Improved data retrieval techniques for crash analysis

by

Aravind Gottemukkula

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Civil Engineering (Transportation Engineering)

Major Professors: Keith K. Knapp and Reginald R. Souleyrette

Iowa State University

Ames, Iowa

2001

Graduate College
Iowa State University

This is to certify that the Master's thesis of

Aravind Gottemukkula

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

"It's better to be one among hundreds
rather than one among millions"

To my Parents

"People who don't have dreams
don't have much"
- Unknown

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am thankful to all my committee members, Dr. Reginald Souleyrette, Dr. Keith Knapp, Dr. Alicia Carriquiry, and Mr. Dan Gieseman. Additionally, I would like to thank Dr. Reginald Souleyrette, and Dr. Keith Knapp for serving as major professors on my committee and all the time they have spent correcting my mistakes in writing. I can never forget all the weekends Dr.Souleyrette and Dr.Knapp spent to talk to me about my thesis despite their busy schedules. At the Center for Transportation Research and Education, I learned a lot from Mr. Dan Gieseman, who was so helpful and friendly, and I would not have completed this thesis without him. I always wondered how he could be so cool with so much workload. I wish I always had such a person as my boss. I also wish to thank Dr. Joyce Emery and Mr. Bob Schultz for providing me all the information I needed to complete my thesis successfully.

I have made friends with so many people here at Iowa State that I have never felt lonely in this alien country. Richard Storm and his brother Brandon, Jerry Shadewald and his wife Laura, Jason Stribiak, Eric Padget, Rich Herrick, Dan Smith, and my roomies Srini, Kamesh, and Raj are just a few of them. I thank everyone who made my life exciting here at Iowa State.

ABSTRACT

The existing crash analysis system used by the Iowa Department of Transportation (DOT), Access-ALAS is a node-based system. Crash Analysis with Access-ALAS is a tedious and time-consuming process. The design of Map Interface software to eliminate the use of node numbers to analyze crashes is described in this thesis. An Access-ALAS user survey done to verify Iowa DOT's requirements with the users' needs is also explained in detail. The survey instrument and the results are also included. The performance test done to evaluate Map Interface is also discussed. The Map Interface was found to be a significant improvement in the Iowa's crash analysis process.

CHAPTER 1. INTRODUCTION

Between 1975 and 1998 40,000 to 50,000 transportation-related fatalities per year occurred in the United States (*1, 2*) and the number of fatalities increased each year. Although the number of fatalities increased, accounting for the increase in vehicle miles traveled during the past two decades, there has been a 40 percent decrease in the fatality rate (*3*). This improvement in safety can be attributed to safety measures implemented over the past two decades.

A crash analysis is typically done prior to implementing a safety measure, and the results of this analysis can be used to improve the safety of a location or corridor by identifying the proper mitigation to reduce crash severity and/or the number of crashes at a location. The potential effectiveness of a safety measure can only be determined by adequate crash analysis. Figure 1.1 is a pictorial representation of the relationship between roadway safety and crash analysis results.

Edward Hughes (*4*) has stated that "...the overall process for the collection, reduction, management, and analysis of accident data can be stratified into six distinct processes". Figure 1.2 shows these six processes in an event sequence from crash occurrence to report generation. As shown in Figure 1.2, emergency vehicles and police arrive at the

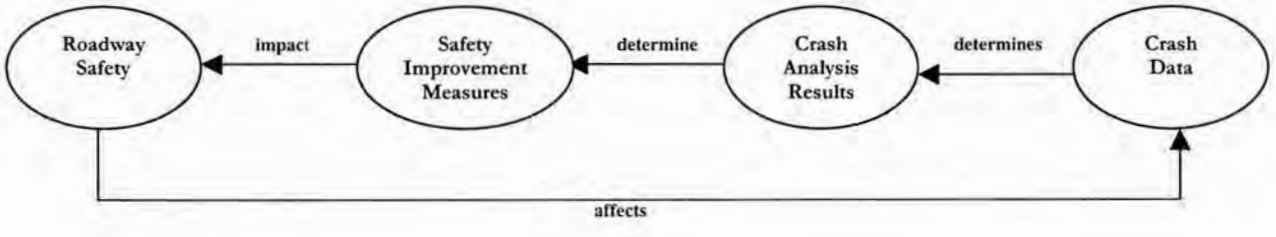


Figure 1.1. Relationship between Roadway Safety and Crash Analysis Results

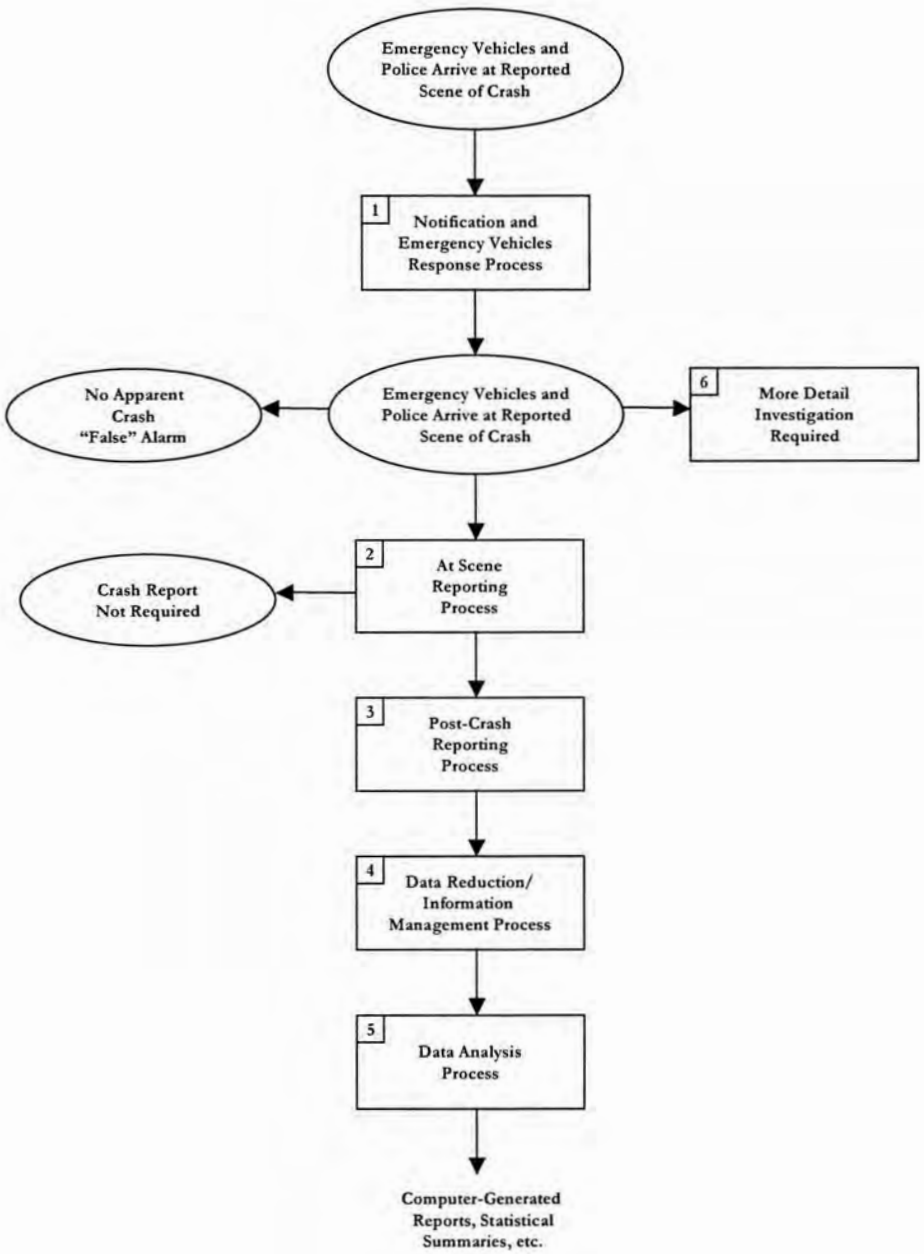


Figure 1.2. Crash Data Collection and Analysis Process (Adapted from *4*)

scene of crash on notification and report the process. The field reports are then used to enter crash data into a database and the unnecessary information is filtered out. A data analysis tool is then used to analyze crash data and generate reports and statistical summaries.

