBufStream;
```

```
  TRec23 : TRecD23;
```

```
implementation
```

```
uses App, MsgBox;
```

```
function FileExists(Name: PathStr): Boolean;
```

```
var
```

```
  SR: SearchRec;
```

```
begin
```

```
  FindFirst(Name, 0, SR);
```

```
  FileExists := DosError = 0;
```

```
end;
```

```

{*****}
{*               TRecD2               *}
{*****}

```

```
constructor TRecD2.Init(Name : String; NewCode : Integer);
```

```
begin
```

```
  BigName := NewStr(Name);
```

```
  New(PCode);
```

```
  PCode^ := NewCode;
```

```
end;
```

```
destructor TRecD2.Done;
```

```
begin
```

```
  DisposeStr(BigName);
```

```
  Dispose(PCode);
```

```
end;
```

```

{*****
(*                               TDataCollection2                               *)
{*****}

function TDataCollection2.KeyOf(Item: Pointer): Pointer;
begin
  KeyOf := PRecD2(Item)^.BigName;
end;

function TDataCollection2.Compare(Key1, Key2: Pointer): Integer;
begin
  if PString(Key1)^ = PString(Key2)^ then
    Compare := 0
  else if PString(Key1)^ < PString(Key2)^ then
    Compare := -1
  else
    Compare := 1;
end;

{*****
(*                               TListKeyBox                               *)
{*****}
( TListKeyBox )

function TListKeyBox.DataSize:Word;
begin
  DataSize:= SizeOf(List)+SizeOf(Word)+SizeOf(Integer);
end;

procedure TListKeyBox.GetData(var Rec);
type
  ZapRec = record
    listname      : PCollection;
    listfocused   : integer;
    RecCode       : integer;
  end;
var
  R : ZapRec Absolute Rec; { Assign same address as Rec }
begin
  R.ListName := List;
  R.ListFocused := Focused;
  R.RecCode := PRecD2(List^.At(Focused))^ .PCode^;
end;

function TListKeyBox.GetText(Item: Integer;maxlength:integer): String;
var
  S: String;

begin
  GetText := PString(PRecD2(List^.At(Item))^ .BigName)^;
end;

{*****
(*                               TListHDialog                               *)
{*****}
( TListHDialog )
{*****}
(*                               Init                               *)
{*****}

```

```

constructor TListHDialog.Init(Collection:PDataCollection;BOKName : String;
                                Total : Integer; ListEmpty : Boolean);
const
  ButtonCt = 4;
  FormX = 20;      { form x position }
  FormY = 1;       { form y position }
  FormWd = 40;     { form width }
  FormHt = 20;     { form length }
  ListX = 2;
  ListY = 5;
  DefaultListWd = 35;
  ListHt = 8;
  ButtonWd = 11;
  ButtonY = ListY+ListHt+2;

var
  R      : TRect;
  SB     : PScrollBar;
  Y      : Integer;
  D      : DirStr;
  N      : String;
  E      : ExtStr;
  ListWd : Word;
  ButtonX : Word;
  IP     : PView;
  StrTotal : String;
  Event  : TEvent;

begin
  R.Assign(FormX, FormY, FormX + FormWd, FormY + FormHt);
  n:='Expert System Include List';
  TDialog.Init(R, N);
  BOKeyName := NewStr(BOKName);
  ListWd := DefaultListWd;

  { Read data collection into memory }

  DataCollection2 := Collection;
  if DataCollection2 <> nil then
    begin
      { Loaded successfully: build ListHDialog dialog }

      { Label }
      R.Assign(2,2,15,3);
      Insert(New(PLabel, Init(R, 'Batch Name', List)));

      {InfoPane}
      R.Assign(16,2,35,3);
      IP := New(PInfoPane, Init(R, BOKeyName^));
      Insert(IP);

      { Label }
      R.Assign(ListX, ListY - 1, ListX + 35, ListY);
      Insert(New(PLabel, Init(R, 'Expert System Rulebases   Weight %', List)));

      { Scrollbar }
      R.Assign(ListX + ListWd, ListY, ListX + ListWd + 1, ListY + ListHt);
      SB := New(PScrollBar, Init(R));
      Insert(SB);
    end
  end;
end;

```

```

{ List box }
R.Assign(ListX, ListY, ListX + ListWd, ListY + ListHt);
List := New(PListKeyBox, Init(R, 1, SB));
List^.NewList(DataCollection2);
Insert(List);

{ Label }
Y := ListY + ListHt;
R.Assign(24, Y, 30, Y+1);
Insert(New(PLabel, Init(R, 'Total', List)));
R.Assign(36, Y, 38, Y+1);
Insert(New(PLabel, Init(R, '%', List)));

{InfoPane}
R.Assign(31, Y, 36, Y+1);
Str(Total, StrTotal);
IP := New(PInfoPane2, Init(R, StrTotal));
Insert(IP);

{ Buttons }
ButtonX := ListX - 1;
Y := ButtonY;
R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~I~nfo', cmInfo, bfNormal)));

ButtonX := ListX + 12;
R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~D~elete', cmDelDataBatchObject, bfNormal)));

ButtonX := ListX + 25;
R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~A~dd', cmAddDataBatchObject, bfNormal)));

ButtonX := ListX - 1;
Y := ButtonY + 2;
R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~H~elp', cmHelp, bfNormal)));

ButtonX := ListX + 11;
R.Assign(ButtonX - 1, Y, ButtonX + ButtonWd + 3, Y + 2);
Insert(New(PButton, Init(R, '~E~dit Weight', cmEditWeight, bfNormal)));

ButtonX := ListX + 25;
R.Assign(ButtonX, Y, ButtonX + ButtonWd, Y + 2);
Insert(New(PButton, Init(R, '~O~k', cmCheckAnExit, bfDefault)));

SelectNext(False);      { Select first field }
IsValid := True;

{ Force input of data first if list is empty }
if ListEmpty = true then
begin
  MessageBox(#3'List is Empty in box', nil, mfInformation + mfOkButton);
  Event.What := evCommand;
  Event.Command := cmAddDataBatchObject;
  PutEvent(Event);
  if Message(owner, evBroadcast, cmAddDataBatchObject, nil) = nil then
    MessageBox(#3'nobody sorted out', nil, mfInformation + mfOkButton);
end;
end;

```

```

end;
end;

(*****
(*                               Handle Event                               *)
(*****
procedure TListHDialog.HandleEvent(var Event: TEvent);

var
  DataWanted : Pointer;

(*****
Procedure AddBatchDataObject;
(*****
Label
  Start, Finished;
var
  Collection1          : PDataCollection4;
  Pld                  : PListADialog;
  Control1, Control2, Option : Word;
  StreamName           : String[12];
  FoundEO              : PExpertObject;
  CodeReturned, NewRBCode : Integer;
  Name2, Descrip1_2, Descrip2_2 : String;
  Descrip3_2           : String;
  1, EditEONo, ErrorNo, Code2, Error : Integer;
  BatchDataList        : PCollection;
  NewCode, RecCode, Weight, Total : Integer;
  NewESName, Description, WeightStr : String;
  PED4                 : PEditDialog4;
  PNewBatchDataObject  : PBatchDataObject;
  PNewRecD2            : PRecD2;
  CheckName, ArrayStr, TotalStr : String;
  PTotal               : PString;
  NewProgName, NewRBName : String;

  function CodeMatch(E0:PExpertObject):Boolean;far;
  begin
    CodeMatch := CodeReturned = EO^.PCode^;
  end;

begin
  Start:
  with ExpertStream do
  begin
    StreamName := 'ExptList.Stm';
    Init(StreamName, stOpenRead, 1024);
    ExpertList := PCollection(Get);
    Done;
    if Status < 0 then
    begin
      MessageBox(#3'Error loading : '+StreamName, nil, mfInformation + mfOkButt
on);
    end;
  end;

  end;

  Collection1 := New(PDataCollection4, Init(ExpertList^.Count-1, 5));

  for i:= 0 to ExpertList^.Count-1 do
  begin

```

```

Name2      := PExpertObject(ExpertList^.At(i))^Name^;
Descrip1_2 := PExpertObject(ExpertList^.At(i))^Descrip1^;
Descrip2_2 := PExpertObject(ExpertList^.At(i))^Descrip2^;
Descrip3_2 := PExpertObject(ExpertList^.At(i))^Descrip3^;
Code2      := PExpertObject(ExpertList^.At(i))^PCode^;

( FillBlanks(Name2,20);)
Collection1^.Insert(New(PRecD4,Init(Name2,Descrip1_2,Descrip2_2,
                               Descrip3_2,Code2)));

end;

( Display ListADialog for rulebase selection )
Pld := New(PListADialog,Init(Collection1));
Pld^.HelpCtx := hcSelectRulebase;
Control1 := Desktop^.ExecView(Pld);
if Control1 <> cmCancel then
begin
    Pld^.GetData(TransWCode);
    CodeReturned := TransWCode.RecCode;
end;
Dispose(Pld, Done);

if Control1 <> cmCancel then
begin
    ( Search for Application with the code returned )
    FoundEO := ExpertList^.FirstThat(@CodeMatch);
    EditEONo := ExpertList^.IndexOf(FoundEO);

    NewESName := PExpertObject(ExpertList^.At(EditEONo))^Name^;
    NewProgName := PExpertObject(ExpertList^.At(EditEONo))^ProgName^;
    NewRBName := PExpertObject(ExpertList^.At(EditEONo))^RuleBaseName^;
    Description := PExpertObject(ExpertList^.At(EditEONo))^Descrip1^;
    NewRBCode := PExpertObject(ExpertList^.At(EditEONo))^PCode^;

    ( Prompt for weighting )

    TRec23.Name := ' ';
    PED4 := New(PEditDialog4,Init);
    PED4^.HelpCtx := hcNewWeight;
    PED4^.SetData(TRec23);
    Control2 := Desktop^.ExecView(PED4);
    if Control2 <> cmCancel then PED4^.GetData(TRec23);
    Dispose(PED4,Done);

    if Control2 <> cmCancel then
    begin
        WeightStr := TRec23.Name;
        Val(WeightStr,Weight,Error);
        if Error <> 0 then
        begin
            MessageBox(#3'Error with weight', nil, mfInformation + mfOkButton);
            Exit;
        end;

        ( Read in batch data from ResourceFile )
        RezName := 'Batch.Rez';
        BatchRez.Init(New(PBufStream,Init(RezName,stOpen,1024)));
        if BatchRez.Stream^.Status <> 0 then
            MessageBox(#3'Error loading Resourcefile', nil, mfInformation + mfOk
kButton);

```

```

BatchDataList := PCollection(BatchRez.Get(BOKeyName^));

( Check to see whether rulebase already used )
for i:= 0 to BatchDataList^.Count-1 do
begin
    CheckName := PBatchDataObject(BatchDataList^.At(i))^ESysName^;
    if CheckName = NewESName then
    begin
        Option := MessageBox(#3+'Duplicate name exists'+#13#13#3+
                              'Do you wish to try a again?', nil, mfYesNoCancel);

        if Option = cmYes then goto start;
        if (Option = cmNo) or (Option = cmCancel) then goto finished;
    end;
end;

( Check for new code )
NewCode :=0;
for i:= 0 to BatchDataList^.Count-1 do
begin
    RecCode := PBatchDataObject(BatchDataList^.At(i))^PCode^;
    if RecCode > NewCode then NewCode := RecCode;
end;
NewCode:=NewCode+1;

( Insert into resourcefile )
PNewBatchDataObject := New(PBatchDataObject,Init(NewESName,Description,
                                                    NewProgName,NewRBName,NewRBCode,Weight,NewCo
de));

BatchDataList^.Insert(PNewBatchDataObject);
BatchRez.Put(BatchDataList,BOKeyName^);
BatchRez.Done;

( Sum up values in BatchDataList )
Total :=0;
for i:= 0 to BatchDataList^.Count-1 do
begin
    Total:= Total + PBatchDataObject(BatchDataList^.At(i))^PWeight^;
end;

Str(Total:3,TotalStr);
PTotal:= NewStr(TotalStr);
if Message(Application,evBroadcast,cmUpdatePane2,PTotal) = nil then
    MessageBox(#3'Message = nil', nil, mfInformation + mfOkButton);

( Update List )
FillBlanks(NewESName,29);
FillBlanks(WeightStr,3);
ArrayStr := NewESName+' '+WeightStr;

PNewRecD2 := New(PRecD2,Init(ArrayStr,NewCode));
DataCollection2^.Insert(PNewRecD2);
List^.SetRange(DataCollection2^.Count);
List^.DrawView;

end; ( Control2 )
end; ( Control1 )
Finished:

```

```

end;

(*****
Procedure DelBatchDataObject;
(*****
var
  Option           : Word;
  ShowName, TotalStr : String;
  PDelCode         : PInteger;
  DelCode, CodeReturned : Integer;
  BatchDataList    : PCollection;
  PDelBatchDataObject : PBatchDataObject;
  PTotal           : PString;
  Total, i         : Integer;

  function CodeMatch(App:PBatchDataObject):Boolean;far;
  begin
    CodeMatch := CodeReturned = App^.PCode^;
  end;

begin
  ShowName := PRecD2(DataCollection2^.At(List^.Focused)).BigName^;
  System.Delete(ShowName,29,5);
  Option := MessageBox(#3+'Are you sure you wish to remove :'+#13#13#3+
    'this rulebase : '+ShowName, nil, mfYesNoCancel);

  if Option = cmYes then
  begin
    CodeReturned := PRecD2(DataCollection2^.At(List^.Focused)).PCode^;

    DataCollection2^.AtFree(List^.Focused);
    List^.SetRange(DataCollection2^.Count);
    List^.DrawView;

    { Update Resource file }

    { Read in batch info from ResourceFile }
    RezName := 'Batch.Rez';
    BatchRez.Init(New(PBufStream, Init(RezName, stOpen, 1024)));
    if BatchRez.Stream^.Status <> 0 then
      MessageBox(#3+'Error loading Resourcefile', nil, mfInformation + mfOkB
utton);

    BatchDataList := PCollection(BatchRez.Get(BOKeyName^));

    { Locate BatchDataObject with that Code }
    PDelBatchDataObject := BatchDataList^.FirstThat(@CodeMatch);

    { Modify BatchObject list then re-insert into resource }
    BatchDataList^.Free(PDelBatchDataObject);
    BatchDataList^.Pack;
    BatchRez.Put(BatchDataList, BOKeyName^);

    { Sum up values in BatchDataList }
    Total := 0;
    for i:= 0 to BatchDataList^.Count-1 do
    begin
      Total:= Total + PBatchDataObject(BatchDataList^.At(i)).PWeight^;
    end;

```

```

    Str(Total:3, TotalStr);
    PTotal:= NewStr(TotalStr);
    if Message(Application, evBroadcast, cmUpdatePane2, PTotal) = nil then
      MessageBox(#3+'Message = nil', nil, mfInformation + mfOkButton);

    BatchRez.Done;
  end;
end; { DelBatchDataObject }

(*****
Procedure EditWeight;
(*****
var
  Option, Control : Word;
  DescripStr      : String;
  DelCode, CodeReturned, i : Integer;
  BatchDataList   : PCollection;
  PEditBatchDataObject : PBatchDataObject;
  PNewBatchDataObject : PBatchDataObject;
  Weight, Error, Total : Integer;
  PED5            : PEditDialog5;
  WeightStr, TotalStr : String;
  PTotal          : PString;
  OldESName, OldDescrip, ArrayStr : String;
  OldCode, OldRBCode : Integer;
  PNewRecD2       : PRecD2;
  OldProgName, OldRBName : String;

  function CodeMatch(App:PBatchDataObject):Boolean;far;
  begin
    CodeMatch := CodeReturned = App^.PCode^;
  end;

begin
  CodeReturned := PRecD2(DataCollection2^.At(List^.Focused)).PCode^;

  { Read in batch info from ResourceFile }
  RezName := 'Batch.Rez';
  BatchRez.Init(New(PBufStream, Init(RezName, stOpen, 1024)));
  if BatchRez.Stream^.Status <> 0 then
    MessageBox(#3+'Error loading Resourcefile', nil, mfInformation + mfOkButto
n);

  BatchDataList := PCollection(BatchRez.Get(BOKeyName^));

  { Locate BatchDataObject with that Code }
  PEditBatchDataObject := BatchDataList^.FirstThat(@CodeMatch);
  Weight := PEditBatchDataObject^.PWeight^;
  Str(Weight, WeightStr);

  { Get new weight }
  TRec23.Name := WeightStr;
  PED5:= New(PEditDialog5, Init);
  PED5^.HelpCtx := hcEditWeight;
  PED5^.SetData(TRec23);
  Control :=Desktop^.ExecView(PED5);
  if Control <> cmCancel then PED5^.GetData(TRec23);
  Dispose(PED5, Done);

  if Control <> cmCancel then

```

```

begin
  WeightStr := TRec23.Name;
  Val(WeightStr,Weight,Error);
  if Error<> 0 then
    begin
      MessageBox(#3'Error with weight', nil, mfInformation + mfOkButton);
      Exit;
    end;

  OldESName := PEditBatchDataObject^.ESysName^;
  OldDescrip := PEditBatchDataObject^.ESysDescrip^;
  OldProgName:= PEditBatchDataObject^.ProgName^;
  OldRBName := PEditBatchDataObject^.RBName^;
  OldRBCode := PEditBatchDataObject^.RBCode^;
  OldCode := PEditBatchDataObject^.PCode^;

  BatchDataList^.Free(PEditBatchDataObject);
  BatchDataList^.Pack;

  PNewBatchDataObject := New(PBatchDataObject,Init(OldESName,OldDescrip,
    OldProgName,OldRBName,OldRBCode,Weight,OldCode)
);

  BatchDataList^.Insert(PNewBatchDataObject);
  BatchRez.Put(BatchDataList,BOKeyName^);

  { Sum up values in BatchDataList }
  Total :=0;
  for i:= 0 to BatchDataList^.Count-1 do
    begin
      Total:= Total + PBatchDataObject(BatchDataList^.At(i))^PWeight^;
    end;

  Str(Total:3,TotalStr);
  PTotal:= NewStr(TotalStr);
  if Message(Application,evBroadcast,cmUpdatePane2,PTotal) = nil then
    MessageBox(#3'Message = nil', nil, mfInformation + mfOkButton);

  { Update List }
  FillBlanks(OldESName,29);
  FillBlanks(WeightStr,3);
  ArrayStr := OldESName+' '+WeightStr;

  DataCollection2^.AtFree(List^.Focused);
  PNewRecD2 := New(PRecD2,Init(ArrayStr,OldCode));
  DataCollection2^.Insert(PNewRecD2);
  List^.SetRange(DataCollection2^.Count);
  List^.DrawView;

end; { Control }
BatchRez.Done;
end; { EditWeight }

(*****
Procedure Info;
(*****
var
  Option,Control      : Word;
  DescripStr          : String;
  DelCode,CodeReturned,i: Integer;
  BatchDataList       : PCollection;

```

```

PViewBatchDataObject : PBatchDataObject;
PTotal                : PString;
PVP1                  : PViewPane1;

  function CodeMatch(App:PBatchDataObject):Boolean;far;
  begin
    CodeMatch := CodeReturned = App^.PCode^;
  end;

begin
  CodeReturned := PRecD2(DataCollection2^.At(List^.Focused))^PCode^;

  { Read in batch info from ResourceFile }
  RezName := 'Batch.Rez';
  BatchRez.Init(New(PBufStream,Init(RezName,stOpen,1024)));
  if BatchRez.Stream^.Status <> 0 then
    MessageBox(#3'Error loading Resourcefile', nil, mfInformation + mfOkButto
n);

  BatchDataList := PCollection(BatchRez.Get(BOKeyName^));

  { Locate BatchDataObject with that Code }
  PViewBatchDataObject := BatchDataList^.FirstThat(@CodeMatch);
  DescripStr := PViewBatchDataObject^.ESysDescrip^;

  PVP1:= New(PViewPane1,Init(DescripStr));
  PVP1^.HelpCtx := hcInfoPane;
  Control :=Desktop^.ExecView(PVP1);
  Dispose(PVP1,Done);

  BatchRez.Done;
end; { Info }

(*****
Function CheckWeight:Boolean;
(*****
var
  BatchDataList      : PCollection;
  Total,i             : Integer;

begin
  { Read in batch info from ResourceFile }
  RezName := 'Batch.Rez';
  BatchRez.Init(New(PBufStream,Init(RezName,stOpen,1024)));
  if BatchRez.Stream^.Status <> 0 then
    MessageBox(#3'Error loading Resourcefile', nil, mfInformation + mfOkButto
n);

  BatchDataList := PCollection(BatchRez.Get(BOKeyName^));
  BatchRez.Done;

  { Sum up values in BatchDataList }
  Total :=0;
  for i:= 0 to BatchDataList^.Count-1 do
    begin
      Total:= Total + PBatchDataObject(BatchDataList^.At(i))^PWeight^;
    end;
    if Total=100 then CheckWeight:=True else CheckWeight:=False;
end; {CheckWeight}

```

```

begin
  ( Respond to CANCEL button and ESC )
  if (Event.What = evKeyDown) and (Event.KeyCode = kbEsc) then
    begin
      ClearEvent(Event);
      Event.What := evCommand;
      Event.Command := cmCancel;
      PutEvent(Event);
    end;

    TDialog.HandleEvent(Event);

  with Event do
    if What=evCommand then
      begin
        case Command of
          cmAddDataBatchObject : AddBatchDataObject;
          cmDelDataBatchObject : DelBatchDataObject;
          cmEditWeight          : EditWeight;
          cmCheckWeight         : CheckWeight;
          cmInfo                 : Info;
          cmSelect               : begin
                                  ClearEvent(Event);
                                  Event.What := evCommand;
                                  Event.Command := cmOK;
                                  PutEvent(Event);
                                end;
          cmCheckAnExit         : begin
                                  ClearEvent(Event);
                                  Event.What := evCommand;
                                  Event.Command := cmCheckWeight;
                                  PutEvent(Event);
                                  if CheckWeight then
                                    begin
                                      Event.What := evCommand;
                                      Event.Command := cmOK;
                                      PutEvent(Event);
                                    end
                                  else
                                    MessageBox(#3'Sum of weights <> 100', nil,
                                      mfInformation + mfOkButton);
                                end;
        end;
      end; { if }

    end; { Case }
  end; { if }

  with Event do
    if What=evBroadcast then
      begin
        case Command of
          cmBAddDataBatchObject : AddBatchDataObject;
          cmListItemSelected: ( Respond to broadcast from TSortedListBox )
            begin
              ClearEvent(Event);
              Event.What := evCommand;
              Event.Command := cmOK;
              PutEvent(Event);
            end;
        end;
      end;
    end;
  end;

```

```

    end;
  end;
  ClearEvent(Event);
end;

(*****
*)           Done
*)
(*****
destructor TListHDialog.Done;
begin
  if List <> nil then Dispose(List, Done);
  if DataCollection2 <> nil then Dispose(DataCollection2, Done);
  TDialog.Done;
end;

(*****
*)           Close
*)
(*****
procedure TListHDialog.Close;
begin
  ( TDialog.Close calls Valid and then Free. Before calling
    Free (which calls Done), tell all attached forms to close.
  )
  if Valid(cmClose) then
    begin
      ( Stop desktop video update in case there are scores of attached forms )
      Desktop^.Lock;
      Message(Desktop, evBroadcast, cmCloseForm, @Self);
      Desktop^.Unlock;
      Free;
    end;
  end;

  (*****
*)           Fill Blanks
*)
  (*****
  Procedure TListHDialog.FillBlanks(var S:String; StrLength:Integer);
    var j,l : integer;
    begin
      j:=length(s);
      l:=StrLength;
      if j < l then
        begin
          while j < l do
            begin
              inc(j);
              s[j] := chr(32);
            end;
          end;
          if j > l then j:=l;
          s[0]:=chr(l);
        end; { FillBlanks }
      end;
    end; { Unit }

```



```

{*****}
{
{   Filename : PkList1.pas   Author K G Mercer }
{   Date    : September 1993 }
{
{   Objects  : TRecD5        }
{             : TDataCollection5 }
{             : TPickListKeyBox }
{             : TPickDialog1 }
{
{   Other    : PInteger      - integer ptr }
{             : TransferWithCode - record }
{             : TRecD2       - record }
{*****}

```

```
unit PkList1;
```

```
($F+,O+,X+,S-,D+)
```

```
interface
```

```
uses
```

```
Dos, Objects, Memory, Drivers, Views, Dialogs, StdDlg, CmdnList;
```

```
type
```

```
PInteger = ^Integer;
```

```
PRecD5 = ^TRecD5;
```

```
TRecD5 = object(TObject)
```

```
  Name1 : PString;
```

```
  PCode : PInteger;
```

```
  constructor Init(Name : String ; NewCode : Integer);
```

```
  destructor Done; virtual;
```

```
end;
```

```
PDataCollection5 = ^TDataCollection5;
```

```
TDataCollection5 = object(TSortedCollection)
```

```
  function KeyOf(Item: Pointer): Pointer; virtual;
```

```
  function Compare(Key1, Key2: Pointer): Integer; virtual;
```

```
end;
```

```
PPickListKeyBox = ^TPickListKeyBox;
```

```
TPickListKeyBox = object(TSortedListBox)
```

```
  function GetText(Item: Integer; maxlength:integer): String; virtual;
```

```
  procedure GetData(var Rec);virtual;
```

```
  function DataSize:Word;virtual;
```

```
end;
```

```
PPickDialog1 = ^TPickDialog1;
```

```
TPickDialog1 = object(TDialog)
```

```
  DataCollection5 : PDataCollection5;
```

```
  List            : PPickListKeyBox;
```

```
  IsValid         : Boolean;
```

```
  Modified        : Boolean;
```

```
  constructor Init(Collection:PDataCollection5);
```

```
  destructor Done; virtual;
```

```
  procedure Close; virtual;
```

```
  procedure HandleEvent(var Event: TEvent); virtual;
```

```
end;
```

```

TransferWithCode = record
  Listname : PDataCollection5;
  Listfocused : Integer;
  RecCode   : Integer;
end;

```

```

TRecD2 = record
  Name : String;
  Code : String;
end;

```

```
function FileExists(Name: PathStr): Boolean;
```

```
implementation
```

```
uses App, MsgBox;
```

```
function FileExists(Name: PathStr): Boolean;
```

```
var
```

```
  SR: SearchRec;
```

```
begin
```

```
  FindFirst(Name, 0, SR);
```

```
  FileExists := DosError = 0;
```

```
end;
```

```
{*****}
```

```
{* TRecD5 *
```

```
{*****}
```

```
constructor TRecD5.Init(Name : String ; NewCode : Integer);
```

```
begin
```

```
  Name1 := NewStr(Name);
```

```
  New(PCode);
```

```
  PCode^ := NewCode;
```

```
end;
```

```
destructor TRecD5.Done;
```

```
begin
```

```
  DisposeStr(Name1);
```

```
  Dispose(PCode);
```

```
end;
```

```
{*****}
```

```
{* TDataCollection5 *
```

```
{*****}
```

```
function TDataCollection5.KeyOf(Item: Pointer): Pointer;
```

```
begin
```

```
  KeyOf := PRecD5(Item)^.Name1;
```

```
end;
```

```
function TDataCollection5.Compare(Key1, Key2: Pointer): Integer;
```

```
begin
```

```
  if PString(Key1)^ = PString(Key2)^ then
```

```
    Compare := 0
```

```
  else if PString(Key1)^ < PString(Key2)^ then
```

```
    Compare := -1
```

```
  else
```

```
    Compare := 1;
```

```

end;

(*****
*)
*)
TPickListKeyBox
*)
(*****
*)
TPickListKeyBox
*)

function TPickListKeyBox.DataSize:Word;
begin
    DataSize:= SizeOf(List)+SizeOf(Word)+SizeOf(Integer);
end;

procedure TPickListKeyBox.GetData(var Rec);
type
    ZapRec = record
        listname      : PCollection;
        listfocused   : integer;
        RecCode       : integer;
    end;
var
    R : ZapRec Absolute Rec; { Assign same address as Rec }
begin
    R.ListName := List;
    R.ListFocused := Focused;
    R.RecCode := PRecD5(List^.At(Focused)).PCode;
end;

function TPickListKeyBox.GetText(Item: Integer;maxlength:integer): String;
var
    S: String;
begin
    GetText := PString(PRecD5(List^.At(Item)).Name1)^;
end;

(*****
*)
*)
TPickDialog1
*)
(*****
*)
TPickDialog1
*)
(*****
*)
Init
*)
(*****
*)
constructor TPickDialog1.Init(Collection:PDataCollection5);
const
    FormX = 45;      { form x position }
    FormY = 5;        { form y position }
    FormWd = 26;      { form width }
    FormHt = 15;      { form length }
    ListX = 1;
    ListY = 1;
    ListWd = 23;
    ListHt = 13;
)
var
    R: TRect;
    SB: PScrollBar;
    Y: Integer;
    D: DirStr;
    N: String;
    E: ExtStr;

```

```

ListHt : Integer;
begin
    ListHt:=Collection^.Count;
    if ListHt > 15 then ListHt:=10;
    R.Assign(FormX, FormY, FormX + FormWd, FormY + ListHt + 2);
    n:='Batch History';
    TDialog.Init(R, N);

    { Read data collection into memory }

    DataCollection5 := Collection;
    if DataCollection5 <> nil then
    begin
        { Loaded successfully: build PickDialog1 dialog }

        { Scrollbar }
        R.Assign(ListX + ListWd, ListY, ListX + ListWd + 1, ListY + ListHt);
        SB := New(PScrollBar, Init(R));
        Insert(SB);

        { List box }
        R.Assign(ListX, ListY, ListX + ListWd, ListY + ListHt);
        List := New(PPickListKeyBox, Init(R, 1, SB));
        List^.NewList(DataCollection5);
        Insert(List);

        SelectNext(False);      { Select first field }
        IsValid := True;
    end;
end;

(*****
*)
*)
Handle Event
*)
(*****
*)
procedure TPickDialog1.HandleEvent(var Event: TEvent);
var
    DataWanted : Pointer;
begin
    { Respond to ENTER button }
    if (Event.What = evKeyDown) and (Event.KeyCode = kbEnter) then
    begin
        ClearEvent(Event);
        Event.What := evCommand;
        Event.Command := cmOK;
        PutEvent(Event);
    end;
}
    { Respond to CANCEL button and ESC }
    if (Event.What = evKeyDown) and (Event.KeyCode = kbEsc) then
    begin
        ClearEvent(Event);
        Event.What := evCommand;
        Event.Command := cmCancel;
        PutEvent(Event);
    end;
end;

```

```

    TDialog.HandleEvent(Event);

with Event do
  if What=evCommand then
  begin
    case Command of
      cmSelect : begin
                    ClearEvent(Event);
                    Event.What := evCommand;
                    Event.Command := cmOK;
                    PutEvent(Event);
                  end;
    end; { Case }
    ClearEvent(Event);
  end; { if }

with Event do
  if What=evBroadcast then
  begin
    case Command of
      cmListItemSelected: { Respond to broadcast from TSortedListBox }
      begin
        ClearEvent(Event);
        Event.What := evCommand;
        Event.Command := cmOK;
        PutEvent(Event);
      end;
    end; { case }
  end;
end;

(*****
*)           Done           *)
(*****)
destructor TPickDialog1.Done;
begin
  if List <> nil then Dispose(List, Done);
  if DataCollection5 <> nil then Dispose(DataCollection5, Done);
  TDialog.Done;
end;

(*****
*)           Close          *)
(*****)
procedure TPickDialog1.Close;
begin
  { TDialog.Close calls Valid and then Free. Before calling
    Free (which calls Done), tell all attached forms to close.
  }
  if Valid(cmClose) then
  begin
    { Stop desktop video update in case there are scores of attached forms }
    Desktop^.Lock;
    Message(Desktop, evBroadcast, cmCloseForm, @Self);
    Desktop^.Unlock;
    Free;
  end;
end;

end. { Unit }

```

```

{*****}
{
  Program for compiling and testing the
  TExpertObject stream
}
{
  Filename : StrmComp.pas   Author K G Mercer
  Date      : September 1993
}
{*****}

program StrmComp;

uses
  Objects, StrmUnit, Crt;

var
  ExpertList   : PCollection;
  ExpertStream : TBufStream;
  FileName     : String[12];

begin
  ClrScr;
  ExpertObjectStreamRegistration;
  Writeln('Starting memory =', Memavail);

  { Make the collection and display it }
  MakeExpertCollection(ExpertList);      { Generate and collect figures }
  Writeln('Memory after loading =', Memavail);
  WriteApps(ExpertList);                 { Use iterator to draw all }

  FileName := 'ExptList.Stm';
  { Put the collection in a stream on disk }
  ExpertStream.Init(FileName, stCreate, 1024);
  ExpertStream.Put(ExpertList);           { Output collection }
  ExpertStream.Done;                       { Shut down stream }
  Writeln('Memory after closing stream =', Memavail);

  Readln;                                 { Pause to view figures }
  { Clean up }
  Dispose(ExpertList, Done);               { Delete collection }
  Writeln('Memory after closing collection =', Memavail);

  with ExpertStream do
  begin
    Init(FileName, stOpen, 1024);          { Open stream }
    ExpertList := PCollection(Get);         { Load collection }
    Done;                                   { Shut down stream }
    if Status <> 0 then                       { Check for error }
      Abort('Error loading Orchard.stm');
  end;

  WriteApps(ExpertList);                   { Use iterator to draw all }
  Dispose(ExpertList, Done);               { Delete collection }
end.

```

```
{*****}  
{  
{ Program for listing the TExpertObject stream }  
{  
{ Filename : StrmOut.pas Author K G Mercer }  
{ Date : September 1993 }  
{  
{*****}  
  
program StrmOut;  
  
uses  
    Objects, StrmUnit, Crt;  
  
var  
    ExpertList : PCollection;  
    ExpertStream : TBufStream;  
    FileName : String[12];  
  
begin  
    ClrScr;  
    ExpertObjectStreamRegistration;  
  
    FileName := 'ExptList.Stm';  
  
    with ExpertStream do  
    begin  
        Init(FileName, stOpen, 1024); { Open stream }  
        ExpertList := PCollection(Get); { Load collection }  
        Done; { Shut down stream }  
        if Status <> 0 then { Check for error }  
            Abort('Error loading Orchard.stm');  
    end;  
  
    WriteApps(ExpertList); { Use iterator to draw all }  
    Dispose(ExpertList, Done); { Delete collection }  
end.
```

```

{*****}
{
{   Unit for stream object definition and streams   }
{
{   Filename : StrmUnit.pas   Author K G Mercer   }
{   Date      : September 1993   }
{
{   Objects   : TExpertObject   }
{               : TBatchObject   }
{               : TBatchDataObject   }
{               : TBatchStream   }
{
{   Other      : PInteger         - integer   }
{
{*****}

```

```

unit StrmUnit;
($R-)

```

```

interface

```

```

uses
  Objects;

```

```

{ ***** }
{ ***** Objects ***** }
{ ***** }

```

```

type

```

```

  PInteger = ^Integer;

```

```

PExpertObject = ^TExpertObject;
TExpertObject = object(TObject)
  Name          : PString;
  ProgName      : PString;
  RuleBaseName  : PString;
  Descrip1      : PString;
  Descrip2      : PString;
  Descrip3      : PString;
  PCode         : PInteger;
  constructor Init(NewN,NewP,NewRBN,NewDes1,NewDes2,NewDes3: String;
                  NewCode : Integer);
  constructor Load(var S: TStream);
  procedure WriteContents;virtual;
  procedure Store(var S: TStream);
  destructor Done; virtual;
end;

```

```

PBatchObject = ^TBatchObject;
TBatchObject = object(TObject)
  BatchName      : PString;
  BatchCallName  : PString;
  PCode          : PInteger;
  constructor Init(NewBN,NewBCN : String;NewCode : Integer);
  constructor Load(var S: TStream);
  procedure WriteContents;virtual;
  procedure Store(var S: TStream);
  destructor Done; virtual;
end;

```

```

PBatchDataObject = ^TBatchDataObject;
TBatchDataObject = object(TObject)
  ESysName       : PString;
  ESysDescrip    : PString;
  ProgName       : PString;
  RBNName        : PString;
  RBCode         : PInteger;
  PWeight        : PInteger;
  PCode          : PInteger;
  constructor Init(NewESN,NewESD,NewPN,NewRBN : String;NewRBC,NewWeight,NewCode
: Integer);
  constructor Load(var S: TStream);
  procedure WriteContents;virtual;
  procedure Store(var S: TStream);
  destructor Done; virtual;
end;

```

```

PBatchStream = ^TBatchStream;
TBatchStream = object(TBufStream)
  procedure Error(Code,Info : Integer);virtual;
end;

```

```

const

```

```

  RExpertObject: TStreamRec = (
    ObjType: 200;
    VmtLink: Ofs(TypeOf(TExpertObject)^);
    Load: @TExpertObject.Load;
    Store: @TExpertObject.Store);

```

```

  RBatchObject: TStreamRec = (
    ObjType: 201;
    VmtLink: Ofs(TypeOf(TBatchObject)^);
    Load: @TBatchObject.Load;
    Store: @TBatchObject.Store);

```

```

  RBatchDataObject: TStreamRec = (
    ObjType: 202;
    VmtLink: Ofs(TypeOf(TBatchDataObject)^);
    Load: @TBatchDataObject.Load;
    Store: @TBatchDataObject.Store);

```

```

procedure ExpertObjectStreamRegistration;
procedure Abort(Msg: String);
procedure MakeExpertCollection(var List: PCollection);
procedure WriteApps(C: PCollection);

```

```

procedure WriteBatchObjects(C: PCollection);
procedure MakeBatchCollection(var List: PCollection);

```

```

procedure WriteBatchDataObjects(C: PCollection);
procedure MakeBatchDataCollection(var List1,List2,List3,List4,List5,List6
: PCollection);

```

```

{ ***** }
{ ***** implementation ***** }
{ ***** }
{ ***** }
{ ***** }
{ ***** TBatchStream ***** }

```

```

( ***** )
procedure TBatchStream.Error(Code,Info :Integer);
begin
  Writeln('StreamError : ',Code,' Info : ',info);
  Halt(1);
end;

( ***** )
( ***** TExpertObject ***** )
( ***** )

constructor TExpertObject.Init(NewN,NewP,NewRBN,NewDes1,NewDes2,NewDes3: String;
  NewCode : Integer);
begin
  Name      := NewStr(NewN);
  ProgName  := NewStr(NewP);
  RuleBaseName := NewStr(NewRBN);
  Descrip1  := NewStr(NewDes1);
  Descrip2  := NewStr(NewDes2);
  Descrip3  := NewStr(NewDes3);
  New(PCode);
  PCode^    := NewCode;
end;

( ***** )

procedure TExpertObject.WriteContents;
begin
  Writeln(' ',
    Name^,'', :20-Length(Name^),
    ProgName^,'', :10-Length(ProgName^),
    RuleBaseName^,'', :15-Length(RuleBaseName^),
    Descrip1^,'', :8-Length(Descrip1^),
    Descrip2^,'', :8-Length(Descrip2^),
    Descrip3^,'', :8-Length(Descrip3^),
    PCode^:5);
end;

( ***** )

constructor TExpertObject.Load(var S: TStream);
var
  NameSize,ProgNameSize,RuleBaseNameSize,Descrip1Size,Descrip2Size,
  Descrip3Size: Word;
begin
  S.Read(NameSize ,SizeOf(NameSize ));
  S.Read(ProgNameSize ,SizeOf(ProgNameSize ));
  S.Read(RuleBaseNameSize ,SizeOf(RuleBaseNameSize ));
  S.Read(Descrip1Size ,SizeOf(Descrip1Size ));
  S.Read(Descrip2Size ,SizeOf(Descrip2Size ));
  S.Read(Descrip3Size ,SizeOf(Descrip3Size ));

  GetMem(Name ,NameSize );
  GetMem(ProgName ,ProgNameSize );
  GetMem(RuleBaseName ,RuleBaseNameSize );
  GetMem(Descrip1 ,Descrip1Size );
  GetMem(Descrip2 ,Descrip2Size );
  GetMem(Descrip3 ,Descrip3Size );

```

```

  S.Read(Name^ ,NameSize );
  S.Read(ProgName^ ,ProgNameSize );
  S.Read(RuleBaseName^ ,RuleBaseNameSize );
  S.Read(Descrip1^ ,Descrip1Size );
  S.Read(Descrip2^ ,Descrip2Size );
  S.Read(Descrip3^ ,Descrip3Size );

  New(PCode);
  S.Read(PCode^ ,SizeOf(Integer ));
end;

( ***** )

procedure TExpertObject.Store(var S: TStream);
var
  NameSize,ProgNameSize,RuleBaseNameSize,Descrip1Size,Descrip2Size,
  Descrip3Size: Word;
begin
  NameSize := Length(Name^+'1');
  ProgNameSize := Length(ProgName^+'1');
  RuleBaseNameSize := Length(RuleBaseName^+'1');
  Descrip1Size := Length(Descrip1^+'1');
  Descrip2Size := Length(Descrip2^+'1');
  Descrip3Size := Length(Descrip3^+'1');

  S.Write(NameSize ,SizeOf(NameSize ));
  S.Write(ProgNameSize ,SizeOf(ProgNameSize ));
  S.Write(RuleBaseNameSize ,SizeOf(RuleBaseNameSize ));
  S.Write(Descrip1Size ,SizeOf(Descrip1Size ));
  S.Write(Descrip2Size ,SizeOf(Descrip2Size ));
  S.Write(Descrip3Size ,SizeOf(Descrip3Size ));

  S.Write(Name^ ,NameSize );
  S.Write(ProgName^ ,ProgNameSize );
  S.Write(RuleBaseName^ ,RuleBaseNameSize );
  S.Write(Descrip1^ ,Descrip1Size );
  S.Write(Descrip2^ ,Descrip2Size );
  S.Write(Descrip3^ ,Descrip3Size );

  S.Write(PCode^ ,SizeOf(Integer ));
end;

( ***** )

destructor TExpertObject.Done;
begin
  DisposeStr(Name);
  DisposeStr(ProgName);
  DisposeStr(RuleBaseName);
  DisposeStr(Descrip1);
  DisposeStr(Descrip2);
  DisposeStr(Descrip3);
  Dispose(PCode);
end;

( ***** )
( ***** TBatchObject ***** )
( ***** )

constructor TBatchObject.Init(NewBN,NewBCN : String;NewCode : Integer);

```

```

begin
  BatchName      := NewStr(NewBN);
  BatchCallName  := NewStr(NewBCN);
  New(PCode);
  PCode^        := NewCode;
end;

( ***** )

procedure TBatchObject.WriteContents;
begin
  Writeln(' ',
    BatchName^,' ' :20-Length(BatchName^),
    BatchCallName^,' ' :20-Length(BatchName^),
    PCode^:5);
end;

( ***** )

constructor TBatchObject.Load(var S: TStream);
var
  BatchNameSize,BatchCallNameSize : Word;
begin
  S.Read(BatchNameSize,SizeOf(BatchNameSize));
  S.Read(BatchCallNameSize,SizeOf(BatchCallNameSize));
  GetMem(BatchName, BatchNameSize);
  GetMem(BatchCallName, BatchCallNameSize);
  S.Read(BatchName^,BatchNameSize);
  S.Read(BatchCallName^,BatchCallNameSize);
  New(PCode);
  S.Read(PCode^,SizeOf(Integer));
end;

( ***** )

procedure TBatchObject.Store(var S: TStream);
var
  BatchNameSize,BatchCallNameSize : Word;
begin
  BatchNameSize := Length(BatchName^+'1');
  BatchCallNameSize := Length(BatchCallName^+'1');
  S.Write(BatchNameSize,SizeOf(BatchNameSize));
  S.Write(BatchCallNameSize,SizeOf(BatchCallNameSize));
  S.Write(BatchName^,BatchNameSize);
  S.Write(BatchCallName^,BatchCallNameSize);
  S.Write(PCode^,SizeOf(Integer));
end;

( ***** )

destructor TBatchObject.Done;
begin
  DisposeStr(BatchName);
  DisposeStr(BatchCallName);
  Dispose(PCode);
end;

( ***** )
( ***** TBatchDataObject ***** )
( ***** )

```

```

constructor TBatchDataObject.Init(NewESN,NewESD,NewPN,NewRBN : String;
  NewRBC,NewWeight,NewCode : Integer);
begin
  ESysName      := NewStr(NewESN);
  ESysDescrip    := NewStr(NewESD);
  ProgName      := NewStr(NewPN);
  RBName        := NewStr(NewRBN);
  New(RBCode);
  RBCode^       := NewRBC;
  New(PWeight);
  PWeight^      := NewWeight;
  New(PCode);
  PCode^        := NewCode;
end;

( ***** )

procedure TBatchDataObject.WriteContents;
begin
  Writeln(' ',
    ESysName^,' ' :15-Length(ESysName^),
    ESysDescrip^,' ' :15-Length(ESysDescrip^),
    ProgName^,' ' :15-Length(ProgName^),
    RBName^,' ' :15-Length(RBName^),
    RBCode^:5,
    PWeight^:5,
    PCode^:5);
end;

( ***** )

constructor TBatchDataObject.Load(var S: TStream);
var
  ESysNameSize,ESysDescripSize,ProgNameSize,RBNameSize : Word;
begin
  S.Read(ESysNameSize,SizeOf(ESysNameSize));
  S.Read(ESysDescripSize,SizeOf(ESysDescripSize));
  S.Read(ProgNameSize,SizeOf(ProgNameSize));
  S.Read(RBNameSize,SizeOf(RBNameSize));

  GetMem(ESysName,ESysNameSize);
  GetMem(ESysDescrip,ESysDescripSize);
  GetMem(ProgName,ProgNameSize);
  GetMem(RBName,RBNameSize);

  S.Read(ESysName^,ESysNameSize);
  S.Read(ESysDescrip^,ESysDescripSize);
  S.Read(ProgName^,ProgNameSize);
  S.Read(RBName^,RBNameSize);

  New(RBCode);
  S.Read(RBCode^,SizeOf(Integer));
  New(PWeight);
  S.Read(PWeight^,SizeOf(Integer));
  New(PCode);
  S.Read(PCode^,SizeOf(Integer));
end;