State departments of transportation, insurance agencies, and transportation consultants use a process similar to that shown in Figure 1.2 to evaluate the safety of roadways within the United States. There are, however, various data management systems used by these agencies to perform the specific crash data manipulation and analyses. For example, the Iowa Department of Transportation (DOT) uses both the Geographic Information Systems (GIS)-Accident Location and Analysis System (ALAS) and Access-ALAS. The ALAS systems are used to retrieve crash data and generate customized reports that help determine proper engineering (e.g., design), education, emergency response, and/or enforcement mitigation measures. The research described in this thesis produced tools to improve the existing Iowa DOT crash analysis package by providing map-based software to analyze crashes and generate reports.

Problem Statement

The Access-ALAS crash analysis approach requires the entry of node numbers to access and evaluate a crash at or near a particular location. Nodes, as defined by the Iowa DOT and within Access-ALAS, are generally located at roadway intersections and at points on a roadway with a significant change in alignment (e.g., the road ends, a railroad crossing, etc.). Each node has a unique number, but in Iowa there is no pattern to how these node numbers are assigned. In other words, the analyst must have a document (i.e., a map or list) that relates all the node numbers to a physical location. This node-based approach makes the crash data retrieval process very time-consuming.

The United States Department of Transportation (US DOT) established national highway safety program standards for the identification and surveillance of crash locations in the 1970s (*4*). The purpose of this standard was "...to identify specific locations or sections of streets and highways which have high or potentially high accident experience, as a basis for establishing priorities for improvement, selective enforcement, or other operational practices that will eliminate or reduce the hazards at the location so identified"(*5*).

In addition, in Access-ALAS the analyst does not have the ability to evaluate and analyze crashes between two points on a roadway that do not match designated node locations, and the user cannot select individual crashes. In other words, Access-ALAS has the ability to identify "...accidents that occurred within a particular segment of a roadway, such as a ramp or between intersections, at best, but it (i.e., Access-ALAS) lack[s] the accuracy necessary to analyze crashes that occurred at a specific point such as a gore area or at the end of a merge lane" (*6*). The ability to complete more specific crash analyses should help analysts make better safety improvement decisions (*6*).

Research Objectives

This research will enhance the capabilities of the Access-ALAS software crash analysis package by simplifying the retrieval of Iowa safety data. This simplification will be accomplished by meeting the following objectives:

- The incorporation of features needed to enhance the Access-ALAS software package, and already requested by the Iowa DOT. These features include:
  - The ability to select crashes in an entire county;
  - The ability to select crashes in an entire city;
  - The ability to select crashes in a user-defined area;

- The ability to select crashes along a roadway between any two points;
- The ability to select crashes between intersections with intersection cross-street names as the input, rather than node numbers; and
- The ability to select any individual crash set of crashes.

- The completion of an Access-ALAS user survey to determine user needs, and comparison of the survey results with the improvements requested by the Iowa DOT.
- The completion of a performance test to compare the analysis speeds of the Access-ALAS software package with and without the improvements.

A map interface will be designed as part of this research and it will simplify the crash analysis process by allowing the crashes to be selected more easily. This ability will be provided through a user-friendly Graphical User Interface (GUI). For example, with the map interface, the user will have the ability to select an individual crash, or the crashes on any segment of roadway and within a particular user-selected area. The GUI will help the safety analyst perform crash analysis faster and this will allow more time for the determination of potential solutions to transportation safety concerns. These enhancements to Access-ALAS should improve the transportation safety decision-making capabilities of safety analysts in Iowa.

Thesis Organization

This thesis is organized into six chapters. In Chapter 1 the background of crash analysis systems and an overview of this thesis are presented. Chapter 2 primarily focuses on the crash analysis systems, and the data collection and management methodologies used by different states. The results of the Access-ALAS user survey are summarized and analyzed in Chapter 3. The methodology used to develop a map interface and the features incorporated

into it are discussed in detail in Chapter 4. Also, in Chapter 4, crash analyses speeds before and after the Access-ALAS improvements done in this research are described. Chapter 5 includes a summary of the conclusions and recommendations from this research.

CHAPTER 2. STATE OF THE PRACTICE

This chapter focuses on different crash analysis systems, and the crash data collection and database management techniques used by different states. The importance of maintaining data and using a crash analysis system to determine safety measures is emphasized. The systems and methodologies used by different states are compared to Access-ALAS wherever appropriate. The description of these systems for several states is followed by the description of the analysis system (i.e., Access-ALAS) used in Iowa. A detailed explanation of the Access-ALAS software package accompanied by screenshots from the software is also included. Also, the importance of GUIs and maps are explained in conjunction with the reasons for designing a map interface.

Data Collection and Crash Analysis Outside Iowa

State departments of transportation and state Police use crash analysis software to determine mitigation measures that may enhance the safety of an intersection and/or a roadway. Some states analyze crashes using automated crash analysis systems while others analyze crashes manually. Regardless of the methodology or the crash analysis tool used, a crash database is needed. Several states have developed their own crash databases. The characteristics of these crash databases and/or the crash analysis systems used to analyze the information in these databases are discussed in the following paragraphs.

The Washington DOT (WSDOT) uses a milepost system to locate collisions on its highways. The goal is to locate each collision to the nearest 0.01 mile. Once the data are collected from collision reports, WSDOT stores the data in a mainframe data storage system called an Adaptable Database (ADABAS) environment, and the database is called Transportation Information and Planning Support (TRIPS), which allows WSDOT to

produce standard crash histories for specific state highway sites, as well as identify locations, which have a statistically significant number of collisions. For more detailed analyses of the crash data, the Statistical Package for Social Sciences (SPSS) software is used to produce frequencies, cross tabulations and other customized reports (*7*). Analogous to Access-ALAS, WSDOT has also developed a Collision Records System (CRS) that helps local agencies identify safety problems on their local street systems. CRS stores the collision data in an Access database, and Access is used to assist with the generation of specialized crash summary reports by location, collision frequency, and corridor (*8*).

An automated collision database and reporting system developed for Nashville, Tennessee makes the near real-time entry and analysis of collision records possible (*9*). This system also allows for enhanced manipulation of collision data and increases analysis efficiency and capabilities. Similar to Access-ALAS with improvements, the automated collision database and reporting system of Nashville was also developed using Microsoft Visual Basic 6.0.

Idaho's Crash Analysis Reporting System (CARS) is a Graphical User Interface (GUI) that runs within Statistical Analysis Software (SAS). This package allows the user to produce crash listings, frequency tables and cross-tabulations. CARS also allows the user to write his/her own report using the SAS programming language. Finally, the series of queries and subsequent output reports allowed in CARS can be combined to produce the statewide annual collision report for Idaho. Unlike a map interface produced in this research, CARS requires a relatively expensive annual renewal of the SAS license (*10*).

The Vermont crash reporting system is a GUI that is used to populate their crash database by manually typing in the information from paper crash reports. A spreadsheet

package is used to analyze crashes and sometimes the Access query wizard is used to query crashes. Crash summary reports can be printed using the current program based on town and route location input. Access-ALAS with improvements in Iowa, and the Vermont crash reporting system were designed using the same programming language (Visual Basic) (*11*).

In Virginia, the DOT (VDOT) prepares crash reports and forwards them to the Department of Motor Vehicles (DMV) and information about the vehicles, drivers, and passengers involved in the crash are entered into a database. VDOT then receives these reports back from the DMV and additional information relating to roadway and weather are entered. VDOT has also created digital datasets of the highway system to produce and update maps on a county-by-county basis. Crash data is extracted from the VDOT system with the route/node/offset location converted into a route/milepoint and brought into ArcView as an event theme (a shape file) relating the route/milepoint in the crash database to the roadway-measured shape. A map of crash points is then saved and displayed on VDOTs Intranet web page (*12*).

In Connecticut, the DOT (ConnDOT) stores information in a flat file database on a mainframe computer. The crash data is analyzed with Formula Translation (FORTRAN) and Common Business Oriented Language (COBOL) programs. ConnDOT plans to use an analysis software developed with Visual Basic. This software will be used with an Oracle-based server (*13*).

The West Virginia DOT (WVDOT) receives police reports containing crash information and then enters data into the crash records database. Various batch and SAS programs are executed to identify sites for detailed analysis. The crash analysis software used by the WVDOT is designed in COBOL and is text-driven (*14*).

In Indiana, the state police are responsible for creating and populating a crash database that is primarily a pseudo-number (i.e., a unique number represents each highway, road, and street name in a county) location system. The Indiana DOT maintains and provides this pseudo-number listing. Efforts are being made to locate crashes on a map electronically (*15*). Crash data are retrieved from the crash database and analyzed by querying the pseudo-number system.

The GIS-Based Crash Referencing and Analysis System developed with Highway Safety Information System (HSIS) data for the area of Wake County, North Carolina provides the functions needed to edit tabular and spatial crash and roadway data and perform crash analysis (*16*). This referencing and analysis system indicates that the integration of traditional crash analysis systems with a GIS provides spatial referencing capabilities (*16*), and therefore incorporating mapping capabilities can help develop a more comprehensive crash analysis program. Access-ALAS with additional features in Iowa is another example of a GIS-based crash analysis system. Table 2.1 summarizes state of the practice outside Iowa.

Crash Data Collection and Analysis in Iowa

Access-ALAS is used to retrieve crash record information about vehicle collisions that occur on any roadway within the entire state of Iowa. This information is then used to determine the contributing factors or circumstances that may have caused the crash or crashes (*17*).

Table 2.1. Summary of the State of the Practice Outside Iowa

State	Database used	Crash Analysis Tool/Methodology Used
Washington	Transportation Information and Planning Support	Collision Records System
Tennessee	Automated Collision Database and Reporting System	Automated Collision Database and Reporting System
Idaho	Unknown	Crash Analysis Reporting System
Vermont	Crash Reporting System	Spreadsheet package (e.g., Excel)
Virginia	Unknown	Interactive Web Maps
Connecticut	A Flat File Database	Fortran and COBOL
West Virginia	Accident Records Database	COBOL-based Crash Analysis Software
Indiana	Crash Database	A Querying Software
North Carolina	GIS-based Crash Referencing and Analysis System	GIS-based Crash Referencing and Analysis System

As the name indicates, Access-ALAS was designed using Microsoft Access, but it can be executed on a computer without Microsoft Access installed. Access-ALAS performs two distinct functions (*18*):

1. Identification of crash locations
    - Statewide
    - Countywide
    - Citywide
    - By route
    - By node
    - By link
    - By String of nodes

2. Analysis of the crash data by six data categories

    - Crash – Day of the week, Time of Day, Type of Crash, Major Cause, Type of Collision, and Intersection Class
    - Roadway – Class, Character, and Geometry
    - Environment – Locality, Light Conditions, Surface Conditions, and Weather
    - Vehicle – Type of Vehicle Involved, Roadway Environment, Traffic Controls, Fixed Object Struck, Special Use Vehicle, Vehicle Attachment, Speed Limit, and Vehicle Action
    - Driver – Age of Driver, Gender of Driver, Restriction Compliance, and Driver Changed
    - Injury – Severity, Protective Devices, and Position

Access-ALAS also generates crash summary reports in five different templates, which include reports by major cause, day of the week/time of the day, surface condition/light condition, and case list summaries. Screenshots of some features of the Access-ALAS software package are shown in figures 2.1 through 2.5.

Graphical User Interface (GUI) and Maps

The primary purpose of this section is to emphasize the importance of a GUI-based design of software and why a GUI was used to design Access-ALAS and a map-based interface. A GUI is defined as "...a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use" (*19*). Programs with a GUI are easier to use when compared to text/command-driven interfaces. In a text/command-driven interface the users have to memorize all the commands necessary to run the program, but GUIs are generally self-explanatory and intuitive. Neal Peterson (*20*) stated that "GUIs are more intuitive, make it easier to transition between software programs, better for the visual learner, and make it easier for new users to learn programs".