( ***** )

```



```

procedure TBatchDataObject.Store(var S: TStream);
var
  ESysNameSize, ESysDescripSize, ProgNameSize, RBNameSize : Word;
begin
  ESysNameSize      := Length(ESysName^+'1');
  ESysDescripSize   := Length(ESysDescrip^+'1');
  ProgNameSize      := Length(ProgName^+'1');
  RBNameSize        := Length(RBName^+'1');

  S.Write(ESysNameSize, SizeOf(ESysNameSize));
  S.Write(ESysDescripSize, SizeOf(ESysDescripSize));
  S.Write(ProgNameSize, SizeOf(ProgNameSize));
  S.Write(RBNameSize, SizeOf(RBNameSize));

  S.Write(ESysName^, ESysNameSize);
  S.Write(ESysDescrip^, ESysDescripSize);
  S.Write(ProgName^, ProgNameSize);
  S.Write(RBName^, RBNameSize);

  S.Write(RBCode^, SizeOf(Integer));
  S.Write(PWeight^, SizeOf(Integer));
  S.Write(PCode^, SizeOf(Integer));
end;

( ***** )

destructor TBatchDataObject.Done;
begin
  DisposeStr(ESysName);
  DisposeStr(ESysDescrip);
  DisposeStr(ProgName);
  DisposeStr(RBName);
  Dispose(RBCode);
  Dispose(PWeight);
  Dispose(PCode);
end;

( ***** )
( ***** General ***** )
( ***** )

procedure ExpertObjectStreamRegistration;
begin
  RegisterType(RCollection);
  RegisterType(RExpertObject);
  RegisterType(RBatchObject);
  RegisterType(RBatchDataObject);
end;

( ***** )

procedure Abort(Msg: String);
begin
  Writeln;
  Writeln(Msg);
  Writeln('Program aborting');
  Halt(1);
end;

```

```

( ***** )

procedure WriteApps(C: PCollection);
begin
  procedure CallWrite(P: PExpertObject); far;
  begin
    P^.WriteContents;
  end;

  begin ( WriteAll )
    C^.ForEach(@CallWrite);
  end;

( ***** )

procedure WriteBatchObjects(C: PCollection);
begin
  procedure CallWrite(P: PBatchObject); far;
  begin
    P^.WriteContents;
  end;

  begin ( WriteAll )
    C^.ForEach(@CallWrite);
  end;

( ***** )

procedure WriteBatchDataObjects(C: PCollection);
begin
  procedure CallWrite(P: PBatchDataObject); far;
  begin
    P^.WriteContents;
  end;

  begin ( WriteAll )
    C^.ForEach(@CallWrite);
  end;

( ***** )

procedure MakeExpertCollection(var List: PCollection);
var
  i: Integer;
  P: PExpertObject;
begin
  ( Initialize collection to hold 12 elements first, then grow by 5's )
  List := New(PCollection, Init(12, 5));

  P := New(PExpertObject, Init('june', 'Spiders', 'Hand', 'Foliar', 'Doom', 'hgffgh', 1));
  List^.Insert(P);
  P := New(PExpertObject, Init('june', 'Spiders', 'Hand', 'Foliar', 'Doom', 'hgffgh', 2));
  List^.Insert(P);
  P := New(PExpertObject, Init('june', 'Spiders', 'Hand', 'Foliar', 'Doom', 'hgffgh', 3));
  List^.Insert(P);
  P := New(PExpertObject, Init('june', 'Spiders', 'Hand', 'Foliar', 'Doom', 'hgffgh', 4));
  List^.Insert(P);
end;

```

```

P := New(PExpertObject, Init('june','Spiders','Hand','Foliar','Doom','hgffgh',5));
List^.Insert(P);
P := New(PExpertObject, Init('june','Spiders','Hand','Foliar','Doom','hgffgh',6));
List^.Insert(P);
P := New(PExpertObject, Init('june','Spiders','Hand','Foliar','Doom','hgffgh',7));
List^.Insert(P);
P := New(PExpertObject, Init('june','Spiders','Hand','Foliar','Doom','hgffgh',8));
List^.Insert(P);
P := New(PExpertObject, Init('june','Spiders','Hand','Foliar','Doom','hgffgh',9));
List^.Insert(P);
P := New(PExpertObject, Init('june','Spiders','Hand','Foliar','Doom','hgffgh',10));
List^.Insert(P);
P := New(PExpertObject, Init('june','Spiders','Hand','Foliar','Doom','hgffgh',11));
List^.Insert(P);
end;

```

```

( ***** )

```

```

procedure MakeBatchCollection(var List: PCollection);

```

```

var
  i: Integer;
  P: PBatchObject;
begin
  { Initialize collection to hold 12 elements first, then grow by 5's }
  List := New(PCollection, Init(12, 5));

  P := New(PBatchObject, Init('DuNoon','DuNoon',1));
  List^.Insert(P);
  P := New(PBatchObject, Init('Milnerton','Milnerton',2));
  List^.Insert(P);
  P := New(PBatchObject, Init('SandyBay','SandyBay',3));
  List^.Insert(P);
  P := New(PBatchObject, Init('Clifton','Clifton',4));
  List^.Insert(P);
  P := New(PBatchObject, Init('SeaPoint','SeaPoint',5));
  List^.Insert(P);
  P := New(PBatchObject, Init('Somerset West','Somerset West',6));
  List^.Insert(P);
end;

```

```

( ***** )

```

```

procedure MakeBatchDataCollection(var List1,List2,List3,List4,List5,List6
: PCollection);

```

```

var
  i: Integer;
  P: PBatchDataObject;
begin
  { Initialize collection to hold 12 elements first, then grow by 5's }

  List1 := New(PCollection, Init(12, 5));
  P := New(PBatchDataObject, Init('ESystem1','ESystem1','a','a',2,33,1));
  List1^.Insert(P);

```

```

P := New(PBatchDataObject, Init('ESystem2','ESystem2','a','a',3,33,2));
List1^.Insert(P);

```

```

List2 := New(PCollection, Init(12, 5));
P := New(PBatchDataObject, Init('ESystem3','ESystem3','a','a',2,33,2));
List2^.Insert(P);
P := New(PBatchDataObject, Init('ESystem4','ESystem4','a','a',3,33,2));
List2^.Insert(P);

```

```

List3 := New(PCollection, Init(12, 5));
P := New(PBatchDataObject, Init('ESystem3','Sandybay','a','a',2,33,3));
List3^.Insert(P);

```

```

List4 := New(PCollection, Init(12, 5));
P := New(PBatchDataObject, Init('ESystem4','Clifton','a','a',2,33,4));
List4^.Insert(P);

```

```

List5 := New(PCollection, Init(12, 5));
P := New(PBatchDataObject, Init('ESystem5','Sea Point','a','a',2,33,5));
List5^.Insert(P);

```

```

List6 := New(PCollection, Init(12, 5));
P := New(PBatchDataObject, Init('ESystem6','Somerset West','a','a',3,33,6));
List6^.Insert(P);

```

```

end;

```

```

end. ( Unit )

```

```

(*****)
(      Unit for managing the overall system      )
(      )
(      Filename : Thesis2.pas      Author K G Mercer      )
(      Date      : September 1993      )
(      )
(      Objects   : TMyBack      )
(      : TFormApp      )
(*****)

( default      enabled is '+' )
{$A+} (+) ( Word Align 16-bit and larger data )
{$B-} (+) ( Short Circuit boolean expressions )
{$F+} (-) ( Far code switch, '+' forces all functions + procedures to use far calls )
{$G-} ( ) ( )
{$I+} (+) ( Check for input /output errors )
{$L-} (+) ( Link buffer in memory )
{$O+} (-) ( Overlay switch, '+' program uses overlays )
{$R-} (-) ( Check for range errors/ verify object instance VMT link )
{$S-} (+) ( Check for stack overflow )
{$X+} (-) ( Extended Syntax )
($M,16000,0,100000)

program thesis2;

uses
  Dos, Crt, Overlay, Objects, Memory, Drivers, Views, Menus, Dialogs, HistList,
  StdDlg, App, MsgBox, Editors, DataColl, ListDlg, Fields,
  TBackMen, ColBacku, ColorSel, Gadgets, GrCrumble,
  GrDial, GrTextU,
  ( Expert System related Units )
  CmdnList, CntrlUnt, ViewWin2, ViewWinF, ExptMan, StrmUnit, General2, ListFDlg,
  ListHDlg,
  ( Graphics )
  GraphApp, Graph,
  ( Help )
  HelpFile, Help;

var
  Ticks : LongInt ABSOLUTE $0040:$006C;

type
  PMyBack = ^TMyBack;
  TMyBack = OBJECT(ColBackGround)
    PROCEDURE InitMenuBox; Virtual;
  END;

  TFormApp = ^TFormApp;
  TFormApp = object(TApplication)
    PMBG      : PMyBack;
    Clock     : PClockView;
    Heap      : PHeapView;
    ScreenSave : LongInt;
    SaveInterval : Word;
    constructor Init;
    procedure GetEvent(var Event: TEvent); virtual;
    function GetPalette: PPalette; virtual;
  end;

```

```

  procedure HandleEvent(var Event: TEvent); virtual;
  procedure InitMenuBar; virtual;
  procedure InitStatusLine; virtual;
  procedure BatchFiles;
  procedure Analysis;
  procedure ColorDialog;
  procedure OpenExpertSystems;
  procedure DoCrumDialog;
  procedure ToggleScreenSave;
  procedure Idle; virtual;
  procedure FillBlanks(var S:String; StrLength:Integer);
  destructor Done; virtual;
end;

var
  TInfoR1      : TInfoRec1;
  BatchList    : PCollection;
  BatchStream  : TBufStream;
  TransWCode   : TransferWithCode;
  RezName      : String[12];
  BatchRez     : TResourceFile;

(***** Batch File Analysis *****)
(      Batch File Analysis      )
(*****)

procedure TFormApp.BatchFiles;
var
  PLFD          : PListFDlg;
  PLHD          : PListHDialog;
  Collection1    : PDataCollection1;
  Collection2    : PDataCollection2;
  Control1, Control2 : Word;
  CodeReturned, i, IndexNo : Integer;
  FoundBO       : PBatchObject;
  RetrieveKeyName : String;
  RetrieveBatchName : String;
  ESList        : PCollection;
  AddFirst      : Boolean;
  ESysName2, Weight2Str : String;
  Weight2, Code2 : Integer;
  ArrayStr      : String;
  TotalWeight   : Integer;

  procedure CallWrite1(P: PBatchObject); far;
  begin
    Collection1^.Insert(New(PRecD2, Init(P^.BatchName^, P^.PCode^)));
  end;

  function CodeMatch(Batch:PBatchObject):Boolean; far;
  begin
    CodeMatch := CodeReturned = Batch^.PCode^;
  end;

  procedure OpenRezFile;
  begin
    end;

begin
  ( Read in batch info from ResourceFile )

```

```

RezName := 'Batch.Rez';
BatchRez.Init(New(PBufStream, Init(RezName, stOpen, 1024)));
if BatchRez.Stream^.Status <> 0 then
    MessageBox(#3'Error loading Resourcefile', nil, mfInformation + mfOkButton
);

BatchList := PCollection(BatchRez.Get('KeyNames'));
BatchRez.Done;

Collection1 := New(PDataCollection1, Init(BatchList^.Count - 1, 5));

for i:=0 to BatchList^.Count - 1 do
begin
    CallWrite1(BatchList^.At(i));
end;

PLFD := New(PListFDDialog, Init(Collection1));
PLFD^.HelpCtx := hcBatchDataObjectSelect;
Control1 := Desktop^.ExecView(PLFD);
if Control1 <> cmCancel then
begin
    PLFD^.GetData(TransWCode);
    CodeReturned := TransWCode.RecCode;
end;
Dispose(PLFD, Done);

( Search for BatchObject with the code returned )

if Control1 <> cmCancel then
begin
    FoundBO := BatchList^.FirstThat(@CodeMatch);
    IndexNo := BatchList^.IndexOf(FoundBO);
    RetrieveBatchName := PBatchObject(BatchList^.At(IndexNo)).BatchName^;
    RetrieveKeyName := PBatchObject(BatchList^.At(IndexNo)).BatchCallName^;

    ( Read in batch info from ResourceFile )
    RezName := 'Batch.Rez';
    BatchRez.Init(New(PBufStream, Init(RezName, stOpen, 1024)));
    if BatchRez.Stream^.Status <> 0 then
        MessageBox(#3'Error loading stream', nil, mfInformation + mfOkButton);

    ESList := PCollection(BatchRez.Get(RetrieveKeyName));
    BatchRez.Done;

    Collection2 := New(PDataCollection2, Init(ESList^.Count - 1, 5));
    ( check to see whether list is empty then force entry of data )
    AddFirst := False;
    if ESList^.Count = 0 then
    begin
        AddFirst:=True;
        MessageBox(#3'List is Empty', nil, mfInformation + mfOkButton);
    end;

    TotalWeight:=0;
    for i:=0 to ESList^.Count - 1 do
    begin
        ESysName2 := PBatchDataObject(ESList^.At(i)).ESysName^;
        Weight2 := PBatchDataObject(ESList^.At(i)).PWeight^;
        Code2 := PBatchDataObject(ESList^.At(i)).PCode^;

```

```

        TotalWeight:=TotalWeight+Weight2;
        Str(Weight2, Weight2Str);
        FillBlanks(ESysName2, 29);
        FillBlanks(Weight2Str, 3);
        ArrayStr := ESysName2+' '+Weight2Str;

        Collection2^.Insert(New(PRecD1, Init(ArrayStr, Code2)));
    end;

    PLHD := New(PListHDialog, Init(Collection2, RetrieveBatchName, TotalWeight, Add
First));
    PLHD^.HelpCtx := hcBatchESSelect;
    Control2 := Desktop^.ExecView(PLHD);
    if Control2 <> cmCancel then
    begin
        PLHD^.GetData(TransWCode);
        CodeReturned := TransWCode.RecCode;
    end;
    Dispose(PLHD, Done);

end; ( cmCancel1 )

Dispose(BatchList, Done);
if Control1 <> cmCancel then Dispose(ESList, Done);
end;

{*****}
{               Result Analysis               }
{*****}

procedure TFormApp.Analysis;

label
    Checkout;
type
    String2 = String[2];
    String4 = String[4];
var
    PLFD                : PListFDDialog;
    Collection1         : PDataCollection1;
    Control1, Option    : Word;
    CodeReturned, i, IndexNo : Integer;
    FoundBO             : PBatchObject;
    RetrieveKeyName, Name : String;
    RetrieveBatchName    : String;
    ESList              : PCollection;
    AddFirst, Found     : Boolean;
    RBCode2, Y, X       : Integer;
    ArrayStr, ProgName2, RBName2 : String;
    TotalWeight, Error   : Integer;
    Outfile             : Text;
    ShowWindow          : PDemoWindow2;
    R                   : TRect;
    OneMember           : Member;
    PEditBDObject       : PBatchDataObject;
    ESysName2, NewBatchName, Line : String;
    ExpertNameFoundPtr   : PExpertObject;
    RBaseCodeStr, BatchNameStr2 : String2;
    BatchNameStr4       : String4;
    FileName            : String[121];

```

```

ResFile      : Text;
Position     : Array[1..3] of integer;
Names        : Array[1..20] of string;
Weights      : Array[1..20] of integer;
NamesG       : Array[1..20] of string[20];
Values       : Array[1..20,1..3] of real;
StrValues    : Array[1..20,1..3] of string[6];
StrValuesPlus : Array[1..20,1..3] of string[6];
FillType     : Array[1..20] of byte;
Value1Str,Value2Str,Value3Str : String;
Value10Str,Value20Str,Value30Str : String;
Value1,Value2,Value3 : Real;
ShowWindow  : PDemoWindow;
PlotX,PlotY,MaxX,MaxY : Integer;
YHeight,Weight : Integer;
BoxX,BoxY,NumExpertSys : Integer;
Pattern     : Word;
Event       : TEvent;
Total,TotalRVL : Real;

procedure CallWrite1(P: PBatchObject); far;
begin
  Collection1^.Insert(New(PRecD1, Init(P^.BatchName^,P^.PCode^)));
end;

function CodeMatch(Batch:PBatchObject):Boolean;far;
begin
  CodeMatch := CodeReturned = Batch^.PCode^;
end;

begin
  { Retrieve Batchlist from Resource File}
  RezName := 'Batch.Rez';
  BatchRez.Init(New(PBufStream,Init(RezName,stOpen,1024)));
  if BatchRez.Stream^.Status <> 0 then
    MessageBox(#3'Error loading Resourcefile', nil, mfInformation + mfOkButton);

  BatchList := PCollection(BatchRez.Get('KeyNames'));
  BatchRez.Done;

  Collection1 := New(PDataCollection1,Init(BatchList^.Count -1, 5));

  for i:=0 to BatchList^.Count -1 do
  begin
    CallWrite1(BatchList^.At(i));
  end;

  { Display list and Select BatchObject to Run }
  DisableCommands([cmDelBatchObject,cmAddBatchObject]);
  PLFD := New(PLFDDialog,Init(Collection1));
  PLFD^.HelpCtx := hcResultSelect;
  Control1 := Desktop^.ExecView(PLFD);
  if Control1 <> cmCancel then
  begin
    PLFD^.GetData(TransWCode);
    CodeReturned := TransWCode.RecCode;
  end;
  Dispose(PLFD, Done);

```

```

EnableCommands([cmDelBatchObject,cmAddBatchObject]);

{ Search for BatchObject with the code returned }

if Control1 <> cmCancel then
begin
  FoundBO := BatchList^.FirstThat(@CodeMatch);
  IndexNo := BatchList^.IndexOf(FoundBO);

  RetrieveBatchName := PBatchObject(BatchList^.At(IndexNo))^BatchName^;
  RetrieveKeyName := PBatchObject(BatchList^.At(IndexNo))^BatchCallName^;

  { Retrieve corresponding Expert System List from resource file }

  RezName := 'Batch.Rez';
  BatchRez.Init(New(PBufStream,Init(RezName,stOpen,1024)));
  if BatchRez.Stream^.Status <> 0 then
    MessageBox(#3'Error loading stream', nil, mfInformation + mfOkButton);

  ESList := PCollection(BatchRez.Get(RetrieveKeyName));
  BatchRez.Done;

  { Obtain first 4 and last 2 characters in RetrieveBatchName }
  if Length(RetrieveBatchName) >= 6 then
  begin
    BatchNameStr4 := Copy(RetrieveBatchName,1,4);
    BatchNameStr2 := Copy(RetrieveBatchName,Length(RetrieveBatchName)-1,2);
    NewBatchName := BatchNameStr4 + BatchNameStr2;
  end
  else NewBatchName := RetrieveBatchName;

  { Retrieve assessment information from each result file }

  for i:=0 to ESList^.Count -1 do
  begin
    { Retrieve rulebase name }
    PEditBDOObject := PBatchDataObject(ESList^.At(i));
    ESysName2 := PEditBDOObject^.ESysName^;
    Names[i+1] := ESysName2;
    NamesG[i+1] := ESysName2;

    RBCCode2 := PBatchDataObject(ESList^.At(i))^RBCCode^;
    Str(RBCCode2:2,RBaseCodeStr);
    if RBaseCodeStr[1]=' ' then RBaseCodeStr[1]:=chr(48);
    FileName := NewBatchName+RBaseCodeStr+'.res';
    Assign(ResFile,FileName);
    {$I-}
    Reset(ResFile);
    {$I+}

    if IoResult <> 0 then
    begin
      MessageBox('An error has occurred reading the '+
        #13+'result files. The problem is with'+#13+'rulebase : '+EsysNa
me2,
        nil, mfError + mfOkButton);
      MessageBox('Check to see if the Batchrun file '+
        #13+'has not been altered. Re-run the '+
        #13+'rulebase or analysis as required', nil, mfError + mfOkBut
ton);

```

```

    Goto Checkout;
end;

y:=1;
while not Eof(ResFile) do
begin
    ReadLn(ResFile,Line);
    if Pos('assessment',Line) <> 0 then
    begin
        ( read line with assessment char by char )
        for x := 1 to length(Line) do
        begin
            if Line[x] = ',' then
            begin
                Position[y] := x;
                inc(y);
            end;
        end;
        Value1Str:= Copy(Line,Position[1]+2,Position[2]-Position[1]-2);
        Value10Str := Value1Str;
        Value2Str:= Copy(Line,Position[2]+2,Position[3]-Position[2]-2);
        Value20Str := Value2Str;
        Value3Str:= Copy(Line,Position[3]+2,6);
        Value30Str := Value3Str;

        if Value10Str[1] <> '-' then Value10Str:='-'+Value10Str;
        if Value20Str[1] <> '-' then Value20Str:='-'+Value20Str;
        if Value30Str[1] <> '-' then Value30Str:='-'+Value30Str;

        StrValues[i+1,1] := Value1Str;
        StrValues[i+1,2] := Value2Str;
        StrValues[i+1,3] := Value3Str;

        StrValuesPlus[i+1,1] := Value10Str;
        StrValuesPlus[i+1,2] := Value20Str;
        StrValuesPlus[i+1,3] := Value30Str;

        error := 0;
        Val(Value1Str,Value1,error);
        Val(Value2Str,Value2,error);
        Val(Value3Str,Value3,error);
        if error <> 0 then MessageBox('Error with conversion',
                                   nil, mfError + mfOkButton);

        Values[i+1,1] := Value1;
        Values[i+1,2] := Value2;
        Values[i+1,3] := Value3;
    end;
end;
System.Close(ResFile);
end; (ESList.Count-1)

( Calculate Overall Assessment )

NumExpertSys := ESList.Count;
for i:=0 to ESList.Count-1 do
begin
    ( Retrieve rulebase weight )
    PEditBDOObject := PBatchDataObject(ESList.At(i));

```

```

    Weight := PEditBDOObject^.PWeight^;
    Weights[i+1] := Weight;
end;
Total:=0;
for i:= 1 to NumExpertSys do
begin
    Total := Total + (((Values[i,2]+1)/2)*(Weights[i]*NumExpertSys));
end;
( Convert back to RVL )
TotalRVL:=((Total/(NumExpertSys*100))*2)-1;

( Show Graphics )

FillType[1] := 1;
FillType[2] := 2;
FillType[3] := 3;
FillType[4] := 4;
FillType[5] := 5;
FillType[6] := 6;
FillType[7] := 7;
FillType[8] := 8;
FillType[9] := 9;
FillType[10] := 10;

( Start Graphics )
if not GraphicsStart then
    MessageBox(GraphErrorMsg(GraphResult) + '.', nil, mfError or mfOkButton)
else
begin
    MaxX := GetMaxX;
    MaxY := GetMaxY;
    SetViewport(0,0, MaxX - 1, MaxY - 1, ClipOff);

    ( Text )
    SetColor(LightBlue);
    SetTextStyle(SansSerifFont,0,4);
    OutTextXY(Trunc(MaxX/2-TextWidth('Batchrun Summary')/2), 0, 'Batchrun Summ
ary');
    SetColor(LightGreen);
    OutTextXY(Trunc(MaxX/2-TextWidth(RetrieveBatchName)/2), 35, RetrieveBatchN
ame);