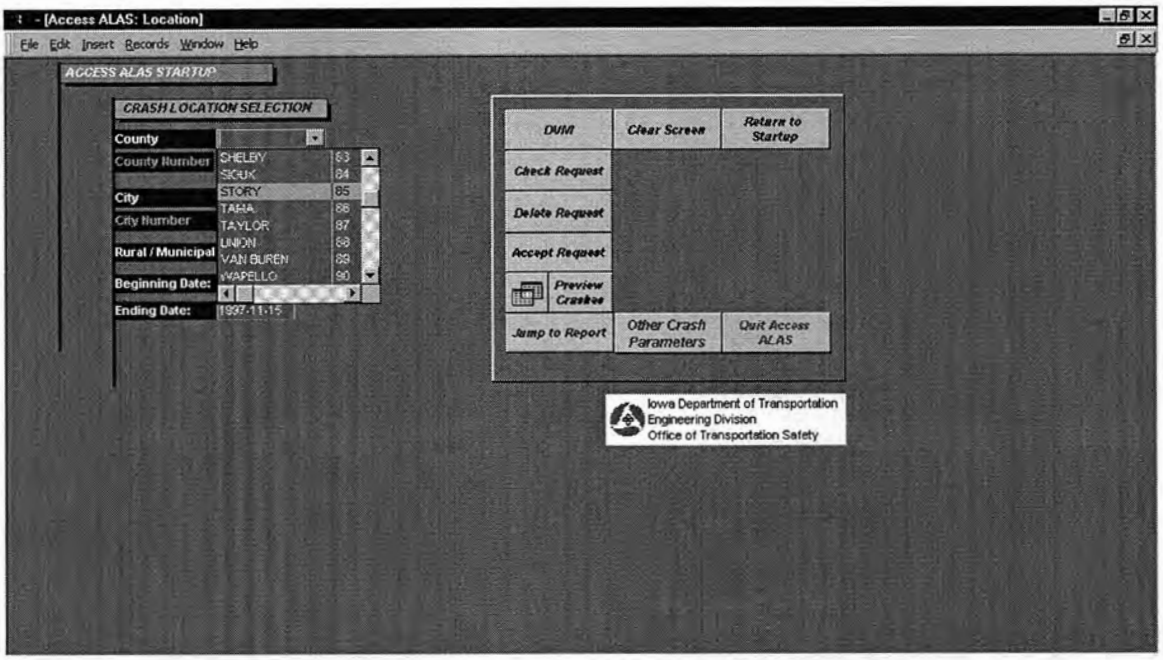


Figure 2.1. Countywide Crash Analysis Screen

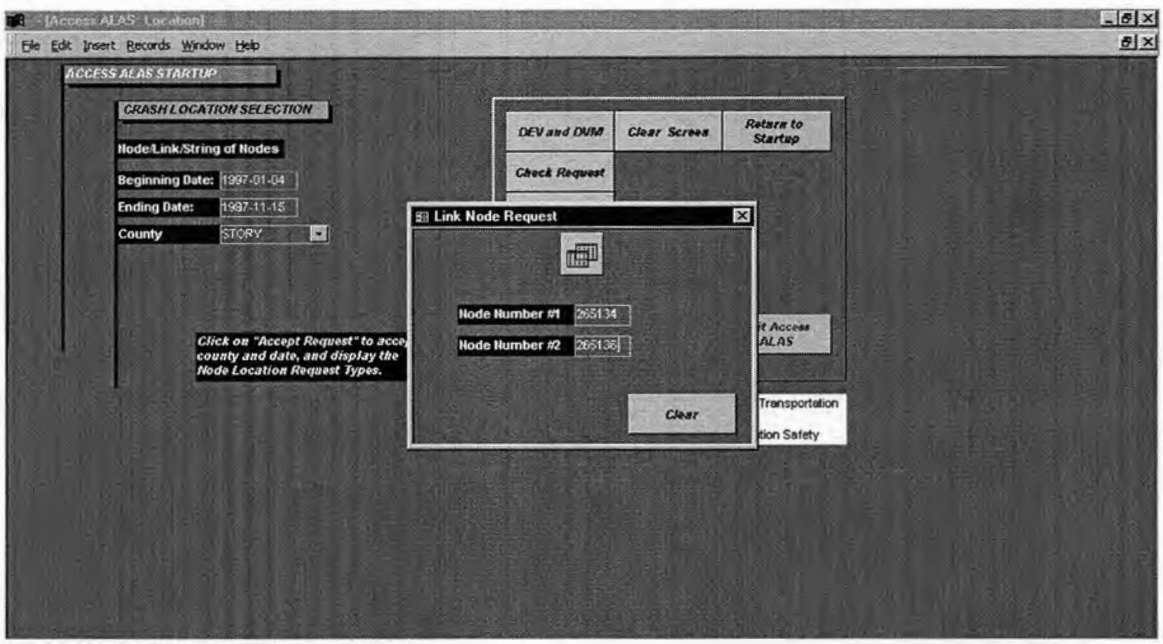


Figure 2.2. Link between Nodes Request Screen

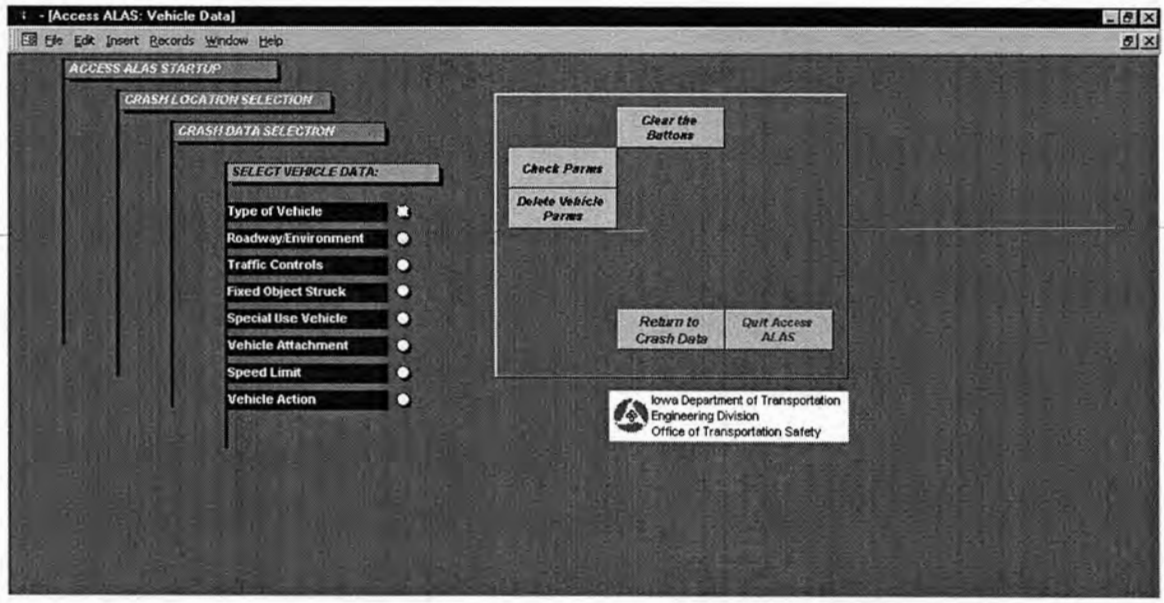


Figure 2.3. Vehicle Data Entry Screen for Crash Analysis

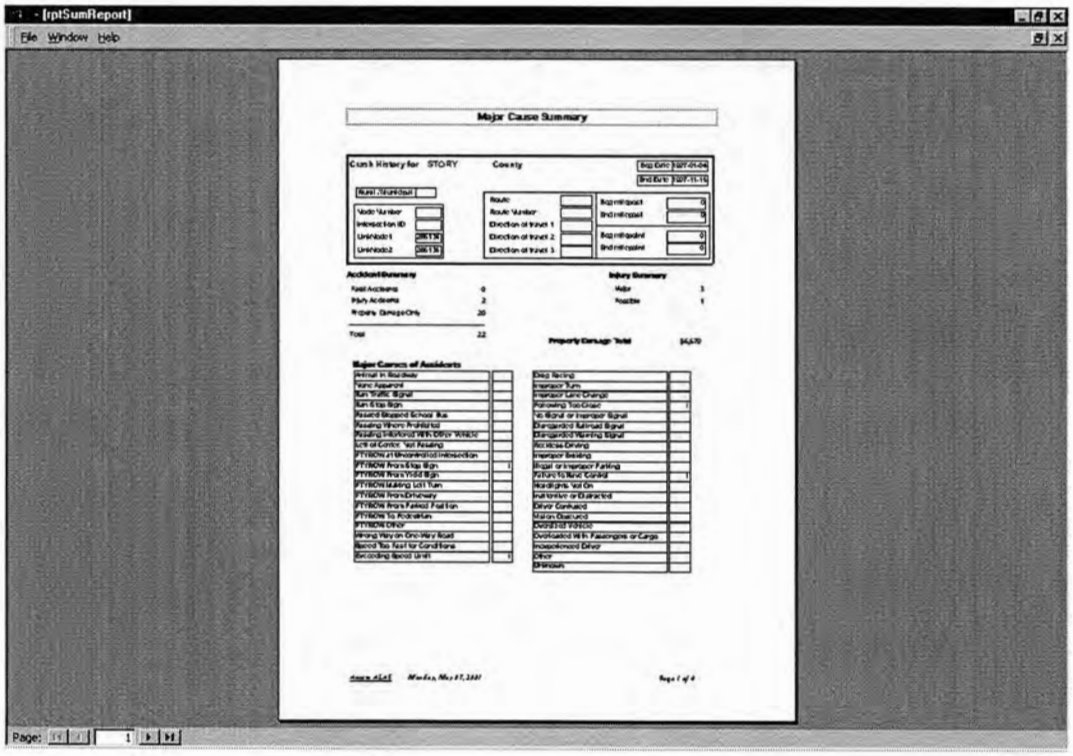


Figure 2.4. Access-ALAS Major Cause Summary Report

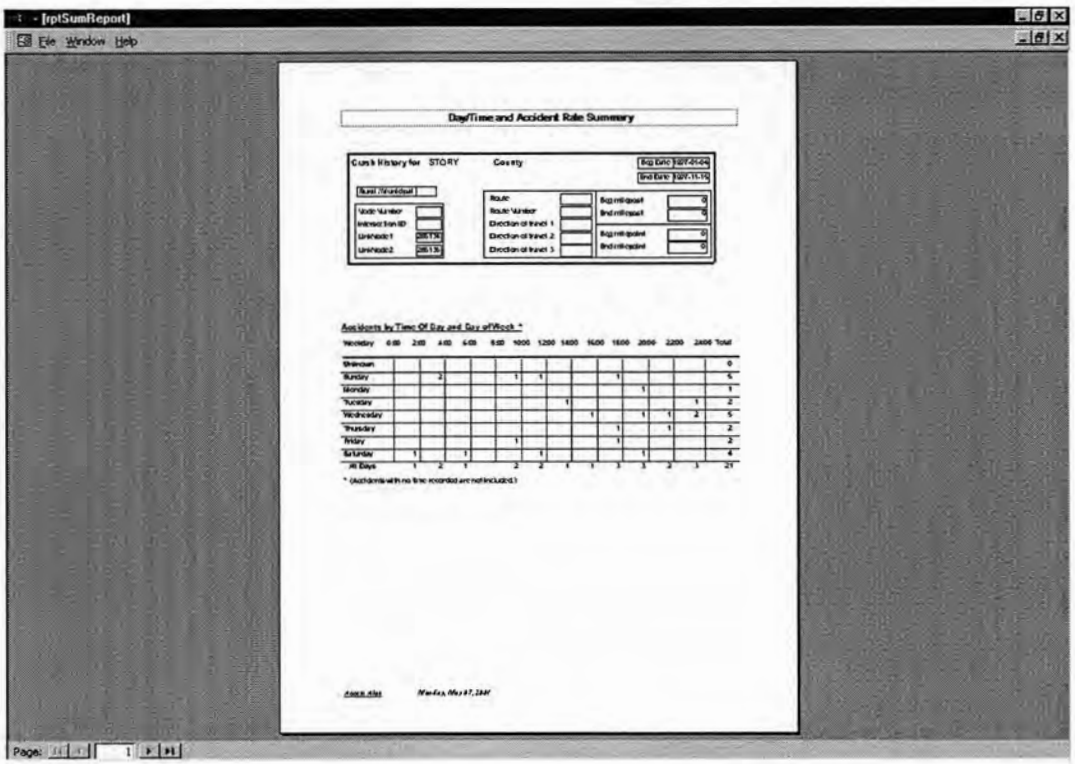


Figure 2.5. Access-ALAS Report Showing Day/Time Matrix

GUIs are now accepted and it is expected that all software will eventually be graphics-driven with high quality sound and video (*21*). Both Access-ALAS with and without additional features are GUIs that provide the user with easy-to-use features and functions such as icons, windows, menus and single-click command buttons. A GUI helps the user concentrate more on the results of the analysis rather than the memorization of commands.

The main difference between Access-ALAS with and without additional features is in the crash selection process. Access-ALAS requires the use of nodes, county and city names, or mileposts to select crashes whereas with Access-ALAS with the additional features, crashes can be selected by pointing to a location on a map. Map interfaces make the crash selection process easier for the user by providing a map of the actual location of a crash. In

general, maps are more intuitive and easier to understand when compared to a series of directions in text.

Safety analysts form the most important group in the crash analysis process, as they are the end-users. Therefore, feedback from users was essential to design an acceptable and robust interface. Chapter 3 describes the Access-ALAS user survey instrument and explains the results of the survey.

CHAPTER 3. ACCESS-ALAS USER SURVEY AND ANALYSIS

The Iowa DOT requested that the project team design a map interface that will be highly intuitive for users. To facilitate this, a survey of user needs and preferences was designed by the project team with input and oversight from the Iowa DOT and Access-ALAS trainers. Surveys were mailed to 290 Access-ALAS users. One hundred and three surveys were completed and returned (representing a response rate of 36 percent) (see Chapter 5 for a discussion on potential bias in the sample). This chapter summarizes the survey responses and their implications on software design.

The user survey consisted of questions on performance and user-friendliness of current Access-ALAS software (see Appendix A for survey questionnaire). Responses to each question are analyzed separately and graphically presented herein. Written comments received along with survey returns are also presented. Finally, survey results are compared to Iowa DOT's needs as presented in Chapter 1.

Survey Responses and Analyses

Question 1

Please indicate whether you use Access-ALAS for any of the reasons stated below.

Reason	Yes	No
a) Improve Law Enforcement	1	2
b) Improve Engineering	1	2
c) Improve Emergency Response	1	2
d) Improve Education	1	2
e) Other (Please Specify)	1	2

Table 3.1. Use of Access-ALAS

Purpose	Number of Subjects
Improve Law Enforcement	8
Improve Engineering	63
Other	3
None	3
Multiple Purposes	25
Total Number of Subjects Responded	***102****

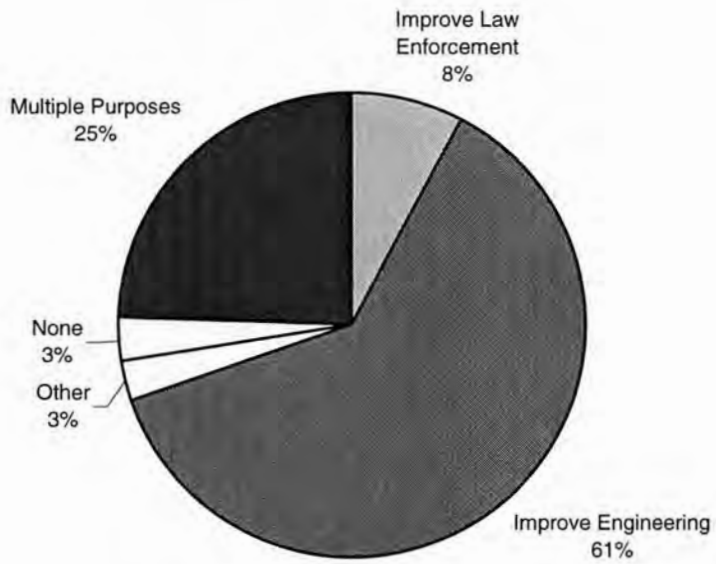


Figure 3.1. Use of Access ALAS

Interpretation

Access-ALAS is a crash analysis system, which is primarily used to improve safety on Iowa's road network. The above results reflect conventional approaches to improving safety through engineering and law enforcement. The majority of respondents use Access-ALAS for engineering analysis.

Question 2

How often do you use Access-ALAS?

1 - Once a week
2 - Once every two weeks
3 - Once a month
4 - Once every six months
5 - Once a year or less
6 - Other (please specify) _____

Table 3.2. Frequency of Use

	Subjects
Once a week	6
Once every two weeks	8
Once a month	29
Once every six months	31
Once a year or less	16
Other	13
Total Subjects Responded	***103***

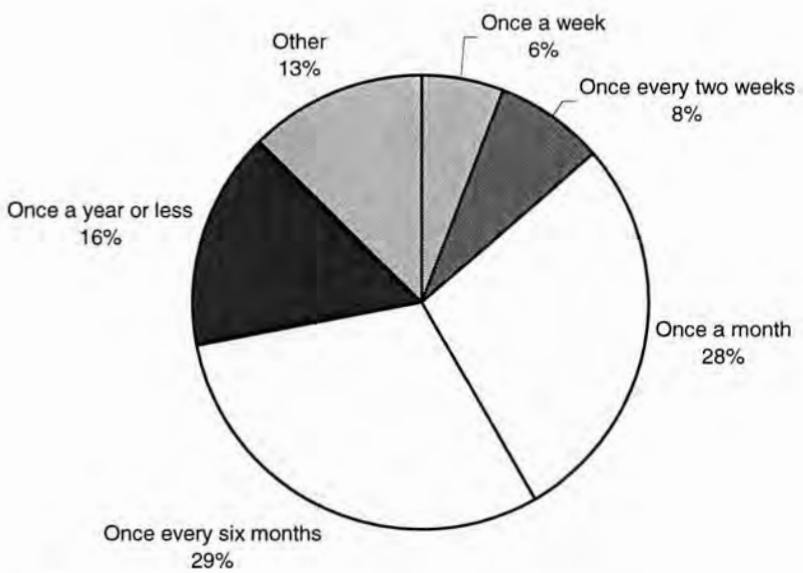


Figure 3.2. Frequency of Use

Interpretation

Over half of the respondents use Access ALAS no more frequently than once per month. Clearly, the software should be designed to be as intuitive as possible with such infrequent use.

Question 3

Have you used any other crash analysis software?

1 = Yes
2 = No

Table 3.3. Indicated use of other Crash Analysis Software

	Subjects
Yes	18
No	81
Total	**99**

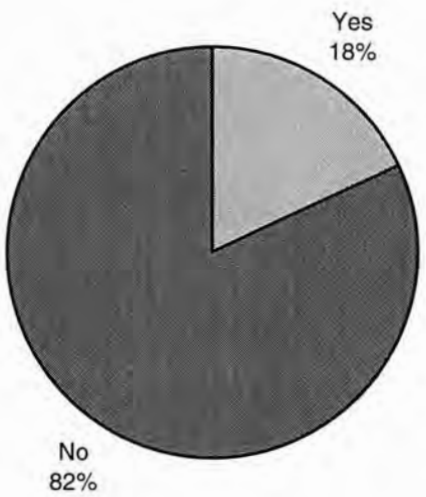


Figure 3.3. Indicated use of Other Crash Analysis Software

Interpretation

Only 18 percent of respondents indicated use of other crash analysis software. These include Intersection Magic, GIS-ALAS and the predecessor to Access ALAS, PC-ALAS.

Question 4

If you have used other crash analysis software, which is your preferred package?

1 = Access-ALAS
2 = Other (please specify) _____

Responses

Table 3.4. Preferred Crash Analysis Software

	Subjects
Access-ALAS	15
Other	2
Total	***17***

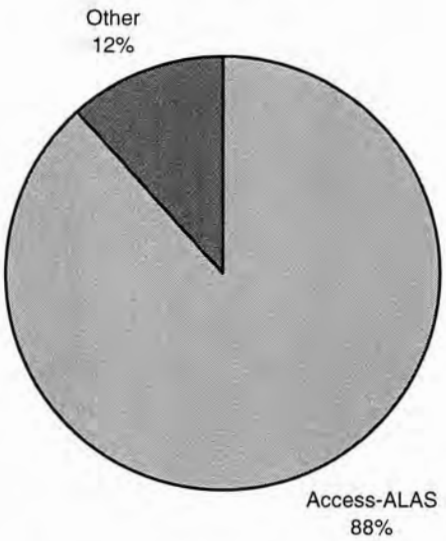


Figure 3.4. Preferred Crash Analysis Software

Interpretation

Despite its limitations, Access-ALAS is the software of choice for 88 percent of respondents who have used other software.

Question 5

Which of the following descriptors of Access-ALAS do you agree with?

(Circle all that apply)

1 = User-friendly

2 = Powerful querying capabilities

3 = Good reporting features

4 = Convenient or useful node-based approach

Table 3.5. Useful or Convenient Features of Access-ALAS

	Subjects
Querying Capabilities	3
Reports	3
Node-based Approach	1
All the Features	2
More than one feature	11
None	83
Total	***103***

Interpretation

As shown in Figure 3.5, only 20 percent of respondents indicated regard for any Access-ALAS features, which perhaps explains some of the observed infrequent use (however, this is speculative as users were not specifically asked to explain infrequent use). Of special interest was the observation that only one respondent found the node-based crash selection feature to be useful or convenient. A principal benefit of the additional features is to eliminate the mandatory use of nodes to perform crash analysis.