    ( Lines )
    ( Boundary )
    SetColor(LightGray);
    Graph.Line(0,0,0,MaxY);
    Graph.Line(0,MaxY,MaxX,MaxY);
    Graph.Line(MaxX,MaxY,MaxX,0);
    Graph.Line(MaxX,0,0,0);

    SetLineStyle(SolidLn,0,ThickWidth);
    SetColor(Cyan);
    Graph.Line(Trunc(MaxX/2),100,Trunc(MaxX/2),MaxY-100);
    Graph.Line(0,MaxY-100,MaxX,MaxY-100);

    Pattern:=$1010;
    SetLineStyle(UserBitLn,Pattern,NormWidth);
    SetColor(Lightgray);
    (Vert)
    Graph.Line(Trunc(MaxX/2)-200,100,Trunc(MaxX/2)-200,MaxY-100);

```

```

Graph.Line(Trunc(MaxX/2)-100,100,Trunc(MaxX/2)-100,MaxY-100);
Graph.Line(Trunc(MaxX/2)+100,100,Trunc(MaxX/2)+100,MaxY-100);
Graph.Line(Trunc(MaxX/2)+200,100,Trunc(MaxX/2)+200,MaxY-100);

SetLineStyle(DashedLn,0,1);
SetColor(LightGray);
Graph.Line(Trunc(MaxX/2)-200,100,Trunc(MaxX/2)+200,100);

SetColor(White);
SetTextStyle(DefaultFont,0,1);
OutTextXY(Trunc(MaxX/2-200-TextWidth('Max.))/2,70,'Max.');
```

OutTextXY(Trunc(MaxX/2-200-TextWidth('Negative')/2),80,'Negative');
OutTextXY(Trunc(MaxX/2-200-TextWidth('Impact')/2),90,'Impact');
OutTextXY(Trunc(MaxX/2-TextWidth('Neutral')/2),80,'Neutral');
OutTextXY(Trunc(MaxX/2-TextWidth('Impact')/2),90,'Impact');
OutTextXY(Trunc(MaxX/2+200-TextWidth('Max.))/2,70,'Max.');
OutTextXY(Trunc(MaxX/2+200-TextWidth('Positive')/2),80,'Positive');
OutTextXY(Trunc(MaxX/2+200-TextWidth('Impact')/2),90,'Impact');

```

OutTextXY(Trunc(MaxX/2+210),200,'Individual');
OutTextXY(Trunc(MaxX/2+210),218,'Assessments');

SetColor(White);
SetLineStyle(SolidLn,0,1);
PlotX := Trunc(MaxX/2);
PlotY := 100;
YHeight:=Trunc((MaxY-100-110)/(NumExpertSys+1));
for i:= 1 to NumExpertSys do
begin
  Rectangle(PlotX,PlotY,PlotX+Trunc(Values[i,2]*200),PlotY+YHeight);
  SetFillStyle(FillType[i],i);
  if Values[i,2] > 0 then FloodFill(PlotX+1,PlotY+1,white)
  else if Values[i,2] < 0 then FloodFill(PlotX-1,PlotY+1,white)
  else if Values[i,2] = 0 then begin end;
  PlotY:=PlotY+YHeight;
end;

( Plot Overall Assessment )

PlotY:=PlotY+10;
Rectangle(PlotX,PlotY,PlotX+Trunc(TotalRVL*200),PlotY+YHeight);
SetFillStyle(FillType[1],3);
if Values[i,2] > 0 then FloodFill(PlotX+1,PlotY+1,white)
else if Values[i,2] < 0 then FloodFill(PlotX-1,PlotY+1,white)
else if Values[i,2] = 0 then begin end;

SetColor(White);
SetTextStyle(DefaultFont,0,1);
OutTextXY(Trunc(MaxX/2+210),PlotY+ 5,'Overall');
OutTextXY(Trunc(MaxX/2+210),PlotY+18,'Weighted');
OutTextXY(Trunc(MaxX/2+210),PlotY+31,'Assessment');

SetLineStyle(DashedLn,0,1);
SetColor(LightGray);
Graph.Line(Trunc(MaxX/2)-200,PlotY-5,Trunc(MaxX/2)+200,PlotY-5);

BoxX := 30;
BoxY := MaxY-100+8;
SetColor(White);
SetLineStyle(SolidLn,0,1);
```

```

SetTextStyle(DefaultFont,0,1);

for i:= 1 to NumExpertSys do
begin
  if i=5 then
  begin
    BoxX:=230;
    BoxY:=MaxY-100+8;
  end;
  if i=9 then
  begin
    BoxX:=440;
    BoxY:=MaxY-100+8;
  end;
  Rectangle(BoxX,BoxY,BoxX+20,BoxY+15);
  SetFillStyle(FillType[i],i);
  FloodFill(BoxX+1,BoxY+1,white);
  OutTextXY(BoxX+30,BoxY+5,NamesG[i]);
  BoxY:=BoxY+23;
end;

OutTextXY(Trunc(MaxX-TextWidth('Hit any key to Continue')-5),
          Trunc(MaxY-TextHeight('T')-5),'Hit any key to Continue');
{ Finish }
repeat
  GetKeyEvent(Event);
until Event.What <> evNothing;
GraphicsStop;
end;

( Display Result Data )

Assign(Resfile,'Results.txt');
Rewrite(Resfile);
Writeln(Resfile);
Writeln(Resfile,'
                                     Batchrun RVL Result Information
');
Writeln(Resfile,'
                                     eeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
');
Writeln(Resfile);
Writeln(Resfile,'
                                     Individual Assessment Results');
Writeln(Resfile,'
                                     RVL Val
ues');
Writeln(Resfile);
Writeln(Resfile,'
                                     Rulebase Name
                                     Lower
                                     Expect
ed
Upper ');
Writeln(Resfile,'
                                     aaaaaaaaaaaaaa
                                     aaaaaaaaaaaaaa
aaaaaaaaaaaaaa ');
Writeln(Resfile);
for i:=1 to ESList^.Count do
begin
  Writeln(Resfile,'
                  ',Names[i],':38-Length(Names[i]),
                  StrValuesPlus[i,1],':10-Length(StrValuesPlus[i,1]),
                  StrValuesPlus[i,2],':10-Length(StrValuesPlus[i,2]),
                  StrValuesPlus[i,3]);
end;
Writeln(Resfile);
Writeln(Resfile);
Writeln(Resfile);
```

```

Writeln(Resfile,'      Overall Weighted Assessment : RVL = ',TotalRvl:5:3);
Writeln(Resfile);

Writeln(Resfile);Writeln(Resfile);Writeln(Resfile);
System.Close(Resfile);

( Show scroll window )

Name := 'Results.txt';
WinCount := 1;
Desktop^.GetExtent(R);
Show1Window := New(PDemoWindow, Init(R, 'Batchrun Results', WinCount, Name))
;
Desktop^.Insert(Show1Window);
CheckOut:
Dispose(ESList,Done);
end; ( Control 1 )
end;

(*****
(*          Fill Blanks          *)
*****)
Procedure TFormApp.FillBlanks(var S:String; StrLength:Integer);
var j,l : integer;
begin
  j:=length(s);
  l:=StrLength;
  if j < l then
  begin
    while j < l do
    begin
      inc(j);
      s[j] := chr(32);
    end;
  end;
  if j > l then j:=l;
  s[0]:=chr(l);
end; ( FillBlanks )

(*****
(          CalcHelpName          )
*****)
( CalcHelpName )
function CalcHelpName: PathStr;
var
  EXEName: PathStr;
  Dir: DirStr;
  Name: NameStr;
  Ext: ExtStr;
begin
  if Lo(DosVersion) >= 3 then EXEName := ParamStr(0)
  else EXEName := FSearch('Thesis2.exe', GetEnv('PATH'));
  FSplit(EXEName, Dir, Name, Ext);
  if Dir[Length(Dir)] = '\' then Dec(Dir[0]);
  CalcHelpName := FSearch('Help.Hlp', Dir);
end;

(*****

```

```

(          BackGround Menu          )
(*****
procedure TMyBack.InitMenuBox;
var R : TRect;
begin
  GetExtent(R);
  R.Assign(R.B.X-1, R.B.Y-1, R.B.X, R.B.Y);
  PM := New(PMenuBox, Init(R, NewMenu(
    NewItem('~M~ove', 'Ctrl-F1', kbF2, cmResize, hcResize,
    NewItem('~N~ext', 'Ctrl-F2', kbF1, cmNext, hcNext,
    NewItem('~T~ile', 'Ctrl-F3', kbF3, cmTile, hcTile,
    NewItem('~C~ascade', 'Ctrl-F4', kbF4, cmCascade, hcCascade,
    NewSubMenu('~P~roject Analysis', hcMenuProjectAnalysis, NewMenu(
      NewItem('~P~roject Batch Management', 'Alt-F8', kbAltF8, cmBatchFiles,hcMe
nuBatchMan,
      NewItem('~E~xpert System Management', 'Alt-F9', kbAltF9, cmExSystem1,hcMen
uESysMan,
        nil))),
    NewItem('~R~esult Analysis', '', 0, cmRAnalysis, hcResultAnalysis,
    NewItem('~Q~uit', '', 0, cmShutDown10, hcQuit,
      nil))))),
    nil));
end;

(*****
(          GetEvent          )
*****)
procedure TFormApp.GetEvent(var Event: TEvent);
var
  S : string;
  W: PWindow;
  HFile: PHelpFile;
  HelpStrm: PDosStream;
const
  HelpInUse: Boolean = False;
begin
  inherited GetEvent(Event);
  case Event.What of
    evCommand:
      if (Event.Command = cmHelp) and not HelpInUse then
      begin
        HelpInUse := True;
        HelpStrm := New(PDosStream, Init(CalcHelpName, stOpenRead));
        HFile := New(PHelpFile, Init(HelpStrm));
        if HelpStrm.Status <> stOk then
        begin
          Str(HelpStrm.Status:1,s);
          MessageBox('Could not open help file.'+#13+
            'Status is : '+S , nil, mfError + mfOkButton);
          Dispose(HFile, Done);
        end
        else
        begin
          W := New(PHelpWindow,Init(HFile, GetHelpCtx));
          if ValidView(W) <> nil then
          begin
            ExecView(W);
            Dispose(W, Done);
          end;
          ClearEvent(Event);
        end;
      end;
  end;
end;

```



```

    end;
    HelpInUse := False;
  end;
  evMouseDown:
  { if Event.Buttons <> 1 then Event.What := evNothing;
  }
end;
end;

{*****}
{          GetPalette          }
{*****}
function TFormApp.GetPalette: PPalette;
const
  CNewColor = CAppColor + CHelpColor;
  CNewBlackWhite = CAppBlackWhite + CHelpBlackWhite;
  CNewMonochrome = CAppMonochrome + CHelpMonochrome;
  P: array[apColor..apMonochrome] of string[Length(CNewColor)] =
    (CNewColor, CNewBlackWhite, CNewMonochrome);
begin
  GetPalette := @P[AppPalette];
end;

{*****}
{          Done          }
{*****}
Destructor TFormApp.Done;
begin
  TApplication.Done;
end;

{*****}
{          Color Dialog          }
{*****}

procedure TFormApp.ColorDialog;
var
  E      : TEvent;
  D      : PColDialog;
  WasColor : Byte;
begin
  D := New(PColDialog, Init);
  WasColor := PMBG^.GetColorValue;
  WITH E DO
    BEGIN
      What      := evBroadcast;
      Command   := cmColorSet;
      InfoByte  := WasColor;
      PutEvent(E);
    END;
  IF DeskTop^.ExecView(D) = cmCancel THEN
    PMBG^.NewColor(WasColor);
  Dispose(D, Done);
end;

{*****}
{          Expert Systems 1          }
{*****}

procedure TFormApp.OpenExpertSystems;

```

```

var
  DB1      : PDBaseDialog;
  Control : Word;
begin
  { Execute PDBaseDialog }
  DB1 := New(PDBaseDialog, Init);
  DB1^.HelpCtx := hcExpertManage;
  DeskTop^.Insert(ValidView(DB1));
  Redraw;
end;

{*****}
{          Idle          }
{*****}

procedure TFormApp.Idle;
var
  Event : TEvent;
begin
  TApplication.Idle;

  IF (SaveInterval > 0) AND
    (Ticks >= ScreenSave) AND
    (NOT InGraphics) THEN
    BEGIN
      Write(#7);
      Event.What := evCommand;
      Event.Command := cmCrum;
    END
  ELSE
    BEGIN
      Event.What := evBroadcast;
      Event.Command := cmGrIdle;
    END;
  PutEvent(Event);
end;

{*****}
{          Screen saver          }
{*****}

procedure TFormApp.DoCrumDialog;
VAR
  P      : PDialog;
  Control : Word;
BEGIN
  P := New(PCrumDialog, Init);
  Control := ExecView(P);
  Dispose(P, Done);
END;

procedure TFormApp.ToggleScreenSave;
VAR P : PMenuItem;
BEGIN
  P := MenuBar^.HotKey(kbF5);
  IF P <> NIL THEN

```

```

BEGIN
  DisposeStr(P^.Name);
  IF SaveInterval = 0 THEN
    BEGIN
      P^.Name := NewStr('Turn Screensave OFF');
      SaveInterval := SaveTicks;
      ScreenSave := Ticks + SaveInterval;
    END
  ELSE
    BEGIN
      P^.Name := NewStr('Turn Screensave ON');
      SaveInterval := 0;
    END
  END;
END;

{*****}
{      TFormApp Init      }
{*****}

constructor TFormApp.Init;
var
  Event      : TEvent;
  filename   : ^Pathstr;
  R          : TRect;
  list1,list2 : PListDialog;
  pat       : Char;
  Col       : Byte;
  ControlStr : String;
  Control    : String;
  Name       : String;
  ShowWindow : PDemoWindow2;

begin
  Control := ParamStr(1);
  if (length(Control) = 0) or (Control <> 'xyz') then
    begin
      ClrScr;
      Writeln('Type esea.exe to run the program');
      Halt;
    end;

  TApplication.Init;

  DeskTop^.Background^.GetExtent(R);
  col := DeskTop^.Background^.GetColor($01);
  Pat := DeskTop^.Background^.Pattern;
  DeskTop^.Delete(DeskTop^.Background);
  Dispose(DeskTop^.Background, done);

  DeskTop^.Background := New(PMyBack, Init(R, #176, col));
  DeskTop^.Insert(DeskTop^.Background);
  PMBG := PMyBack(DeskTop^.Background);
  PMBG^.NewFore(9); {9}
  PMBG^.NewBack(3); {3}

  { Register all objects that will be loaded or stored on streams }
  ExpertObjectStreamRegistration;
  RegisterHelpFile;

```

```

SaveInterval := 0;
ScreenSave := Ticks + SaveInterval;

GetExtent(R);
Dec(R.B.X);
R.A.X := R.B.X - 9; R.A.Y := R.B.Y - 1;
Heap := New(PHeapView, Init(R));
Insert(Heap);

ControlStr := ParamStr(2);
if ControlStr = 'MessageOK' then
  begin
    MessageBox(#3'An Expert System Utility for the'+#13+
      #3'Evaluation of Environmental Impacts'+#13+
      #3'of Civil Engineering Projects', nil, mfOkButton);

    { Preamble window }
    Name := 'Preamble.txt';
    R.Assign(1,1,68,17);
    R.Move(5,2);
    ShowWindow := New(PDemoWindow2, Init(R, '', Name));
    DeskTop^.Insert(ShowWindow);
    Delay(2000);
    Message(Application, evBroadcast, cmShutDown30, nil);
  end;

end; { Init }

{*****}
{      TFormApp Handle Event      }
{*****}

procedure TFormApp.HandleEvent(var Event: TEvent);
var
  NewMode      : Word;
  P            : Pointer;
  NewName,DelKeyName : String;
  NewCode      : Integer;
  PNewBatchObject : PBatchObject;
  List         : PCollection;
  CodeReturned  : Integer;
  PDelBatchObject : PBatchObject;
  Option       : Word;

  function CodeMatch(App:PBatchObject):Boolean;far;
  begin
    CodeMatch := CodeReturned = App^.PCode^;
  end;

  procedure ChangeDir;
  var
    D: PChDirDialog;
  begin
    D := New(PChDirDialog, Init(0, hlChangeDir));
    if ValidView(D) <> nil then
      begin
        DeskTop^.ExecView(D);
        Dispose(D, Done);
      end;

```

```

end;

procedure ShowRuleBase;
var
  FileName : FNameStr;
  WildCard : PathStr;
  R        : TRect;
  Window   : PDemoWindow;
begin
  WildCard := '*.rb';
  FileName := '*.rb';
  if ExecuteDialog(New(PFileDialog, Init(WildCard, 'Open a file',
    '~N-ame', fdOpenButton + fdHelpButton, 100)), @FileName) <> cmCancel then
  begin
    Inc(WinCount);
    R.Assign(0, 0, 60, 20);
    R.Move(Random(55), Random(16));
    Window := New(PDemoWindow, Init(R, 'View '+FileName, WinCount, FileName));
    Desktop^.Insert(Window);
  end;
end;

procedure EditRuleBase1;
begin
  DoneSysError;
  DoneEvents;
  DoneVideo;
  DoneMemory;
  SetMemTop(HeapPtr);
  SwapVectors;
  Exec(GetEnv('COMSPEC'), '/C tvedit.exe ');
  SwapVectors;
  SetMemTop(HeapEnd);
  InitMemory;
  InitVideo;
  InitEvents;
  InitSysError;
  Redraw;
end;

procedure EditRuleBase2;
var
  D: PFileDialog;
  FileName: ^PathStr;
  ListEditor: PDialog;
  Window: PDemoWindow;
  R: TRect;
  filestring1, filestring2: string[35];
begin
  D := New(PFileDialog, Init('*.RB', 'Open File',
    '~N-ame', fdOpenButton, hlOpenListDlg));
  if ValidView(D) <> nil then
  begin
    if Desktop^.ExecView(D) <> cmCancel then
    begin
      New(FileName);
      D^.GetFileName(FileName^);
      if not FileExists(FileName^) then

```

```

    MessageBox('Cannot find file (%s).', @FileName, mfError + mfOkButton)
  else
  begin
    filestring2 := filename^;
    filestring1 := '/C synapse edit ' + filestring2;
    DoneSysError;
    DoneEvents;
    DoneVideo;
    DoneMemory;
    SetMemTop(HeapPtr);
    SwapVectors;
    Exec(GetEnv('COMSPEC'), filestring1);
    SwapVectors;
    SetMemTop(HeapEnd);
    InitMemory;
    InitVideo;
    InitEvents;
    InitSysError;
    Redraw;
  end;
  Dispose(FileName);
end;
Dispose(D, Done);
end;

procedure ExInxDBase1;
begin
  MessageBox(#3'This facility has not been implemented', nil, mfInformation + mfOkButton);
end;

procedure AboutBox1;
begin
  MessageBox(#3'This is Version 2 of the Expert System Utility'#13#3'by K G Merce
r', nil, mfInformation + mfOkButton);
end;

{*****}
begin
  if (Event.What=evBroadcast) and (Event.Command=cmShowAll) then
  begin
    ClearEvent(Event);
    Redraw;
  end;

  if (Event.What=evBroadcast) and (Event.Command=cmShutDown20) then
  begin
    ClearEvent(Event);
    Event.What := evCommand;
    Event.Command := cmQuit;
    PutEvent(Event);
  end;

  if (Event.What=evBroadcast) and (Event.Command=cmNewBatchObjBrdcst) then
  begin
    { Read in batch info from ResourceFile }
    RezName := 'Batch.Rez';
    BatchRez.Init(New(PBufStream, Init(RezName, stOpen, 1024)));
    if BatchRez.Stream^.Status <> 0 then

```

```

    MessageBox(#3'Error loading Resourcefile', nil, mfInformation + mf
OkButton);

    BatchList := PCollection(BatchRez.Get('KeyNames'));

    ( Modify BatchObject list then re-insert into resource )
    NewName := PRecD1(Event.InfoPtr).BigName^;
    NewCode := PRecD1(Event.InfoPtr).PCode^;
    PNewBatchObject := New(PBatchObject, Init(NewName, NewName, NewCode));
    BatchList^.Insert(PNewBatchObject);
    BatchRez.Put(BatchList, 'KeyNames');

    ( Create new BatchDataObject collection then insert new collection
    into resource )
    List := New(PCollection, Init(12, 5));
    BatchRez.Put(List, NewName);

    BatchRez.Done;
    ClearEvent(Event);
end;

if (Event.What=evBroadcast) and (Event.Command=cmDelBatchObjBrdcst) then
begin
    ( Read in batch info from Resourcefile )
    RezName := 'Batch.Rez';
    BatchRez.Init(New(PBufStream, Init(RezName, stOpen, 1024)));
    if BatchRez.Stream^.Status <> 0 then
        MessageBox(#3'Error loading Resourcefile', nil, mfInformation + mf
OkButton);

    BatchList := PCollection(BatchRez.Get('KeyNames'));
    CodeReturned := PInteger(Event.InfoPtr)^;

    ( Locate BatchObject with that Code )
    PDelBatchObject := BatchList.FirstThat(@CodeMatch);
    DelKeyName := PDelBatchObject^.BatchCallName^;

    ( Modify BatchObject list then re-insert into resource )
    BatchList^.Free(PDelBatchObject);
    BatchList^.Pack;
    BatchRez.Put(BatchList, 'KeyNames');

    ( Delete BatchDataObject collection from the Resource )
    BatchRez.Delete(DelKeyName);

    BatchRez.Done;
    ClearEvent(Event);
end;

TApplication.HandleEvent(Event);

IF NOT ((Event.What = evNothing) OR
((Event.What = evBroadcast) AND
(Event.Command = cmGrIdle))) THEN
    ScreenSave := Ticks + SaveInterval;

if Event.What = evCommand then
begin
    case Event.Command of
        cmCrum      : DoCrumDialog;

```

```

    cmScrTog      : ToggleScreenSave;
    cmExSystem1   : OpenExpertSystems;
    cmRBaseShow   : ShowRuleBase;
    cmRBaseEdit   : EditRuleBase1;
    cmRBaseSYN    : EditRuleBase2;
    cmInxDBase1   : ExInxDBase1;
    cmAboutBox1   : AboutBox1;
    cmRAnalysis   : Analysis;
    cmBatchFiles  : BatchFiles;
    cmChangeDir   : ChangeDir;
    cmTempExit    : DosShell;
    cmColorChg    : ColorDialog;
    cmTile        : Tile;
    cmCascade     : Cascade;
    cmTrigger     : P := Message(Desktop^.Background, evMouseDown, $FF, NIL);

    cmVideoMode : begin
        NewMode := ScreenMode xor smFont8x8;
        if NewMode and smFont8x8 <> 0 then
            ShadowSize.X := 1
        else ShadowSize.X := 2;
        SetScreenMode(NewMode);
    end;

    cmShutDown10 : begin
        ClearEvent(Event);
        Option := MessageBox(#3'Are you sure you want to exit',
            nil, mfYesNoCancel);
        if Option = cmNo then Exit;
        Event.What := evCommand;
        Event.Command := cmQuit;
        PutEvent(Event);
    end;

end;
ClearEvent(Event);
end;
end;