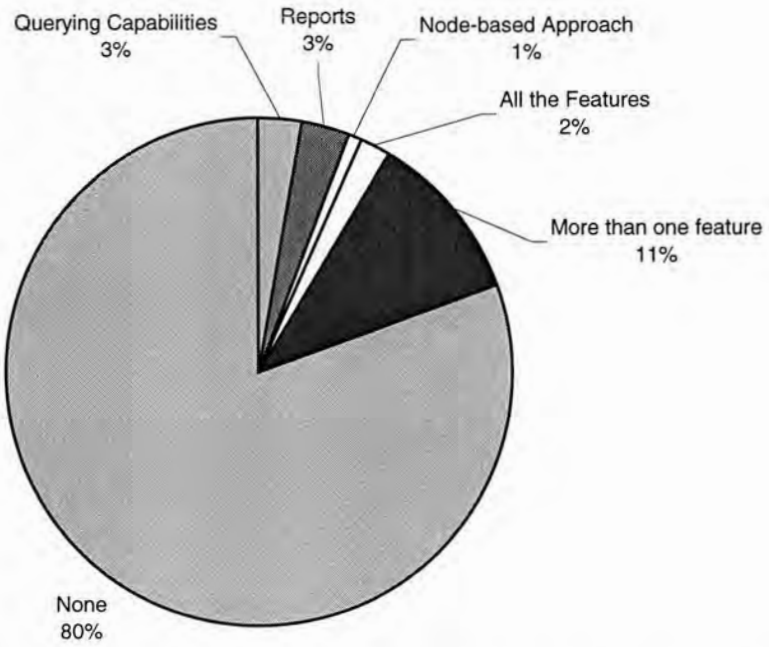


Figure 3.5. Useful or Convenient Features of Access-ALAS

Question 6

Indicate your opinion of the user-friendliness of Access-ALAS is (please use a scale where "1" is not user friendly at all and "5" is very user friendly)

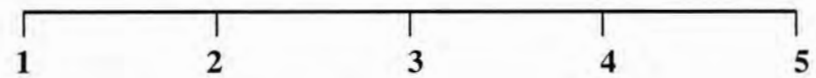


Table 3.6. Access-ALAS User Friendliness

	Subjects
Very User-Unfriendly	7
User-Unfriendly	16
Average	44
User-Friendly	29
Very User-Friendly	0
No Response	7
Total	***103***

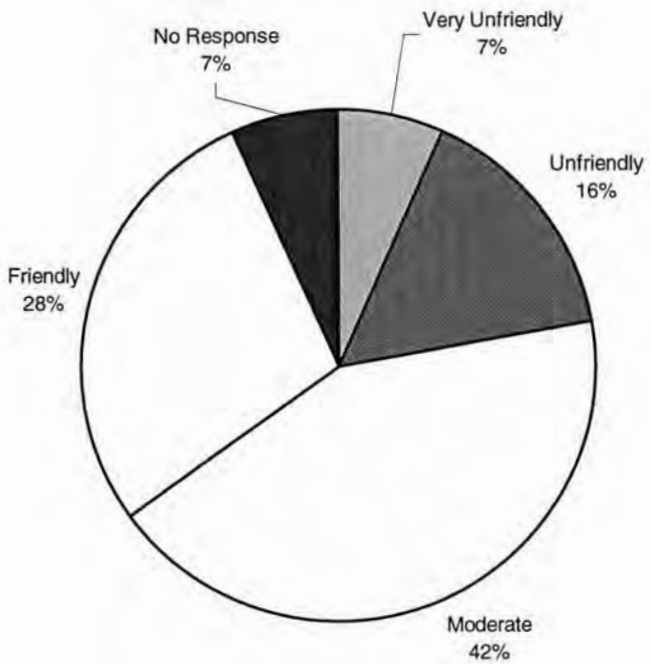


Figure 3.6. Access-ALAS User Friendliness

Interpretation

While user-friendliness alone does not seem to indicate a pressing need for software improvement, fewer than one-third of respondents indicated that Access-ALAS is user friendly.

Question 9

How efficient is using node numbers to get information about crashes? Please use a scale where "1" means not efficient at all and "5" means very efficient.

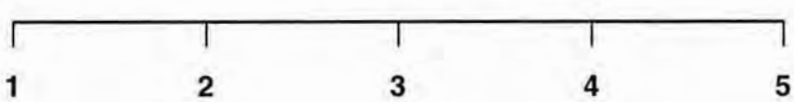


Table 3.7. Efficiency of Node-based Crash Selection

	Subjects
Very Inefficient	19
Inefficient	22
Average	30
Efficient	24
Very Efficient	3
No Response	5
Total	***103***

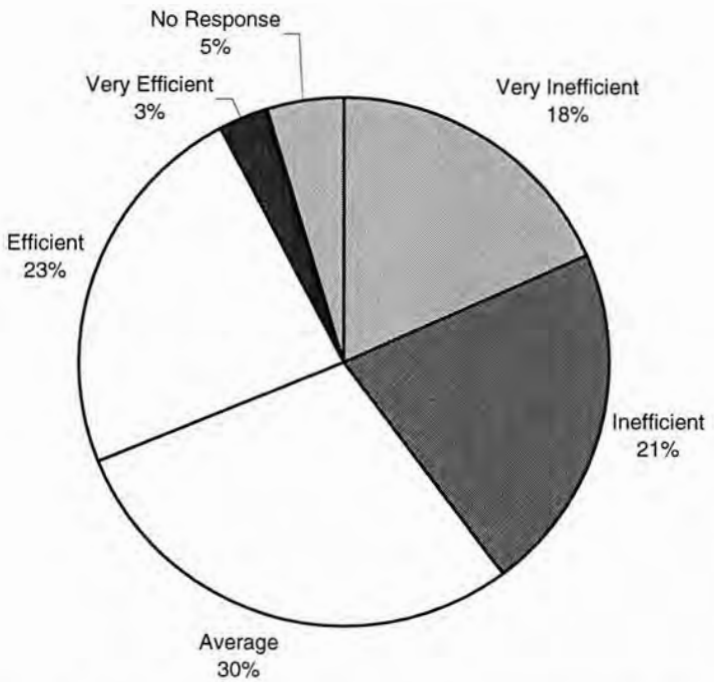


Figure 3.7 Efficiency of Node-based Crash Selection

Interpretation

Nearly 70 percent of respondents consider node-based specification to be inefficient, substantiating the need for an improved selection method.

Question 10

Do you think you need improved software for crash analysis?

1 = Yes
2 = No

If yes, what kind of improvements would you like to have in a future version?

_____

Table 3.8. Need for Improved Crash Analysis Software

	Subjects
Yes	60
No	31
Total	***91***

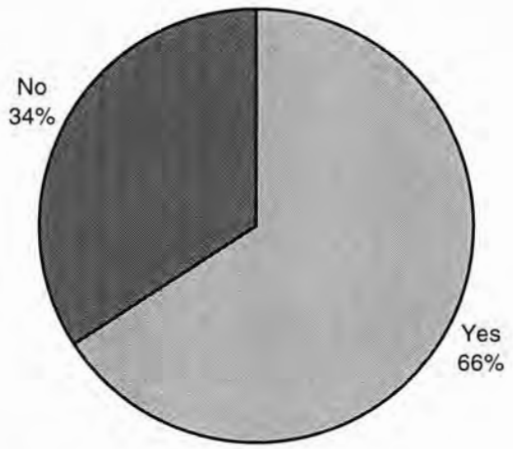


Figure 3.8. Need for Improved Crash Analysis Software

Interpretation

A majority of respondents indicated (unspecified) need for software improvement. Specific comments are summarized in the following section.

User Suggestions and Comments

In addition to the responses listed above, Access-ALAS user who responded to the survey expressed several other concerns for improvement. Some users expressed that with current Access-ALAS, training is absolutely required, and suggested that a more intuitive, user-friendly design may eliminate the need for training. Additional features were also requested, such as:

- Collision diagram sketches;
- Graphs;
- Charts; and
- Capability of assessing the benefit of mitigation measures.

Several users explicitly called for a map as an interface to obviate the need for node numbers, e.g., "Mapping capabilities to show actual accident locations graphically would be a very large improvement," and "ability to highlight area of interest and produce statistics for the highlighted area."