(*****
(                               Init Menu Bar                               )
(*****

procedure TFormApp.InitMenuBar;
var
    R: TRect;
begin
    GetExtent(R);
    R.B.Y := R.A.Y + 1;
    MenuBar := New(PMenuBar, Init(R, NewMenu(
        NewSubMenu('~#240~', hcSystem, NewMenu(
            NewItem('~V~ideo mode' ,'', kbNoKey, cmVideoMode, hcSVideoMode,
            NewItem('~C~olor ' ,'', kbNoKey, cmColorChg , hcSColor,
            NewItem('~E~xit to DOS ,'', kbNoKey, cmTempExit ,hcSExit,
            NewItem('~T~urn Screensave ON ', 'F5', kbF5, cmScrTog ,hcSScreenSave ,
            NewLine(
            NewItem('~A~bout...' ,'', kbNoKey, cmAboutBox1, hcSAbout,
            nil)))))),
        NewSubMenu('~R~ulebases', hcManage, NewMenu(
            NewItem('~V~iew Rulebases' , 'F1', kbF1 , cmRBaseShow ,hcSVi
ew,

```

```

it, NewItem('~E~dit Rulebases' , 'F2' , kbF2 , cmRBaseEdit , hcSED
    NewItem('~S~YNAPSE Integ. Environment' , 'F3' , kbF3 , cmRBaseSYN , hcSIE
    , NewItem('~C~hange directory...' , 'F4' , kbF4 , cmChangeDir , hcSCD
    , nil))))) ,
    NewSubMenu('~W~indow' , hcMenuWindows, NewMenu(
        NewItem('~M~ove' , 'Ctrl-F1' , kbCtrlF1 , cmResize , hcResize,
        NewItem('~N~ext' , 'Ctrl-F2' , kbCtrlF2 , cmNext , hcNext,
        NewItem('~T~ile' , 'Ctrl-F3' , kbCtrlF3 , cmTile , hcTile,
        NewItem('~C~a~scade' , 'Ctrl-F4' , kbCtrlF4 , cmCascade , hcCascade,
        NewItem('~P~rev' , 'Ctrl-F5' , kbCtrlF5 , cmPrev , hcPrev,
        NewItem('~C~lose' , 'Ctrl-F6' , kbCtrlF6 , cmClose , hcClose,
        nil))))) ,
    NewSubMenu('~D~atabases' , hcMenuDatabases, NewMenu(
        NewItem('~W~etlands in SA - Data Base ' , 'Alt-F1' , kbAltF1 , cmInxDBase1
,31, NewItem('~R~egister of Toxic Waste Sites' , 'Alt-F2' , kbAltF2 , cmInxDBase1
,32, NewItem('~D~atabase of Toxic Substances' , 'Alt-F3' , kbAltF3 , cmInxDBase1
,33, NewItem('~E~nvo. Sensitive Esturine Flora' , 'Alt-F4' , kbAltF4 , cmInxDBase1
,33, NewItem('~E~nvo. Sensitive Esturine Fauna' , 'Alt-F5' , kbAltF5 , cmInxDBase1
,33, NewItem('~D~atabase of Toxic Emmissions' , 'Alt-F6' , kbAltF6 , cmInxDBase1
,33, nil))))) ,
    NewSubMenu('~P~roject Analysis' , hcMenuProjectAnalysis, NewMenu(
        NewItem('~B~atchRun Management' , 'Alt-F8' , kbAltF8 , cmBatchFiles, hcM
enuBatchMan,
        NewItem('~E~xpert System Management' , 'Alt-F9' , kbAltF9 , cmExSystem1, hcMen
uESysMan,
        nil)) ,
    NewItem('~R~esult Analysis' , '' , 0 , cmRAnalysis , hcResultAnalysis,
    NewItem('~Q~uit' , '' , 0 , cmShutDown10 , hcQuit,
    nil))))) ,
end;

```

```

{*****}
{          Init Status Line          }
{*****}

```

```
procedure TFormApp.InitStatusLine;
```

```

var
  R: TRect;
begin
  GetExtent(R);
  R.A.Y := R.B.Y - 1;
  StatusLine := New(PStatusLine, Init(R,
    NewStatusDef(0, $FFFF,

```

```

    NewStatusKey('~Alt-X~ Exit' , kbAltX , cmShutDown10,
    NewStatusKey('~F1~ Help' , kbF1 , cmHelp,
    NewStatusKey('~F10~ Bar-Menu ' , kbF10 , cmMenu,
    NewStatusKey('~Alt-Space~ Pop-Up Menu ' , kbAltSpace , cmTrigger,
    nil)) ,
    nil)) ,
end;

{*****}
{*****}
{*****}

var
  FormApp : TFormApp;

begin
  FormApp.Init;
  FormApp.Run;
  FormApp.Done;
end.

```

```

(*****)
(
(   Unit for viewing rulebases   )
(
(   Filename : ViewWin2.pas      Author K G Mercer )
(   Date    : September 1993    )
(
(   Objects  : TInterior        )
(           : TDemoWindow2      )
(
(*****)
{$O+}
{$F+}
unit viewwin2;

```

```

interface

```

```

uses Objects, Drivers, Views, Menus, App, Dos, CmdnList;

```

```

const
    MaxLines      = 100;
    WinCount: Integer = 0;

```

```

var
    LineCount: Integer;
    Lines: array[0..MaxLines - 1] of PString;

```

```

type

```

```

    PInterior = ^TInterior;
    TInterior = object(TScroller)
        constructor Init(var Bounds: TRect; AHScrollBar,
            AVScrollBar: PScrollBar);
        procedure Draw; virtual;
    end;

    PDemoWindow2 = ^TDemoWindow2;
    TDemoWindow2 = object(TWindow)
        constructor Init(Bounds: TRect; WinTitle: String; FileToRead : pathstr);
        procedure MakeInterior(Bounds: TRect);
        destructor Done; virtual;
        procedure HandleEvent(var Event: TEvent); virtual;
    end;

```

```

implementation

```

```

procedure ReadFile( FileToRead : pathstr);

```

```

var
    F: Text;
    S: String;
begin
    LineCount := 0;
    Assign(F, FileToRead);
    ($I-)
    Reset(F);
    ($I+)
    if IOResult <> 0 then
    begin
        Writeln('Cannot open ', FileToRead);
        Halt(1);
    end;

```

```

end;
while not Eof(F) and (LineCount < MaxLines) do
begin
    Readln(F, S);
    Lines[LineCount] := NewStr(S);
    Inc(LineCount);
end;
Close(F);
end;

```

```

procedure DoneFile;

```

```

var
    I: Integer;
begin
    for I := 0 to LineCount - 1 do
        if Lines[I] <> nil then DisposeStr(Lines[I]);
    end;

```

```

(*****)
(                               TInterior                               )
(*****)

```

```

constructor TInterior.Init(var Bounds: TRect; AHScrollBar,
    AVScrollBar: PScrollBar);
begin
    TScroller.Init(Bounds, AHScrollBar, AVScrollBar);
    GrowMode := gfGrowHiX + gfGrowHiY;
    Options := Options or ofFramed;
    SetLimit(128, LineCount);
end;

```

```

procedure TInterior.Draw;

```

```

var
    Color: Byte;
    I, Y: Integer;
    B: TDrawBuffer;
begin
    Color := GetColor(1);
    for Y := 0 to Size.Y - 1 do
    begin
        MoveChar(B, ' ', Color, Size.X);
        I := Delta.Y + Y;
        if (I < LineCount) and (Lines[I] <> nil) then
            MoveStr(B, Copy(Lines[I]^, Delta.X + 1, Size.X), Color);
        Writeln(0, Y, Size.X, 1, B);
    end;
end;

```

```

(*****)
(                               TDemoWindow2                               )
(*****)

```

```

procedure TDemoWindow2.HandleEvent(var Event: TEvent);

```

```

begin
    if (Event.What=evBroadcast) and (Event.Command=cmShutDown30) then
    begin
        ClearEvent(Event);
        Free;
        Exit;
    end;

```

```

        end;

    ( Respond to cmCancel )
    if Event.What = evCommand then
    begin
        case Event.Command of
            cmCancel, cmClose : if State and sfModal <> 0 then
                begin
                    EndModal(Event.Command);
                    ClearEvent(Event);
                end;
        end ( Case )
    end;

    with Event do
        if ((What = evKeyDown) and (KeyCode = kbEsc)) then
        begin
            ClearEvent(Event);
            Free;
            Exit;
        end;

        TWindow.HandleEvent(Event);

        if Event.What=evCommand then
        begin
            ClearEvent(Event);
        end;
    end;

constructor TDemoWindow2.Init(Bounds: TRect; WinTitle: String; FileToRead : paths
tr);
var
    S: string[3];
begin
    ReadFile( FileToRead );
    TWindow.Init(Bounds, '', wnNoNumber);
    MakeInterior(Bounds);
    Options:=Options + ofTileable;
end;

procedure TDemoWindow2.MakeInterior(Bounds: TRect);
var
    HScrollBar, VScrollBar: PScrollBar;
    Interior: PInterior;
    R: TRect;
begin
    VScrollBar := StandardScrollBar(sbVertical + sbHandleKeyboard);
    HScrollBar := StandardScrollBar(sbHorizontal + sbHandleKeyboard);
    GetExtent(Bounds);
    Bounds.Grow(-1, -1);
    Interior := New(PInterior, Init(Bounds, HScrollBar, VScrollBar));
    Insert(Interior);
end;

destructor TDemoWindow2.Done;
begin
    DoneFile;
    TWindow.Done;
end;

```

```

end.

```

```

{*****}
{
  Unit for viewing rulebases 2
}
{
  Filename : ViewWinF.pas      Author K G Mercer
  Date    : September 1993
}
{
  Objects : TInterior
           : TDemoWindow
}
{*****}
{$O+}
{$F+}
unit viewwinf;

interface

uses Objects, Drivers, Views, Menus, App, Dos, CmdnList;

const
  MaxLines      =1100;
  WinCount: Integer = 0;

var
  LineCount: Integer;
  Lines: array[0..MaxLines - 1] of PString;

type

  PInterior = ^TInterior;
  TInterior = object(TScroller)
    constructor Init(var Bounds: TRect; AHScrollBar: PScrollBar,
                    AVScrollBar: PScrollBar);
    procedure Draw; virtual;
  end;

  PDemoWindow = ^TDemoWindow;
  TDemoWindow = object(TWindow)
    constructor Init(Bounds: TRect; WinTitle: String; WindowNo: Word; FileToRead
  : pathstr);
    procedure MakeInterior(Bounds: TRect);
    destructor Done; virtual;
    procedure HandleEvent(var Event: TEvent); virtual;
  end;

implementation

procedure ReadFile( FileToRead : pathstr);
var
  F: Text;
  S: String;
begin
  LineCount := 0;
  Assign(F, FileToRead);
  {$I-}
  Reset(F);
  {$I+}
  if IOResult <> 0 then
  begin
    Writeln('Cannot open ', FileToRead);
  end;
end;

```

```

  Halt(1);
end;
while not Eof(F) and (LineCount < MaxLines) do
begin
  Readln(F, S);
  Lines[LineCount] := NewStr(S);
  Inc(LineCount);
end;
Close(F);
end;

procedure DoneFile;
var
  i: Integer;
begin
  for i := 0 to LineCount - 1 do
  begin
    writeln('disposing line : ', i);
    if Lines[i] <> nil then DisposeStr(Lines[i]);
  end;
end;

{*****}
{
  TInterior
}
{*****}

constructor TInterior.Init(var Bounds: TRect; AHScrollBar,
  AVScrollBar: PScrollBar);
begin
  TScroller.Init(Bounds, AHScrollBar, AVScrollBar);
  GrowMode := gfGrowHiX + gfGrowHiY;
  Options := Options or ofFramed;
  SetLimit(128, LineCount);
end;

procedure TInterior.Draw;
var
  Color: Byte;
  I, Y: Integer;
  B: TDrawBuffer;
begin
  Color := GetColor(1);
  for Y := 0 to Size.Y - 1 do
  begin
    MoveChar(B, ' ', Color, Size.X);
    I := Delta.Y + Y;
    if (I < LineCount) and (Lines[I] <> nil) then
      MoveStr(B, Copy(Lines[I]^, Delta.X + 1, Size.X), Color);
    WriteLine(0, Y, Size.X, 1, B);
  end;
end;

{*****}
{
  TDemoWindow
}
{*****}

procedure TDemoWindow.HandleEvent(var Event: TEvent);
begin
  ( Respond to CANCEL button and ESC )
end;

```



```
with Event do
  if ((What = evKeyDown) and (KeyCode = kbEsc)) then
    begin
      ClearEvent(Event);
      Free;
      Exit;
    end;

  TWindow.HandleEvent(Event);
end;

constructor TDemoWindow.Init(Bounds: TRect; WinTitle: String;
  WindowNo: Word; FileToRead : pathstr);
var
  S: string[3];
begin
  ReadFile( FileToRead );
  Str(WindowNo, S);
  TWindow.Init(Bounds, WinTitle + ' ' + S, wnNoNumber);
  MakeInterior(Bounds);
  Options:=Options + ofTileable;
end;

procedure TDemoWindow.MakeInterior(Bounds: TRect);
var
  HScrollBar, VScrollBar: PScrollBar;
  Interior: PInterior;
  R: TRect;
begin
  VScrollBar := StandardScrollBar(sbVertical + sbHandleKeyboard);
  HScrollBar := StandardScrollBar(sbHorizontal + sbHandleKeyboard);
  GetExtent(Bounds);
  Bounds.Grow(-1,-1);
  Interior := New(PInterior, Init(Bounds, HScrollBar, VScrollBar));
  Insert(Interior);
end;

destructor TDemoWindow.Done;
begin
  { DoneFile;
  } TWindow.Done;
end;

end.
```

APPENDIX 5

IMPLEMENTATION OF THE EXPERT SYSTEM UTILITY

*"Order and simplification are the first steps
toward the mastery of a subject - the
actual enemy is the unknown"*

Thomas Mann (German writer)

Contents

- 5.1 Introduction
- 5.2 Installation
- 5.3 Program Introduction
- 5.4 System Menu
- 5.5 Rulebase Management
- 5.6 Windows Menu
- 5.7 Databases Menu
- 5.8 Project Analysis Menu
- 5.9 Result Analysis Menu
- 5.10 Floating Menu
- 5.11 Help Facilities
- 5.12 Exiting the System

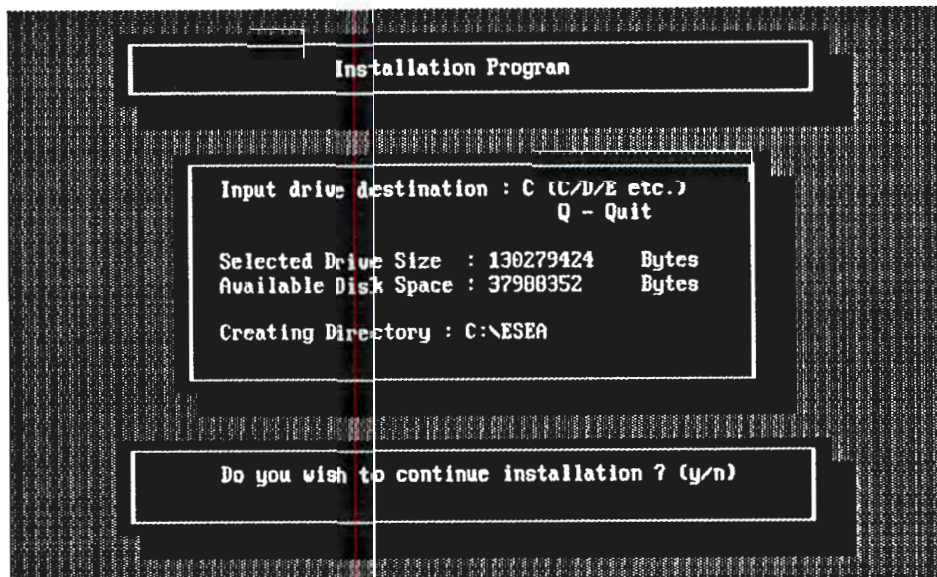
IMPLEMENTATION OF THE EXPERT SYSTEM UTILITY

5.1 Introduction

This chapter reviews how the program utility is implemented using a series of screen captures showing various aspects and capabilities of the system.

5.2 Installation

The program has its own installation program called Install.exe (screen 1). This program prompts the user for the destination drive on which to install the system and then checks to see if there is sufficient memory available. The system is then installed in a subdirectory of that drive called ":\ESEA" which is an acronym for Expert System for Environmental Analysis. The system is installed from files that are "zipped". This means that they have been compressed together so as to require the minimum amount of floppy or stiffy installation discs. As the installation program installs the system these zipped files are then "unzipped" or decompressed and loaded into the correct directory (screen 2).



Screen 1 - Installation Program

Copyright 1989-1993 PKWARE Inc. All Rights Reserved. Shareware Version
PKUNZIP Reg. U.S. Pat. and Tm. Off.

- 80386 CPU detected.
- EMS version 4.00 detected.
- XMS version 3.00 detected.

Searching ZIP: ESYSTEM.ZIP

Inflating: C:/ESEA/BINED.KEY
Inflating: C:/ESEA/BRAND.EXE
Inflating: C:/ESEA/CHKLIST.MS
Inflating: C:/ESEA/CTXCOMP.EXE
Inflating: C:/ESEA/EXTSYS16.FNT
Inflating: C:/ESEA/EXTSYS24.FNT
Inflating: C:/ESEA/EXTSYS32.FNT
Inflating: C:/ESEA/EXTSYS72.FNT
Inflating: C:/ESEA/MARKERS.FNT
Inflating: C:/ESEA/METALOAD.EXE
Inflating: C:/ESEA/METAWND0.EXE
Inflating: C:/ESEA/PCMGRIIDS.EXE
Inflating: C:/ESEA/PCMSHOW.EXE
Inflating: C:/ESEA/PCX2PCM.EXE
Inflating: C:/ESEA/READMES.BAT
Inflating: C:/ESEA/ROMANSIM.FNT
Inflating: C:/ESEA/ROMANTRI.FNT

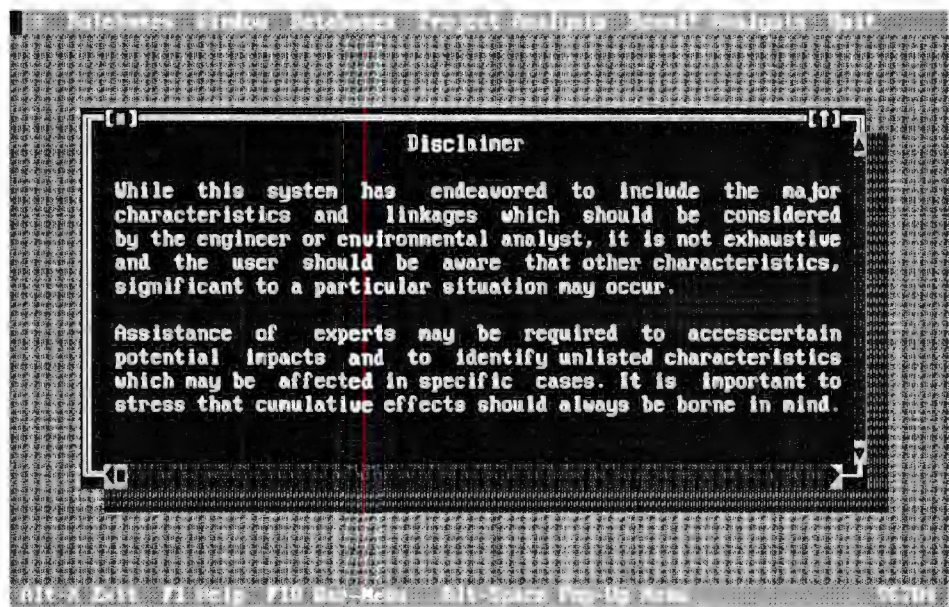
Screen 2 - Unpacking of System Files

5.3 Program Introduction

On typing in the command esea (Enter) the program operation starts. The program "wakes up" with an information window stating the name of the program (screen 3). A mouse click on the OK button clears the screen. A second introductory screen then pops up (screen 4). This shows a disclaimer of which users should be aware before using the system.



Screen 3 - Information Window Showing Program Name

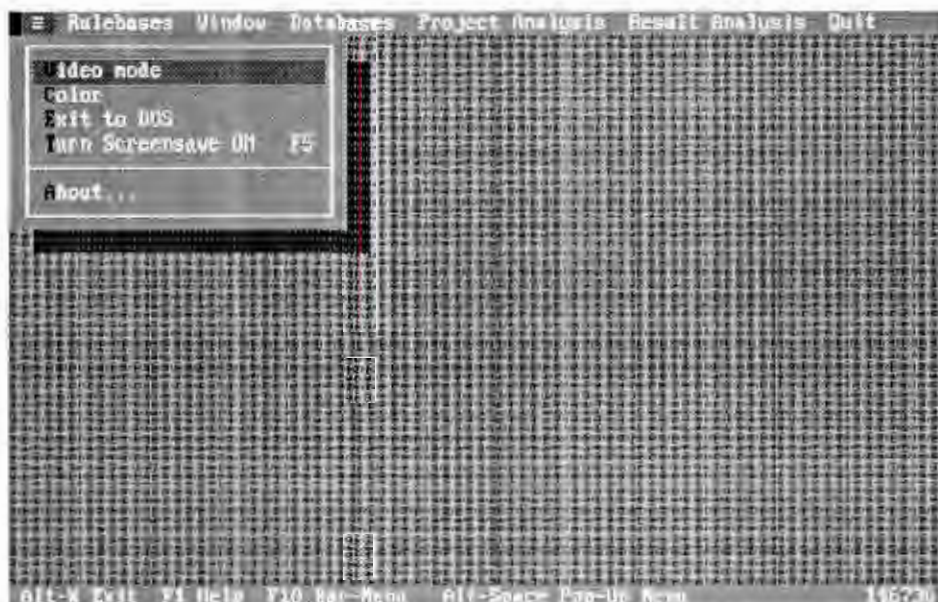


Screen 4 - Disclaimer Window

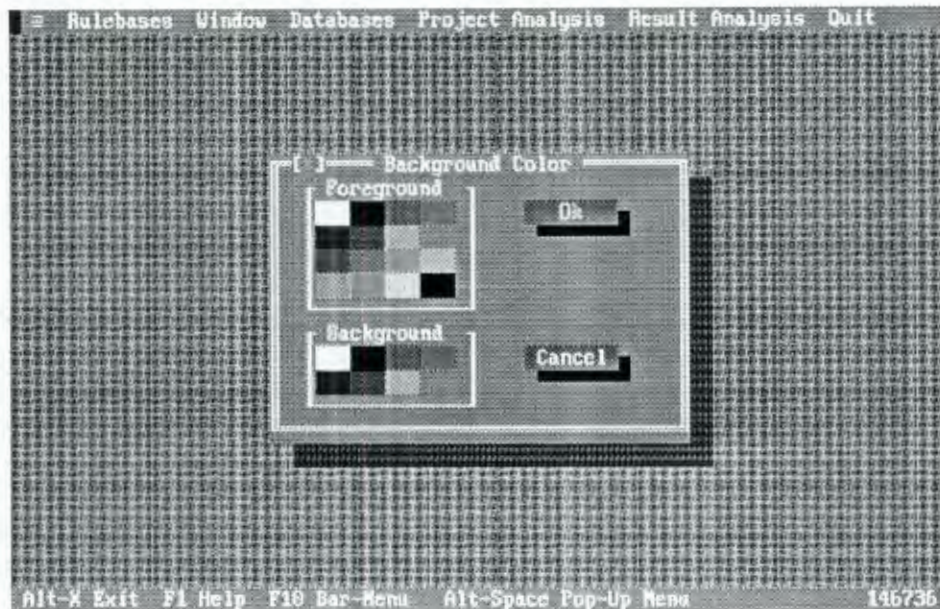
5.4 System Menu

The system menu is selected by clicking on the small triple-striped icon on the left hand side of the bar menu (screen 5). The five options presented are that of changing video mode from 80 x 25 to 80 x 50, changing the default colours, temporarily exiting to DOS, toggling the screen saver on or off and viewing general information about the system.

On selecting the video mode option the only difference in appearance is that there are more lines available on the screen and consequently the text appears smaller. By selecting the colour option the user can change the background and foreground colours according to his/her preference (screen 6). Should the user wish to temporarily exit to DOS without stopping the program operation, the option exit to DOS should be selected (screen 7). A screen saver is a feature available with all modern programs. If this option is selected and the user does not use the program for a limited period of two minutes the screen saver is automatically switched on (screen 8). The screen gradually begins to darken which prevents damage to the monitor. By hitting any key the screen saver is switched off and normal operation can continue.



Screen 5 - System Menu



Screen 6 - System Colour Selection



Screen 7 - Temporary Exit to DOS

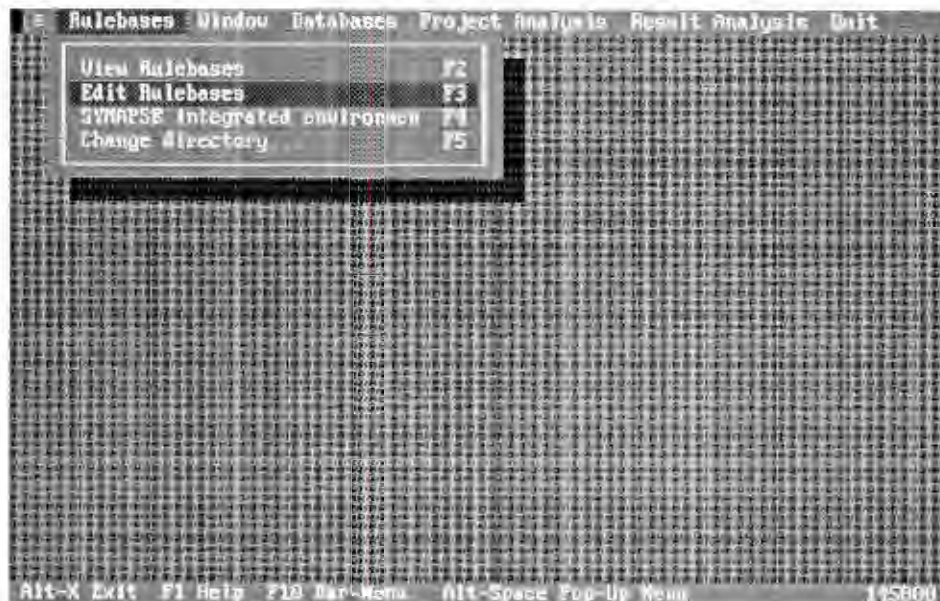


Screen 8 - Screen Saver

5.5 Rulebase Management

Options for managing rulebases are presented by selecting the rulebase pull-down menu on the menu bar (screen 9). The four options presented are for viewing rulebases, editing rulebases using the editor provided, SYNAPSE integrated environment or changing directory to select rulebases in other subdirectories.

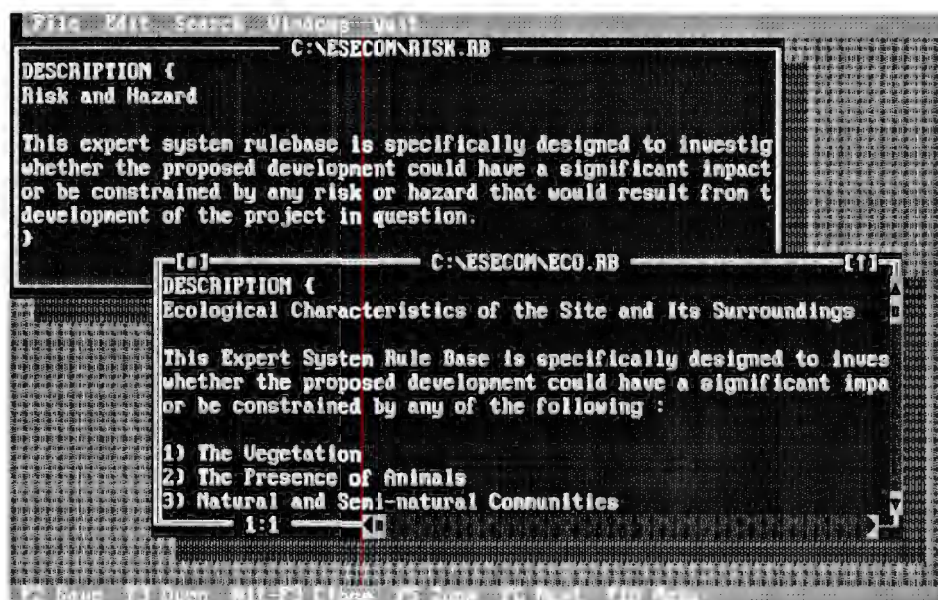
On selecting either the view or edit rulebase options the user is presented with a list of rulebases from which to select (screen 10). The user can view or edit any of one or more rulebases which has been chosen (screen 11). Should the user select the SYNAPSE integrated environment option the program utility then runs this environment (screen 12). Using facilities provided the user can edit, compile and even run rulebases and has other features such as macros and graphical links available. Finally the change directory option provides the user with a facility for changing the current directory (screen 13).



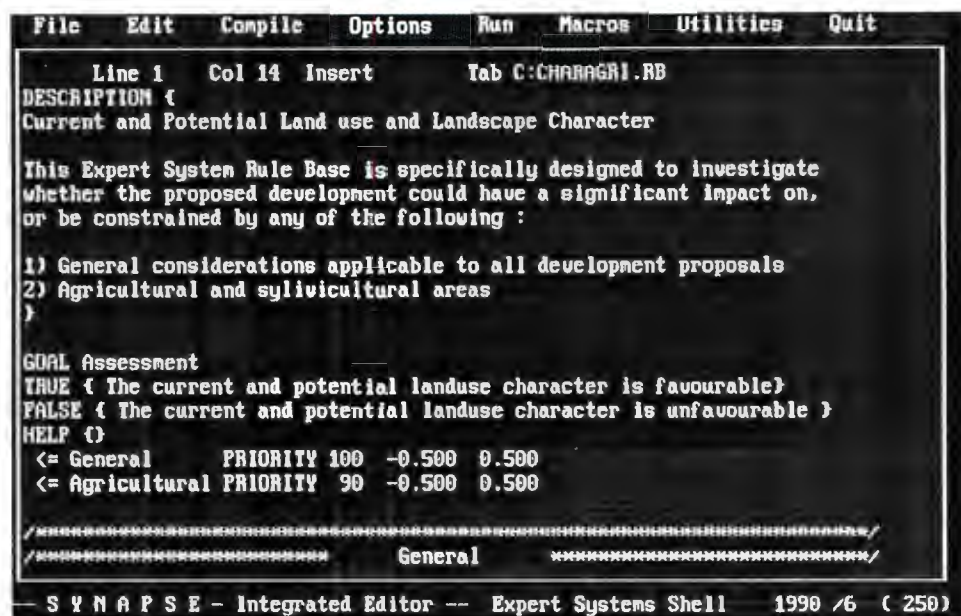
Screen 9 - Rulebase Management Menu



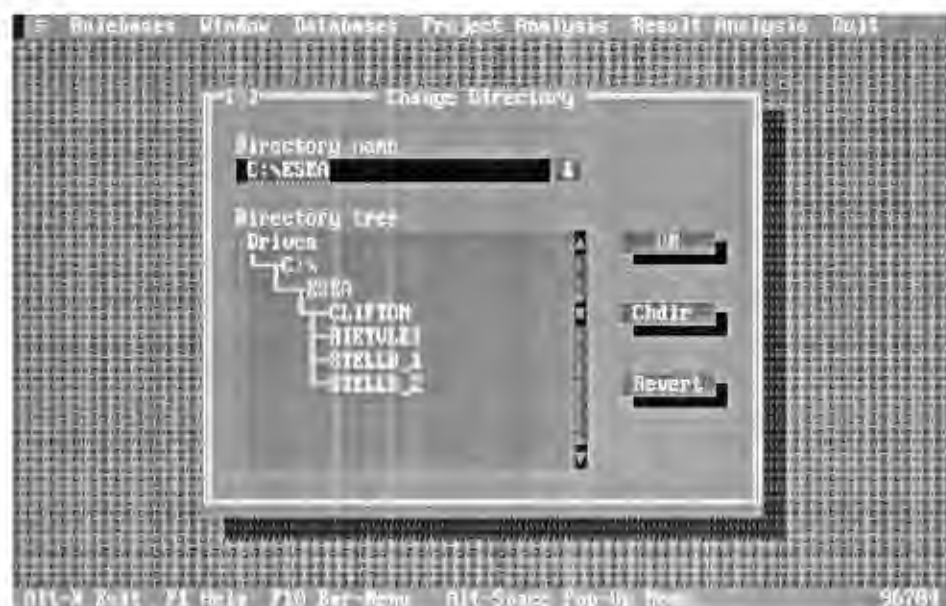
Screen 10 - List of Rulebases to View/Edit



Screen 11 - Viewing/Editing of Multiple Rulebases



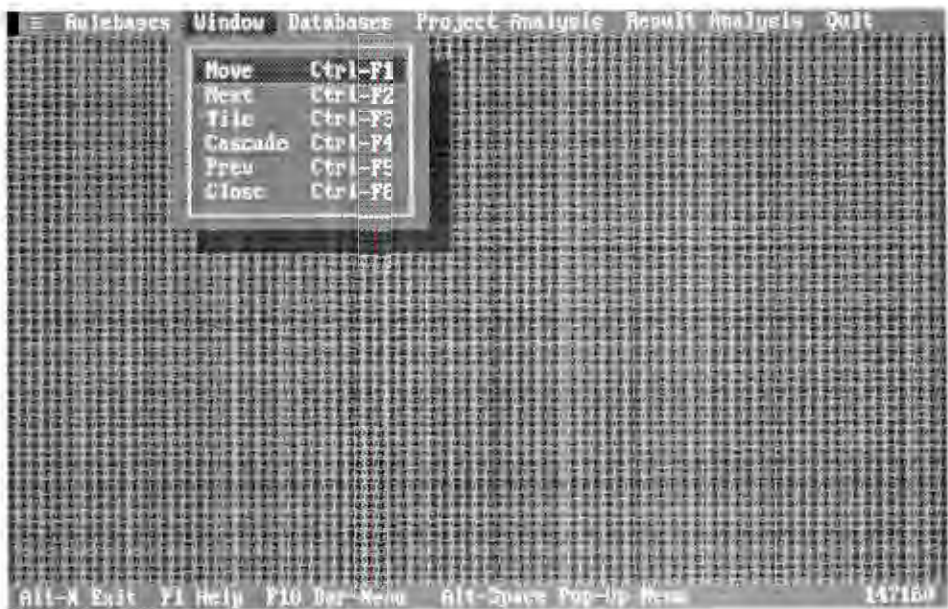
Screen 12 - SYNAPSE Integrated Environment



Screen 13 - Changing Directory

5.6 Windows Menu

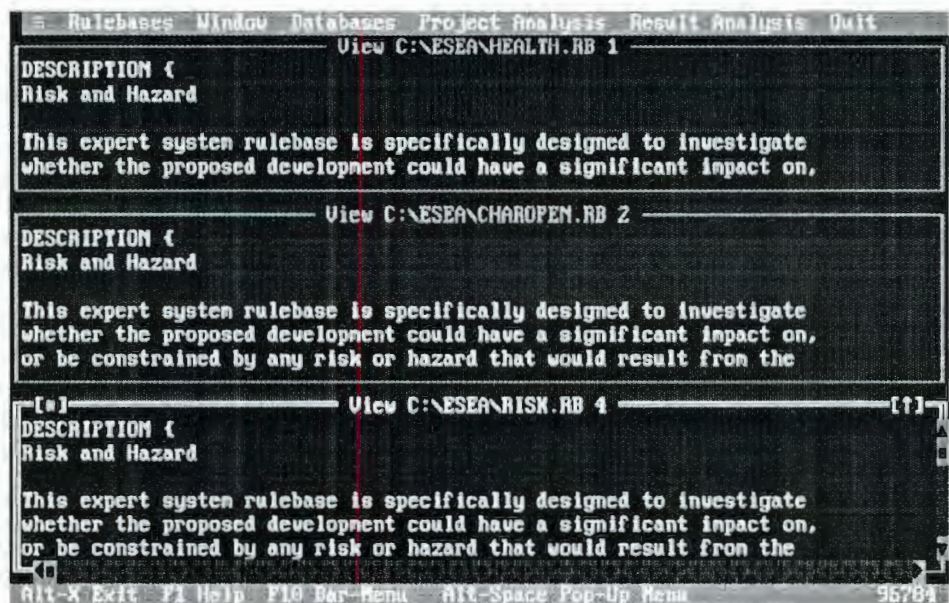
The windows pull-down menu is displayed by selecting the windows option on the menu bar (screen 14). The options presented for manipulating windows are move, next, tile, cascade, prev and close. These are commands that all relate to windows that have been opened usually from the view or edit option in the rulebase menu. The move command allows the user to physically move the window around on the screen using the arrow keys. The 'next' command changes the focus from one window to the next (ie puts the next window on 'top'). The tile and cascade commands control the way in which windows are presented on the screen. From a 'jumble' of windows (screen 15) the user can tile them (screen 16) or cascade them (screen 17). The 'prev' command changes the focus back to the previously selected window (reverse of 'next' command) and the 'close' command closes the window that has the focus.



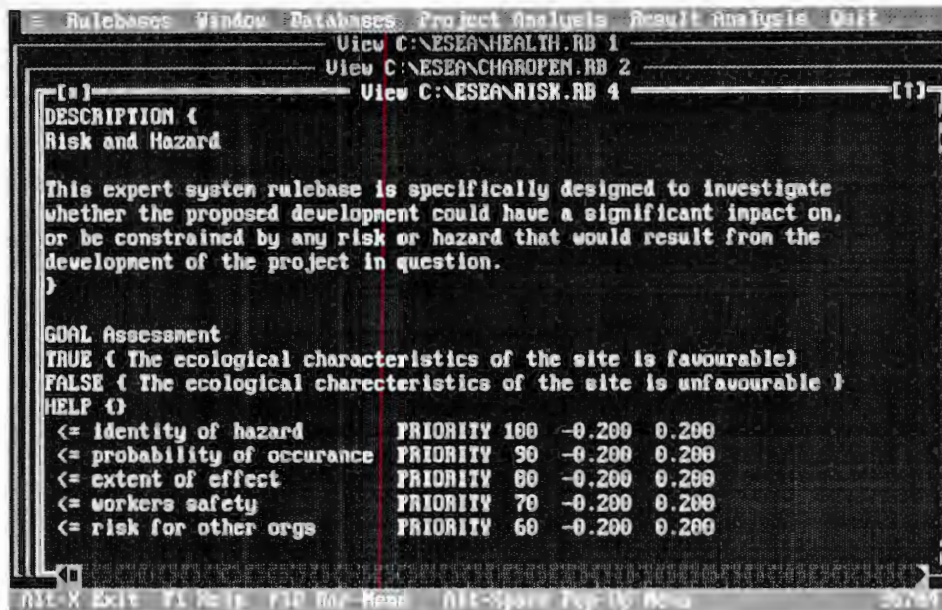
Screen 14 - Windows Menu



Screen 15 - Disorderly Arrangement of Rulebase Windows



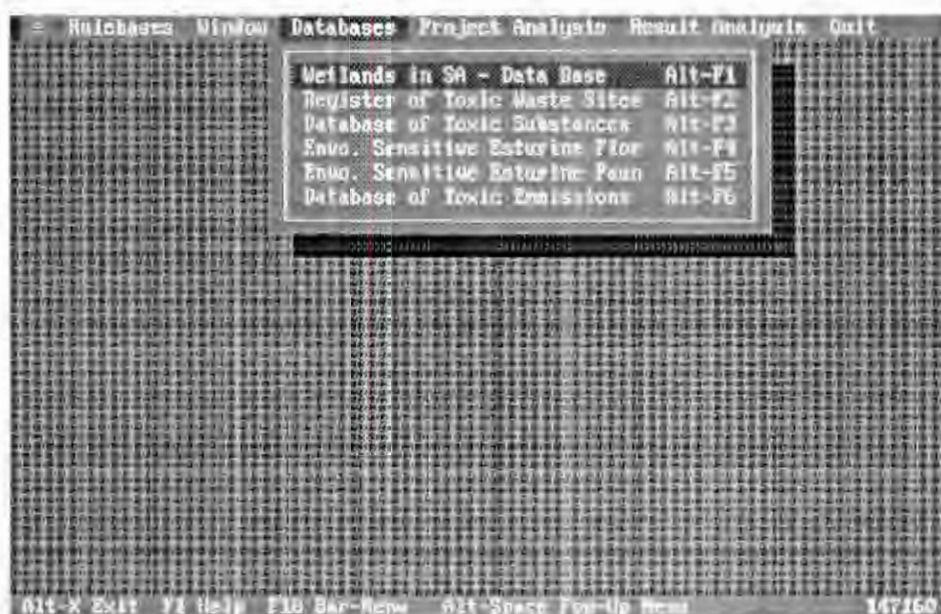
Screen 16 - Tiled Arrangement of Rulebase Windows



Screen 17 - Cascaded Arrangement of Rulebase Windows

5.7 Databases Menu

The database pull-down menu has been included merely for completeness (screen 18). It contains examples of databases that may be further developed in the future and linked to rulebases as the system grows and expands to other related fields. See section 5.9.



Screen 18 - Database Menu

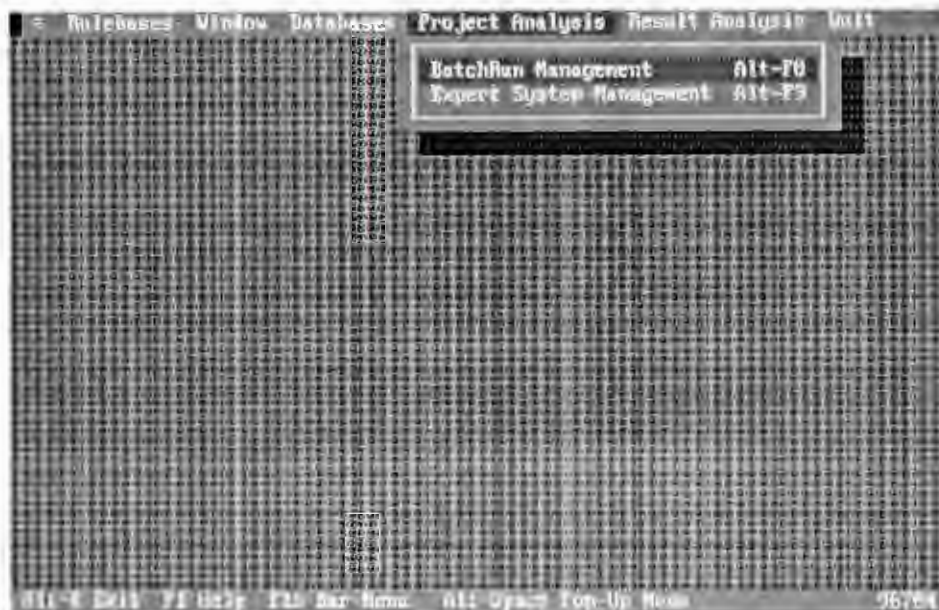
5.8 Project Analysis Menu

The project analysis pull-down menu offers two options, that of batchrun management and expert system management (screen 19). On selecting the batchrun management option a list of batchruns is presented (screen 20) whereby the user can select either a previously defined batchrun or create a new batchrun. Should the user opt to create a new batchrun, a new name is required to be inputted (screen 21) and the batchrun is assembled with selected environmental characteristics and their corresponding weightings (screen 22).

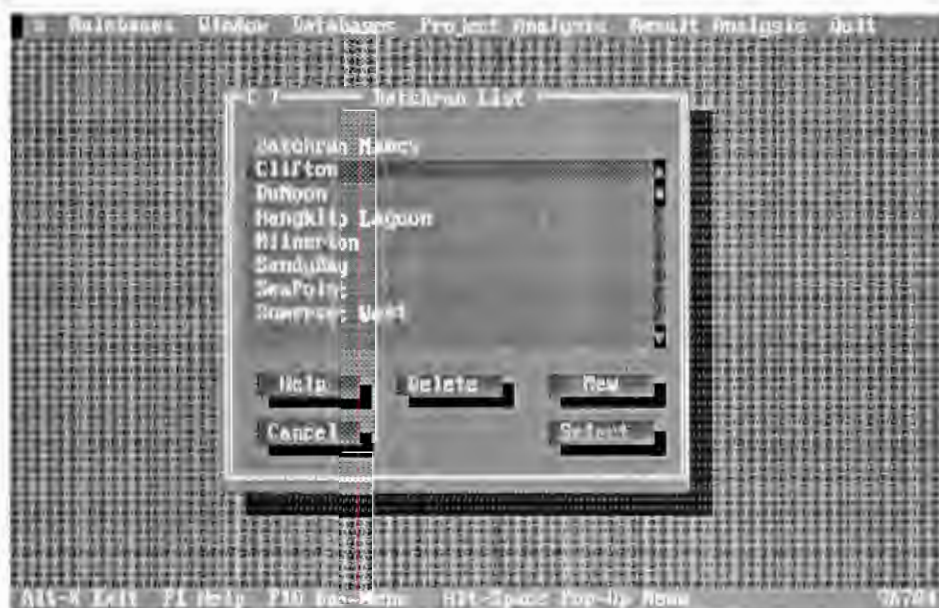
On selecting the expert system management option the user is presented with the expert system control centre window. From here the user can list, add, edit or delete registered rulebases. By selecting 'list' the user can view all the registered rulebases in a scrolling window (screen 24). When editing or deleting registered rulebases the user selects the appropriate rulebase from the list provided (screen 25). When editing or adding a registered rulebase an input window prompts for information that is required for the updating of registration information (screen 26).

The control centre provides capabilities for expert systems to be run on their own or as a batchrun. Should the former option be chosen the user is presented with a selection of rulebases to run (screen 27). If the latter option is chosen the user can select a batchrun to run whereby all the rulebases selected in the batchrun setup are run consecutively (screen 28).

The next three screens show various stages during the execution of a typical SYNAPSE consultation. Screen 29 shows how a typical question is asked, screen 30 shows how a resolved goal is presented and screen 31 shows a typical reasoning process of how SYNAPSE solved the problem.



Screen 19 - Project Analysis Menu



Screen 20 - Batchrun Selection



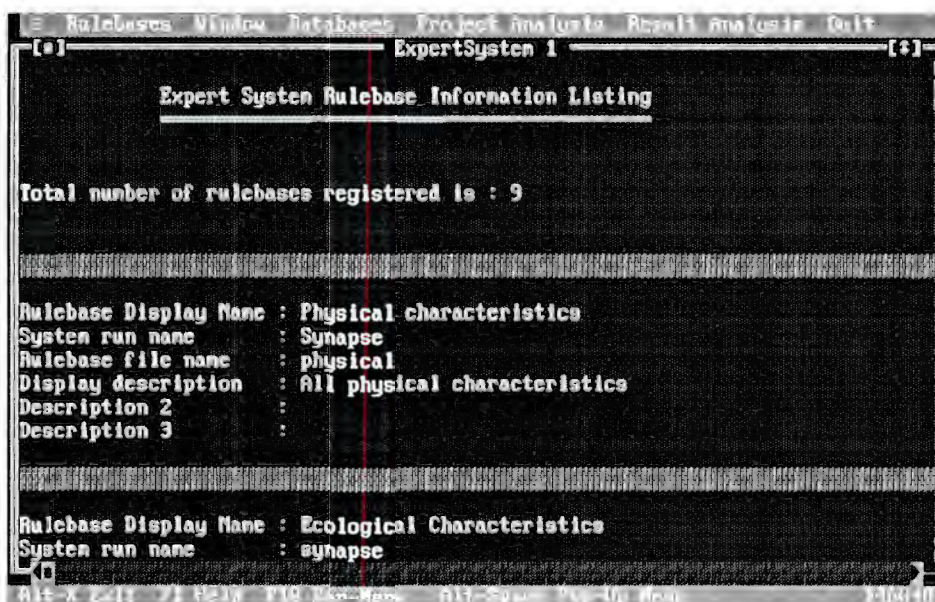
Screen 21 - Inputting New Batchrun Name



Screen 22 - Selected Components and Weightings of a Batchrun



Screen 23 - Expert System Control Centre



Screen 24 - Listing of Registered Rulebases



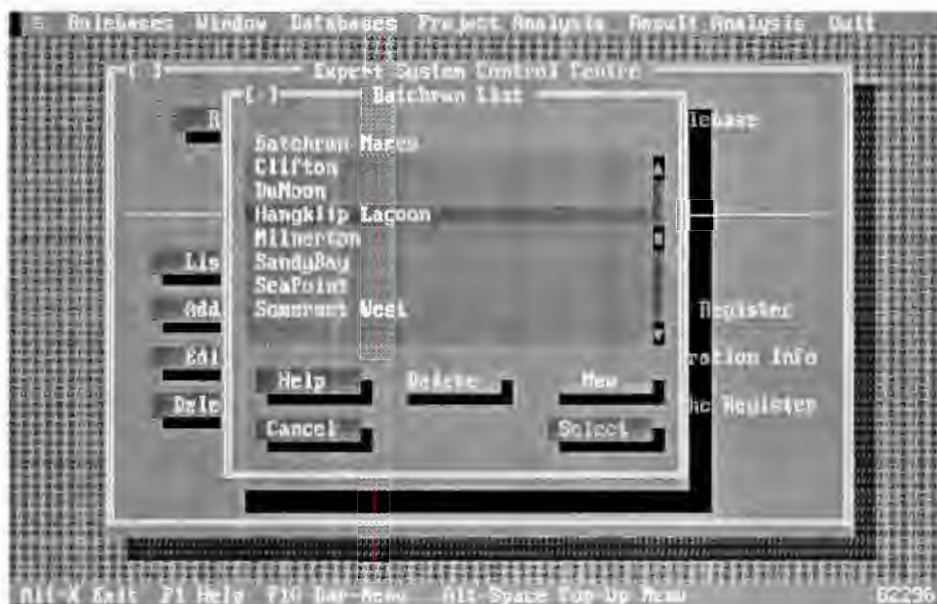
Screen 25 - Selecting an Expert System Rulebase to Modify Registration Information



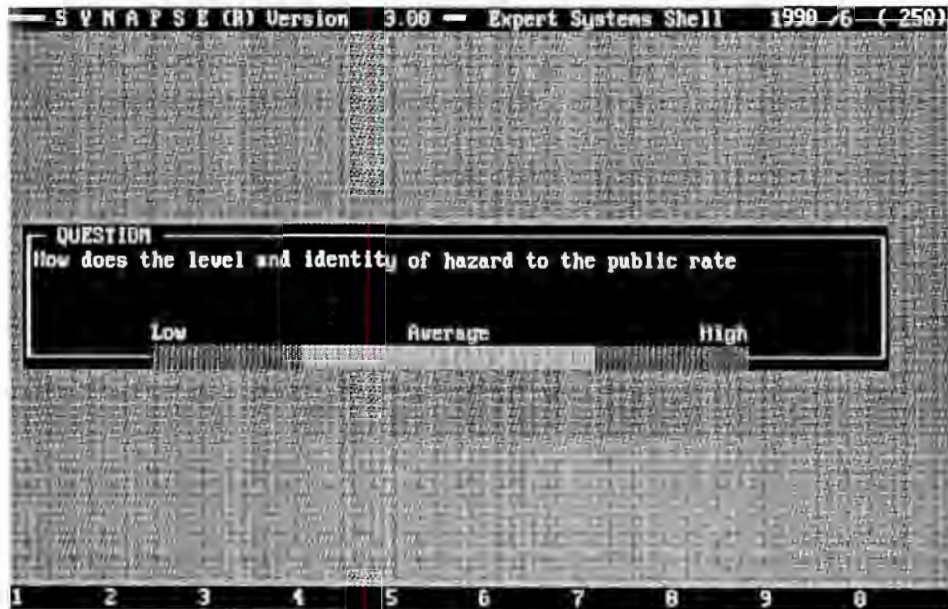
Screen 26 - Adding/Editing of Rulebase Registration Information



Screen 27 - Selecting Expert System Rulebase to Execute



Screen 28 - Selecting Batchrun to Execute



Screen 29 - Typical SYNAPSE Question



Screen 30 - Typical SYNAPSE Resolved Goal