CHAPTER 4. SOFTWARE DEVELOPMENT AND FEATURES

This chapter describes the software development process, the logic involved with the map-based interface and the features added to Access-ALAS with it. The Visual Basic code is provided in Appendix B.

Software Development and Logic

The logic involved in the Map Interface 1.0 software was designed and represented in a graphical form using the Unified Modeling Language (UML). Grady Booch et al. (*22*) define UML as a "…language for specifying, constructing and documenting the artifacts of a software-intensive system". According to Grady Booch et al. flowcharts enable system visualization, structure and behavior specification, efficient system construction, and decision documentation (*22*).

Figure 4.1 shows a flowchart representing the flow of control within the software program developed, as part of this research. This program is called Map Interface 1.0 The flowchart shown in Figure 4.1 is the most simplified graphical representation for Map Interface 1.0. The large and small rectangles in this figure represent Map Interface 1.0 and Access-ALAS respectively. The four bubbles in the larger rectangle represent the four major components of the Map Interface1.0.

Geography Loader is responsible for loading the user-specified geography, for example if the user needs only Story County to be displayed, only Story County will be displayed. Crash Analyzer provides a variety of querying options such as selection of crashes in a rectangular area, an elliptical area or inside a user-defined polygon. Based on the user-selected options available within Crash Analyzer, Query Builder builds a machine-

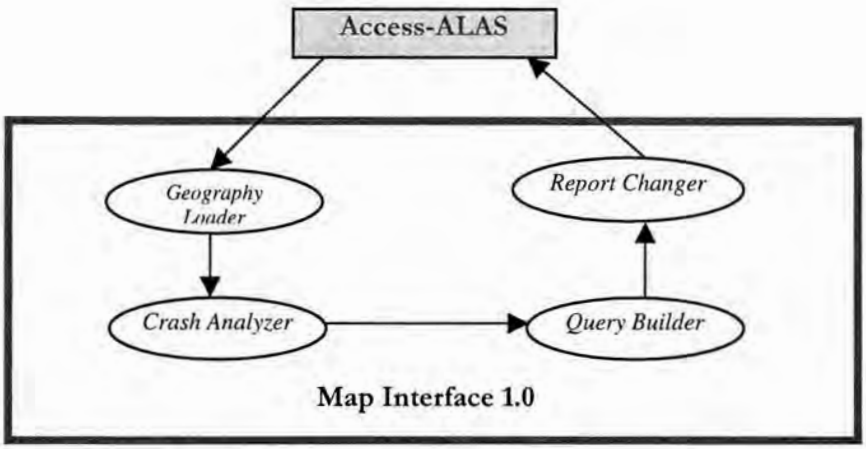


Figure 4.1. Flow of Control

understandable query string. The Report Changer modifies the Access-ALAS reports based on the results from Query Builder.

After a flowchart of the software system was developed, the Map Interface 1.0 source code was written in Microsoft Visual Basic 6.0 (VB). MapObjects 2.0, a Geographic Information System component library, was also used to support the software system mapping capabilities. VB was chosen because it is capable of implementing the UML specified software flowcharts.

Features of Map Interface 1.0

This section describes the features added to Access-ALAS by developing Map Interface 1.0. Each of the new features developed is explained in detail and is accompanied by screenshots from the software. As described in Chapter 1, Map Interface 1.0 provides the following new features to the Access-ALAS user:

- The ability to select crashes in an entire county;
- The ability to select crashes in an entire city and save the spatial query and/or data;

- The ability to select crashes in a user-defined area and save the spatial query and/or data;
- The ability to select crashes along a roadway between any two points and save the spatial query and/or data;
- The ability to select crashes between intersections with intersection cross-street names as the input, and save the spatial query and/or data; and
- The ability to select any individual crash or set of crashes.

Countywide Selection of Crashes

This feature allows the user to select all crashes in the displayed county. The user can select a county from a drop-down list that contains the names of all the counties in Iowa (see Figure 4.2). Selecting a county from the list selects all the crashes within the selected county. Map Interface 1.0 then displays all crashes in that county. The user has an option to use either Access-ALAS or Map Interface 1.0 to perform countywide crash analysis and generate reports.

Citywide Selection of Crashes

Map Interface 1.0 enables the selection of crashes by city name from a drop-down list (see Figure 4.2). Unlike Access-ALAS without Map Interface 1.0, the new version of Access-ALAS (with Map Interface 1.0) displays all the crashes in a city (see Figure 4.3). Map Interface 1.0 also allows the user to select a subset of those crashes in a city and enables the user to perform more detailed crash analysis.