```
SYNAPSE (R) Version 3.00 — Expert Systems Shell 1990 /6 ( 250)

The physical characteristics of the site is favourable

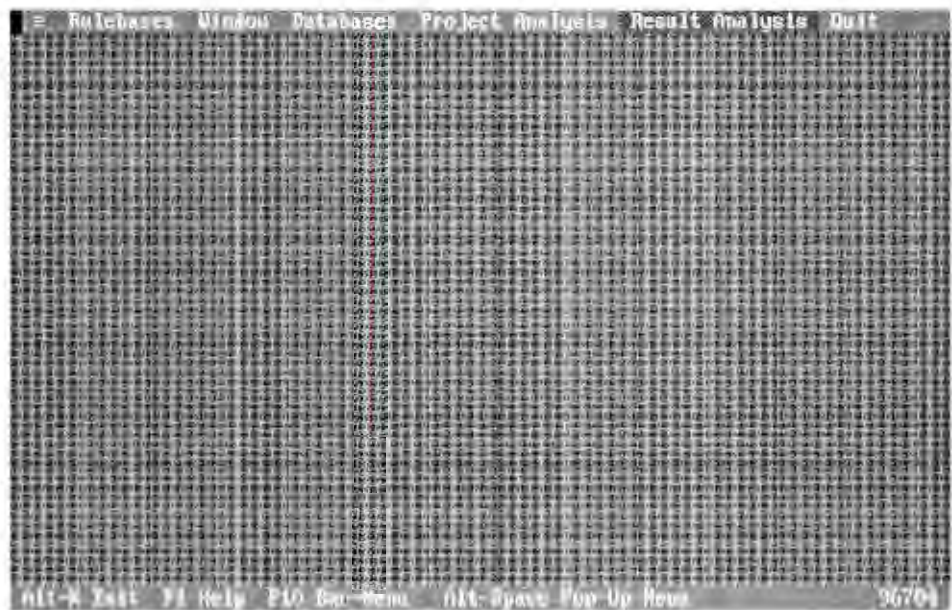
BECAUSE The absence of marine systems makes the site suitable for
development
BECAUSE
The site will adversely affect a river mouth
There are no costal features in the area
Costal islands will be affected by the project
There are existing or altered processes affecting the project
The rocky or sandy shorelines will affect the project
There are inherently unstable ecosystems such as mobile sanddunes
There are sand sources such as mobile sand dunes
ALTHOUGH
The site will not adversely affect the seabed
The site will not affect the functioning of estuary systems

1 2 3 4 5 6 7 8 9 0
```

Screen 31 - Typical SYNAPSE Explanation Facility

5.9 Result Analysis Menu

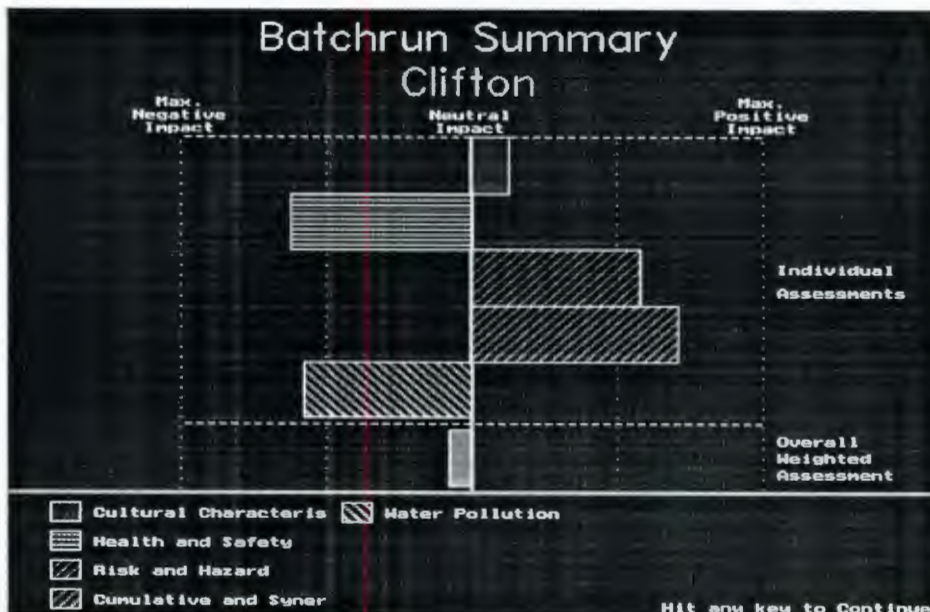
The result analysis menu does not have any options available (screen 32). The user is provided with a list of batchrun names of which to review the results (screen 33). When one of these is selected and provided the analysis has been completed correctly a graphical presentation of the results is shown (screen 34). When finished with this screen a following screen is presented showing the results in RVL terms (screen 35).



Screen 32 - Result Analysis Menu



Screen 33 - Section of Batchrun for Results Review



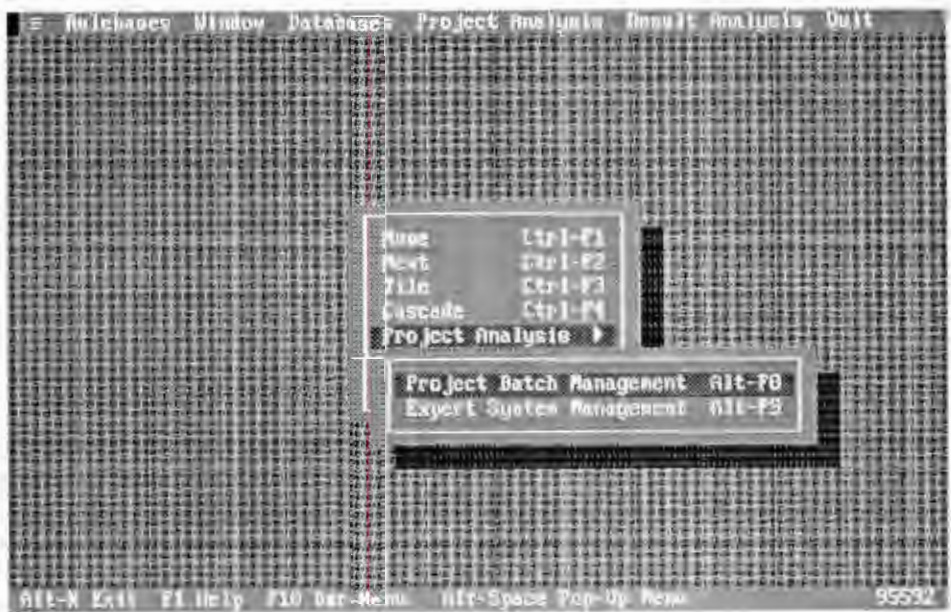
Screen 34 - Graphical Presentation of Results

Batchrun Results 1			
Batchrun RVL Result Information			
Individual Assessment Results		RVL Values	
Rulebase Name		Lower	Expected Upper
Cultural Characteristics		-0.200	+0.130 +0.200
Health and Safety		-0.120	-0.620 -0.120
Risk and Hazard		+0.080	+0.500 +0.480
Cumulative and Synergistic Effects		+0.213	+0.713 +0.213
Water Pollution		-0.308	-0.577 +0.462
Overall Weighted Assessment : RVL = -0.077			

Screen 35 - Results in Terms of RVL Values

5.10 Floating menu

The floating menu is a further feature of the program. It was implemented so that users with a mouse do not have to constantly click on the pull-down menu. By clicking anywhere on the background the floating menu pops up and the user can select most of the major program features from it (screen 36).



Screen 36 - Floating Menu

5.11 Help Facilities

On all of the windows in the program utility there is a help button. By pressing the help button the prepared help text for that window or menu is displayed which provides the user with any further information he/she may require. The help feature also provides cross-reference links to related topics as shown in screen 37.

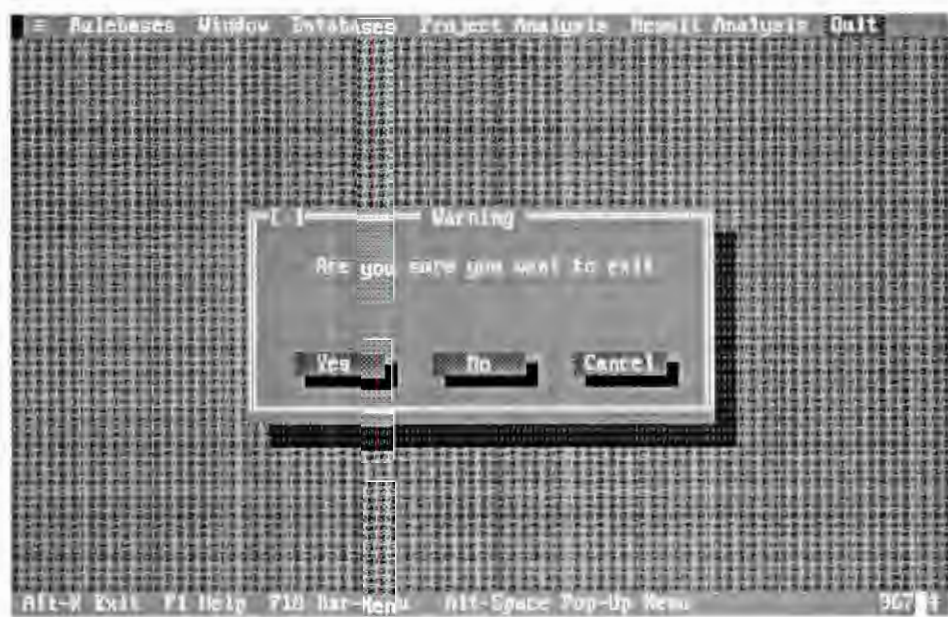


Screen 37 - Typical Help Windows



5.12 Exiting the System

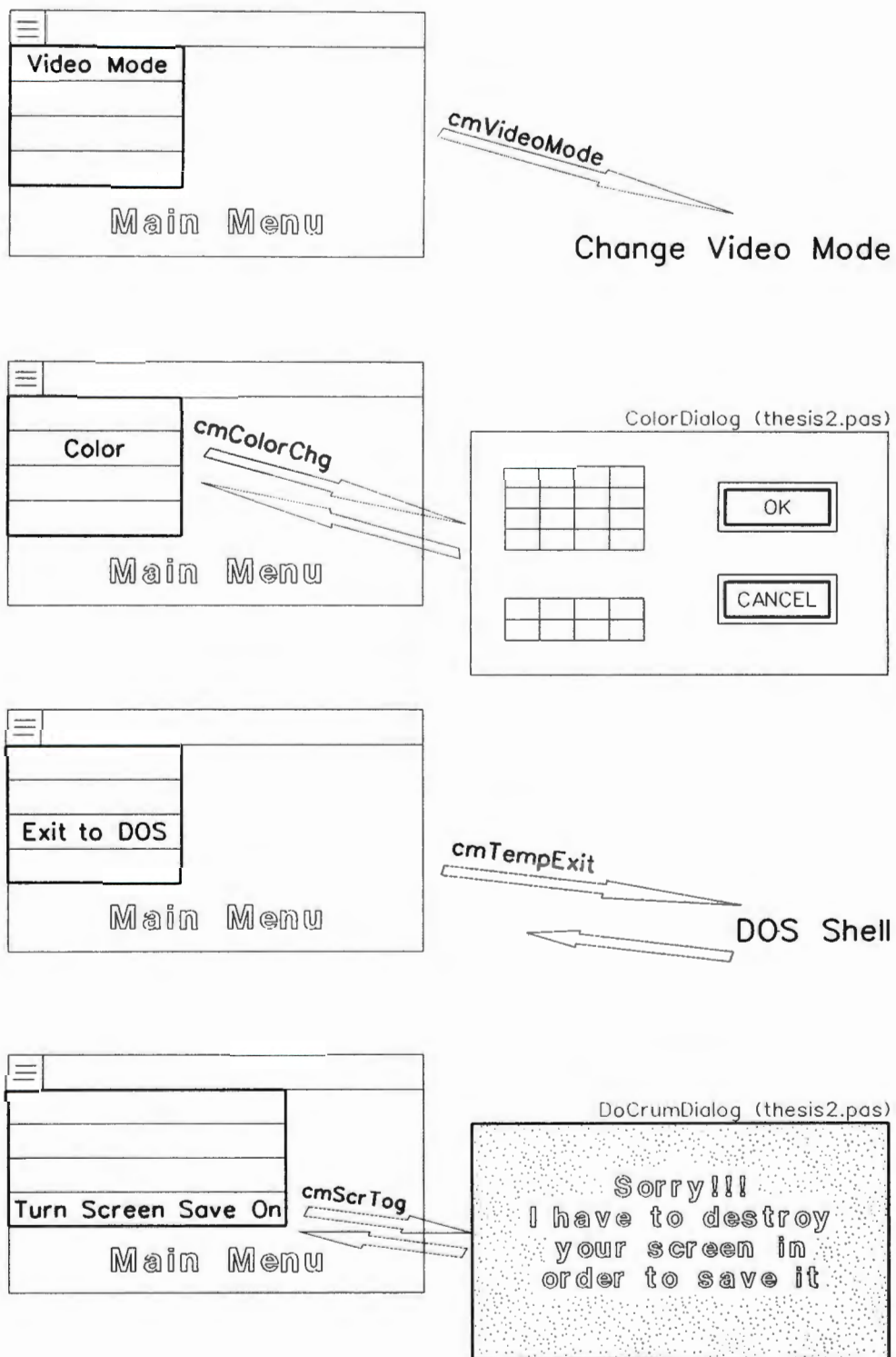
On exiting the system the user is prompted for confirmation of his intention to exit (screen 38).



Screen 38 - Exit Confirmation Window

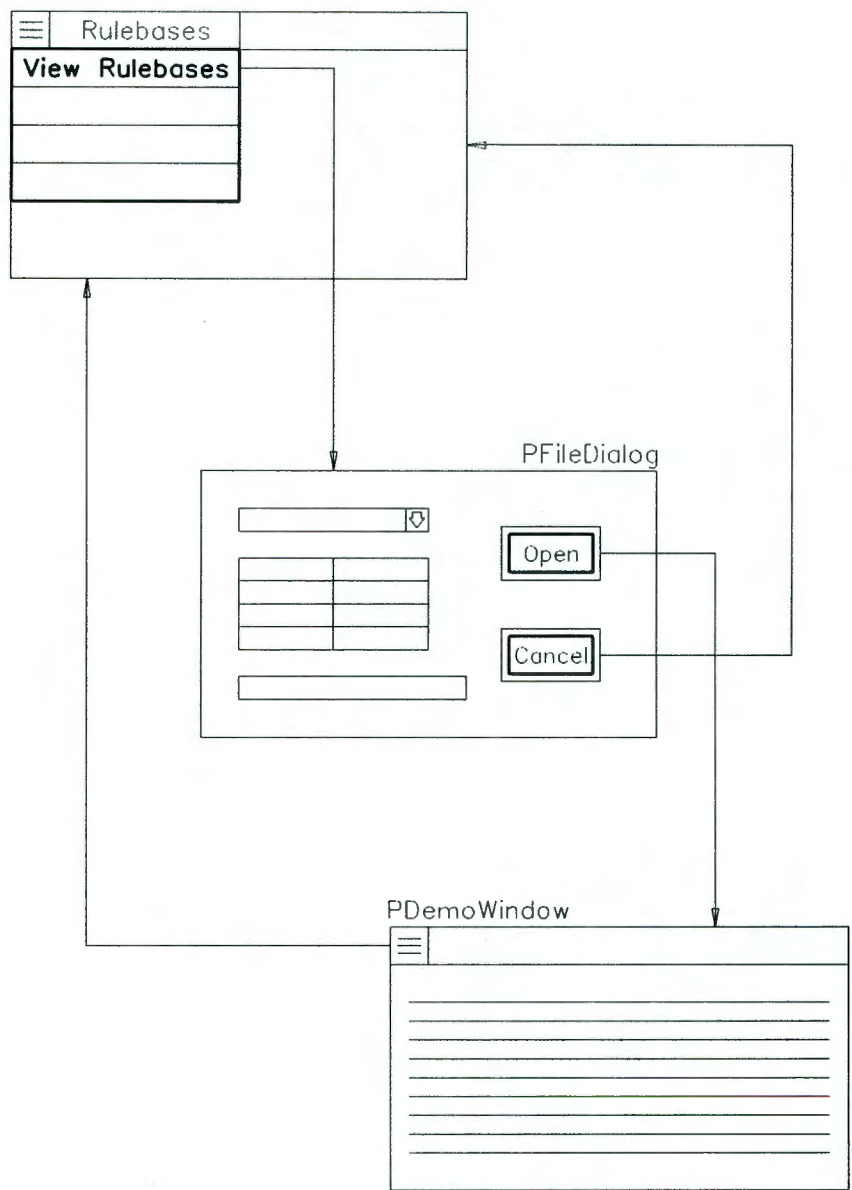
APPENDIX 6

Program Design Charts



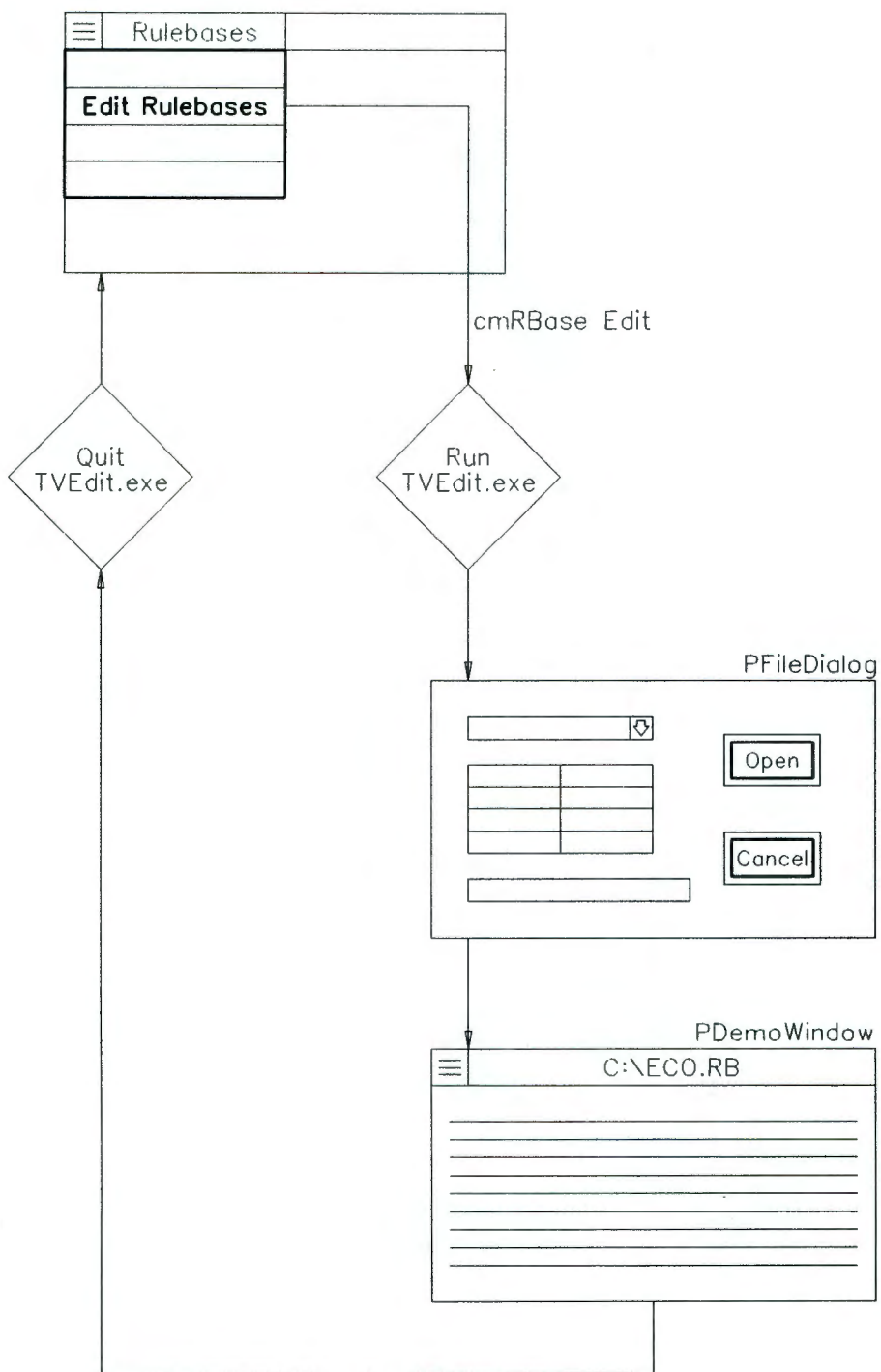
System Menu Structure

Figure 5.06



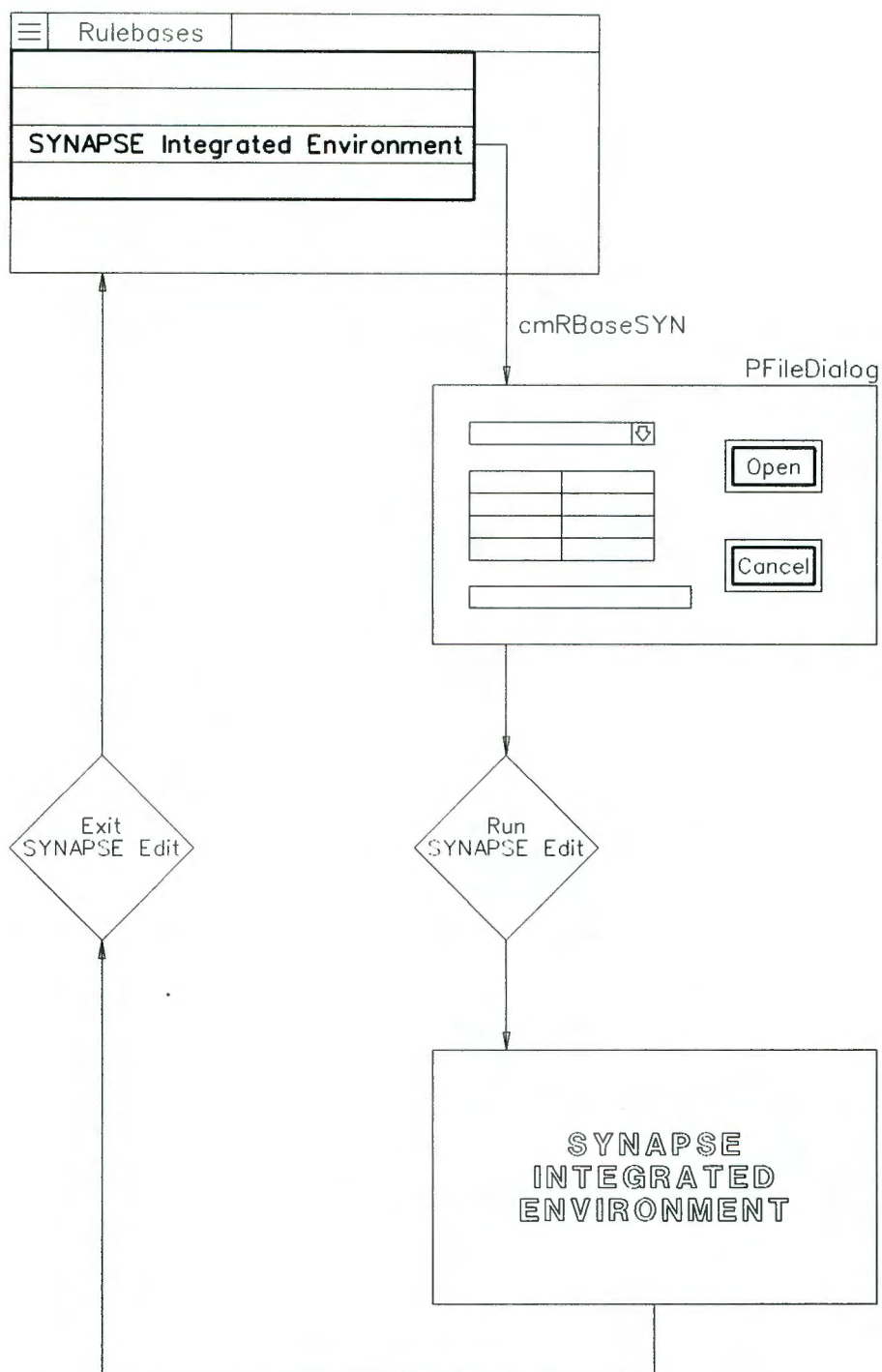
Rulebase Menu - View Rulebases

Figure 5.07



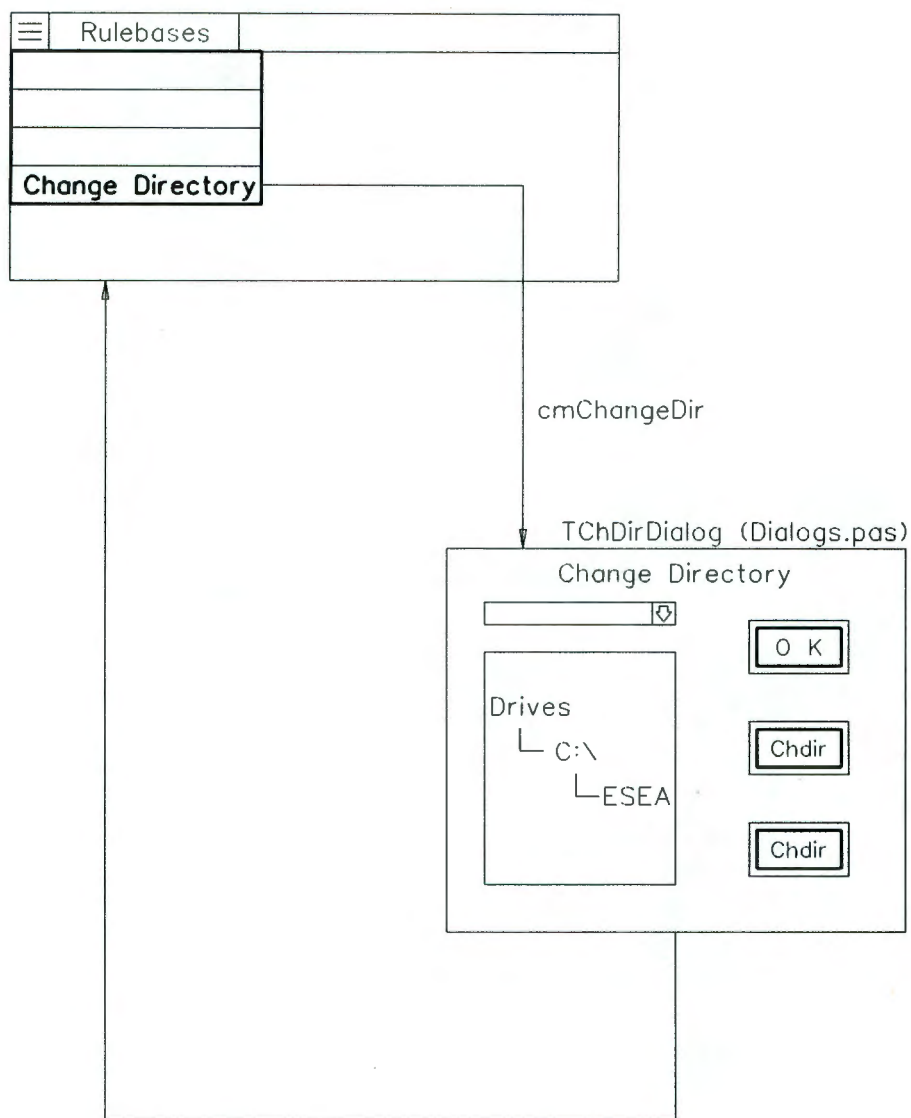
Rulebase Menu - Edit Rulebases

Figure 5.08



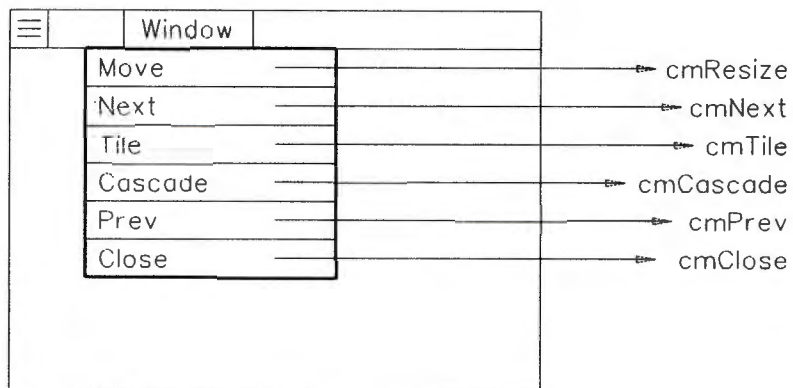
Rulebase Menu - SYNAPSE Integrated Environment

Figure 5.09



Rulebase Menu - Change Directory

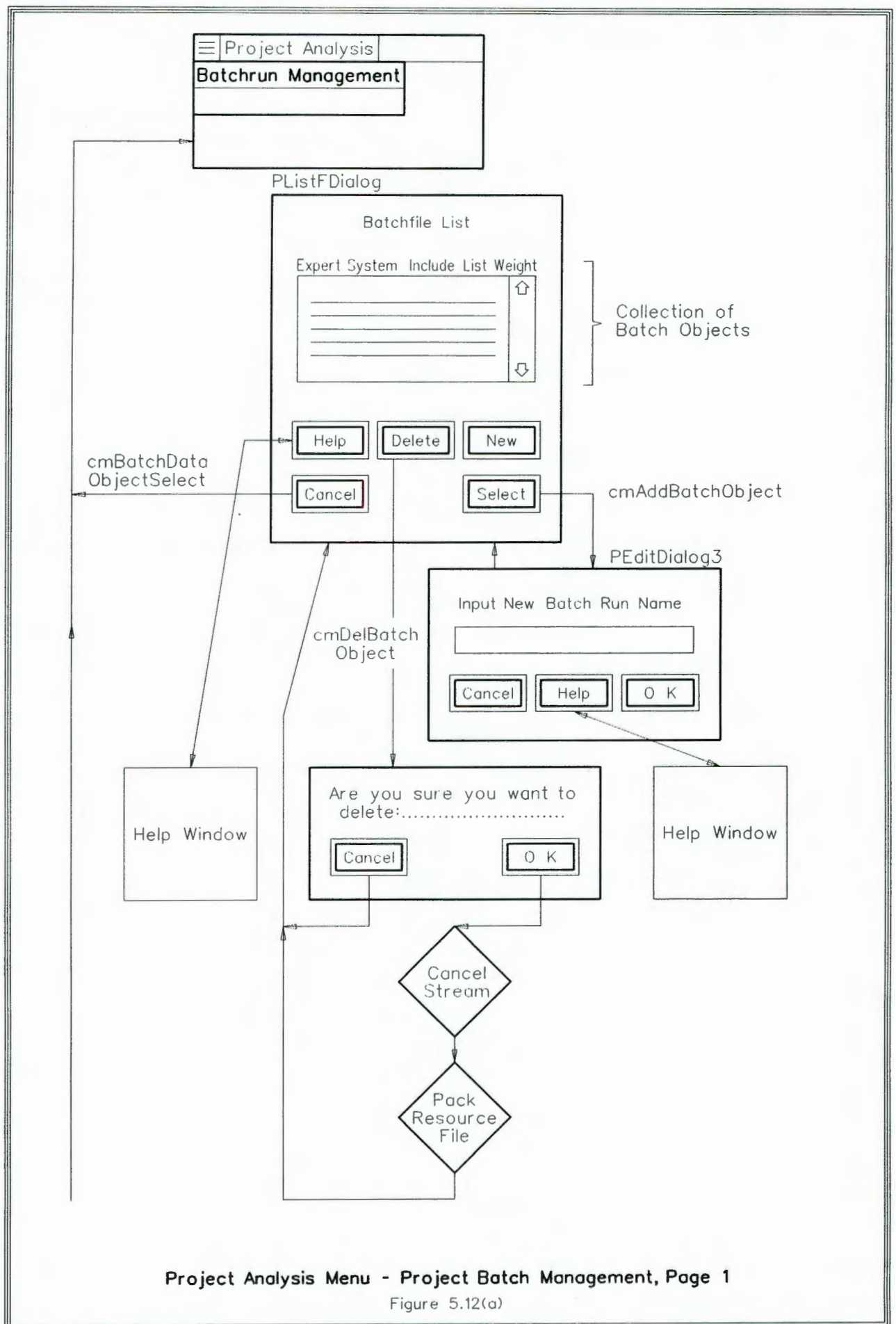
Figure 5.10

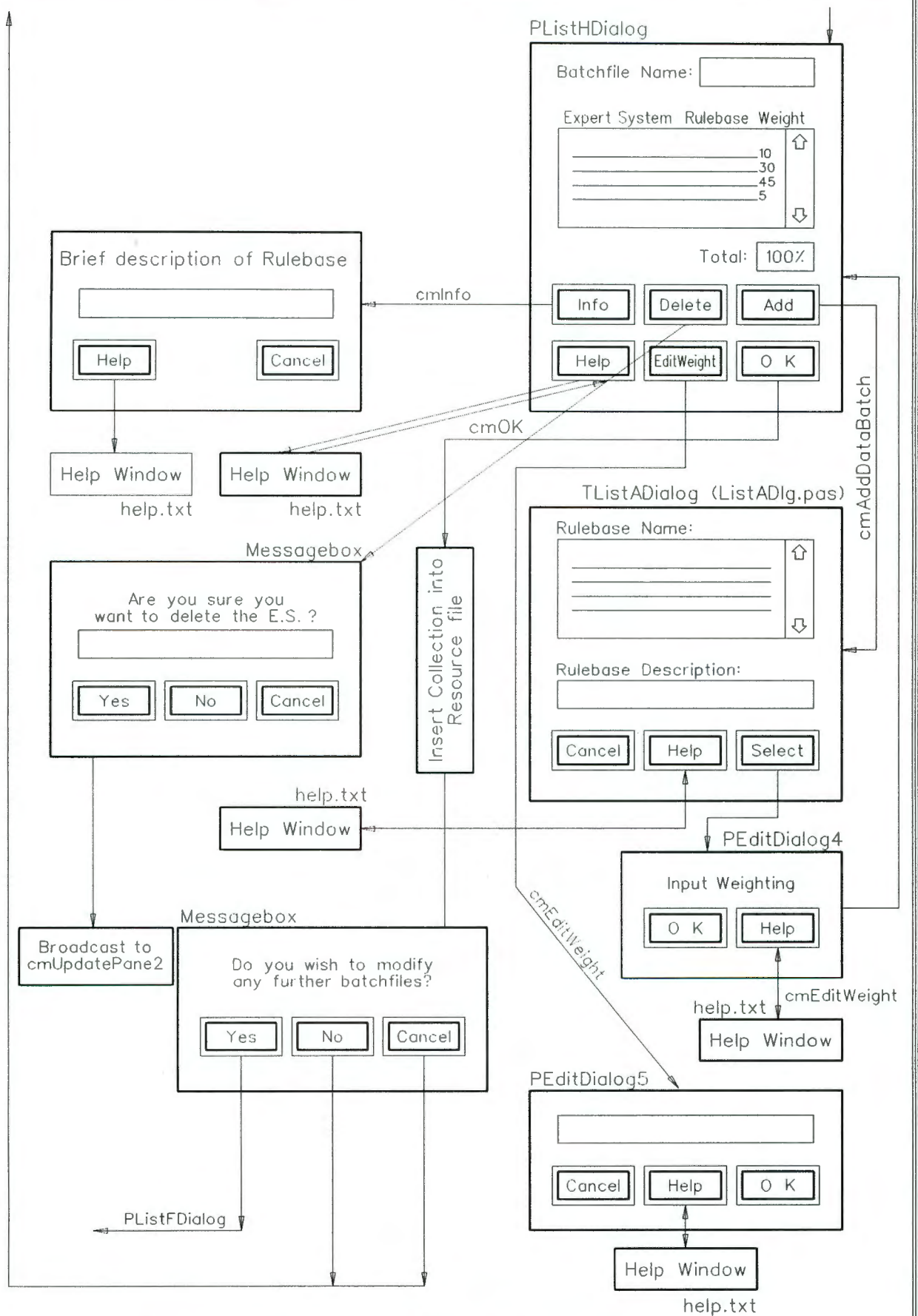


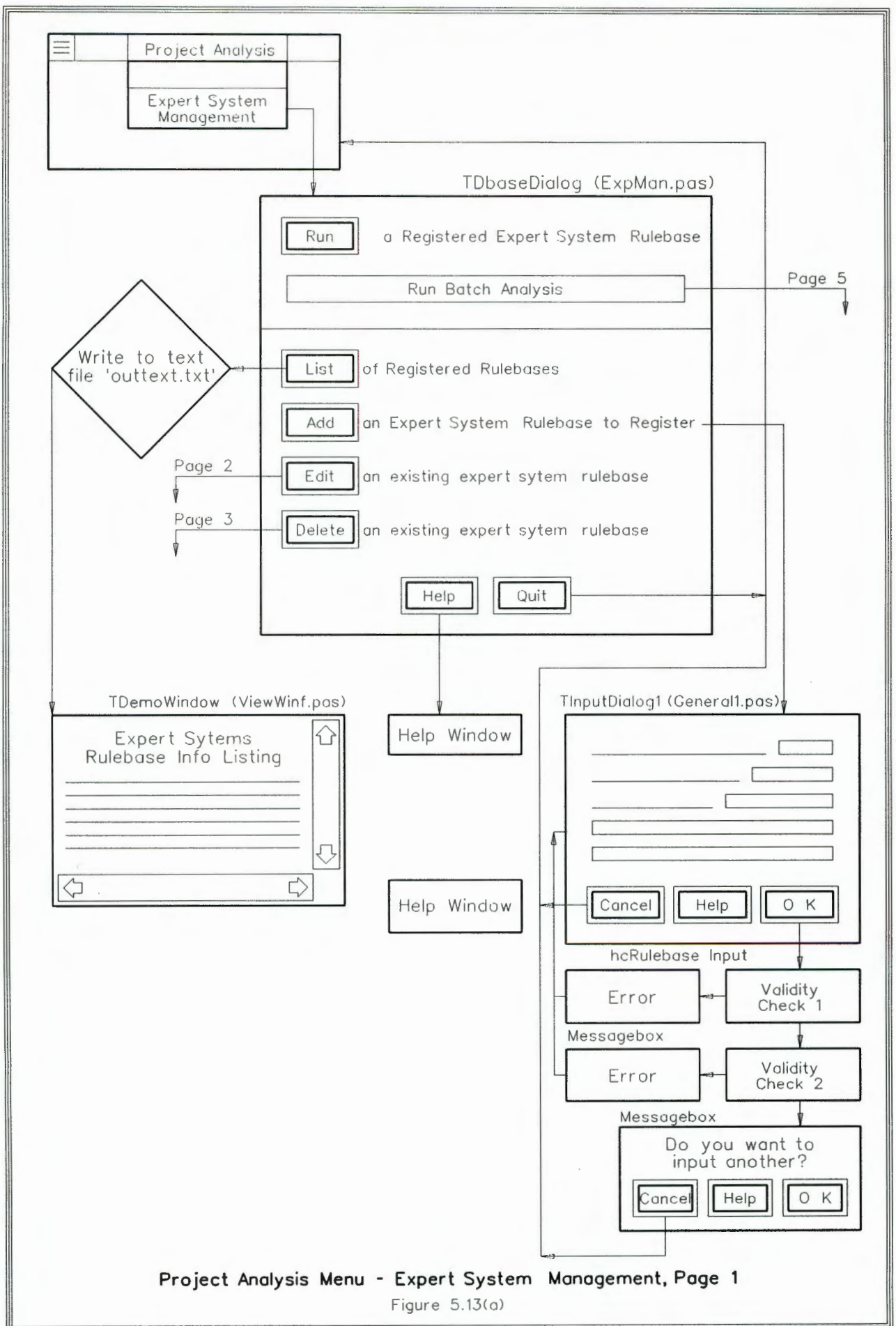
Move..... Move window to position requested
 Next..... Change focus to next window
 Tile..... Tile windows
 Cascade..... Cascade windows
 Prev..... Change focus to previously selected window
 Close..... Close window

Window Menu - Menu Commands

Figure 5.11







Project Analysis Menu - Expert System Management, Page 1

Figure 5.13(a)