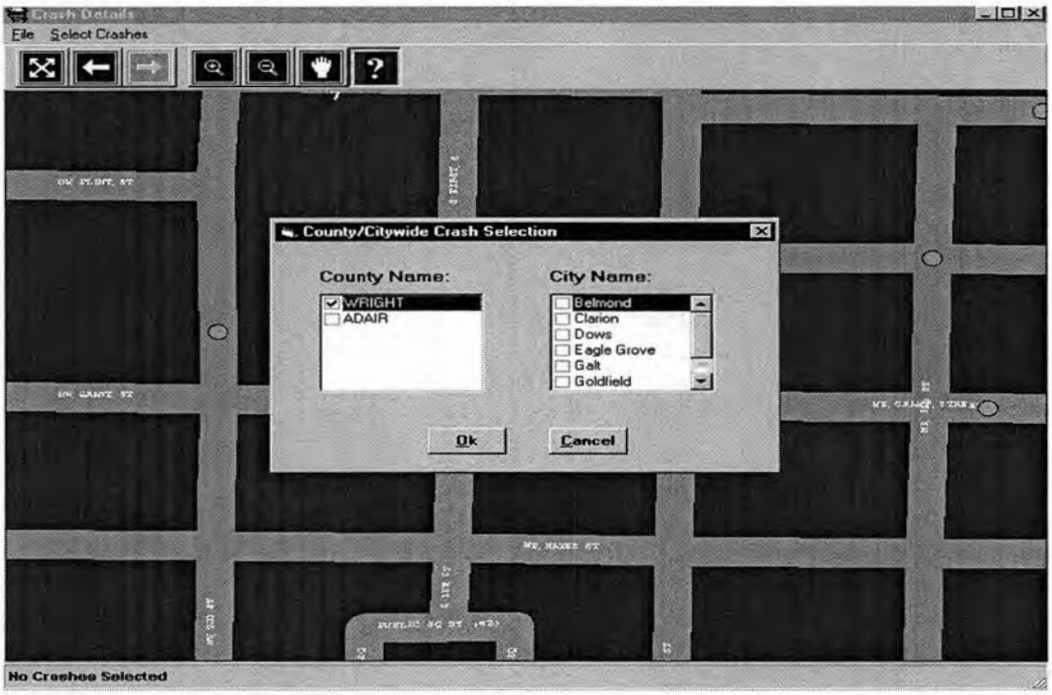


Figure 4.2. Countywide/Citywide Selection of Crashes

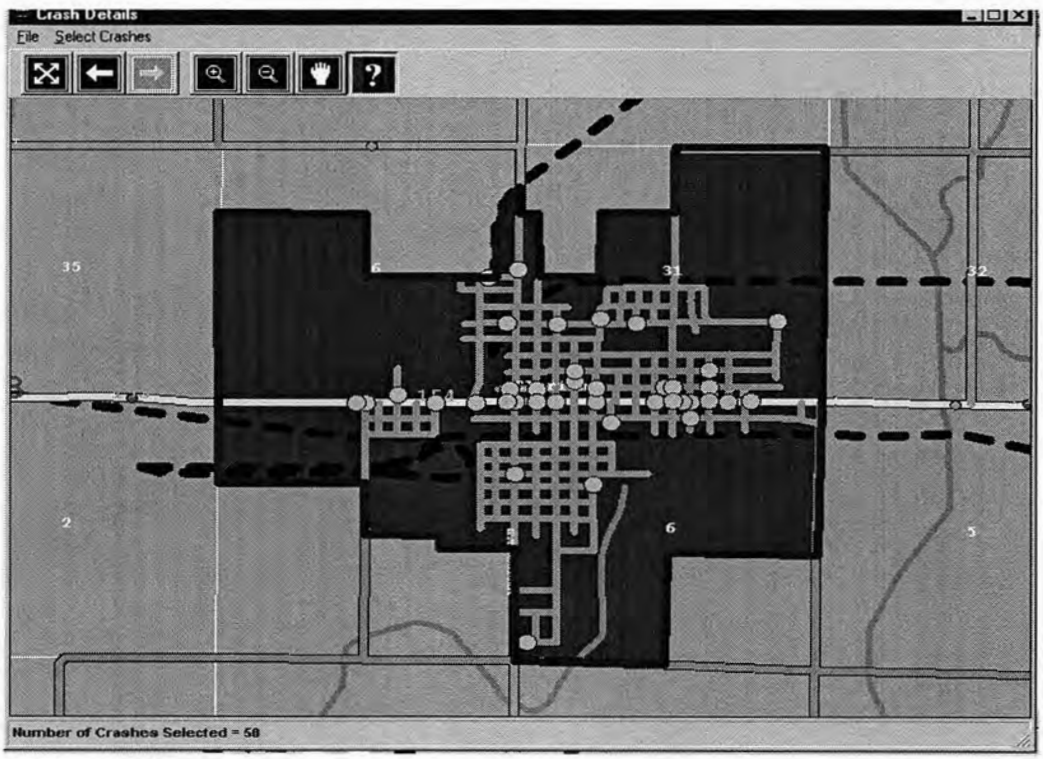


Figure 4.3. Crashes Selected in a City

Crash Selection in a User-Defined Area

With Map Interface 1.0 crashes can also be selected within a shape on the map. Shapes that can be drawn within the program include a rectangle, polygon, and ellipse. When any of these shapes is drawn, the crashes bounded by that shape are selected (selected crashes are displayed in different color and size than the unselected). The number of crashes selected is shown within the status bar in the lower left corner of the Map Interface (see Figure 4.4). The user can unselect the selected crashes by clicking on the "clear selection" option of the file menu. Figures 4.4, 4.5 and 4.6 show screenshots of rectangular, polygonal and elliptical selection of crashes. This feature provides the user with greater flexibility when performing detailed crash analyses to Access-ALAS without Map Interface 1.0. The new Access-ALAS/Map Interface 1.0 package provides the analyst with an infinite number of crash selection combinations. As already stated in Chapter 1, the ability to complete detailed crash analyses will help analysts make better safety improvement decisions (*6*). For example, in determining the adequacy of enforcement in a particular area (e.g., near an establishment that serves alcohol) the safety analyst can determine the number of crashes within the desired radius of the location under consideration (see Figure 4.6). Also, the shapes drawn by the user can be saved for future analysis.

Selection of Crashes on a Roadway between any Two Points on the Roadway

This feature of Map Interface 1.0 allows the user to select crashes between any two points on a roadway (see Figure 4.7). This function also enables the safety analyst to focus on specific transportation corridors when diagnosing problem areas. For example, a user might suggest different solutions to a highway safety problem if he/she selects crashes between two different sets of points on the roadway with Access-ALAS without Map Interface 1.0, which

is a major limitation. A current Access-ALAS user is restricted to selecting only nodes on a roadway, and this limits the user from performing a more specific crash analysis.

Selection of Crashes between Intersections

With the current Access-ALAS software package, crashes can only be selected between any two intersections by entering the node numbers of those intersections. The nodes to be entered are usually obtained from the node maps, which is a time-consuming process. To address this concern, Map Interface 1.0 incorporates a feature enabling the user to enter intersection cross-street names and spend more time determining safety improvement measures. This feature is illustrated in Figure 4.8.

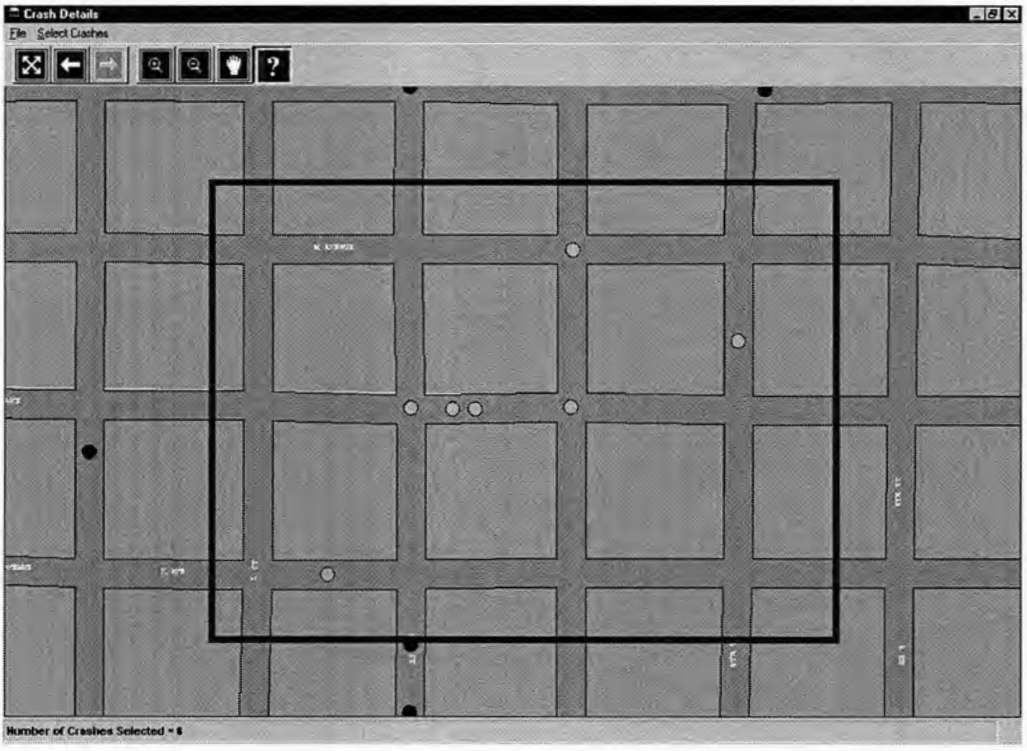


Figure 4.4. Rectangular Selection of Crashes

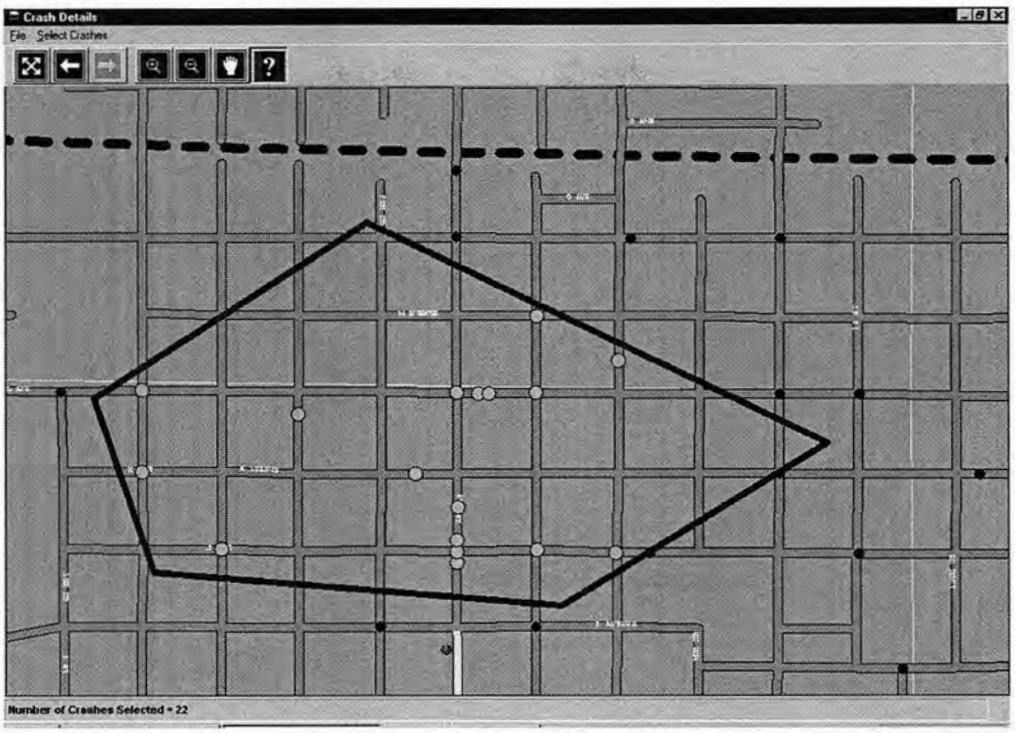


Figure 4.5. Polygonal Selection of Crashes

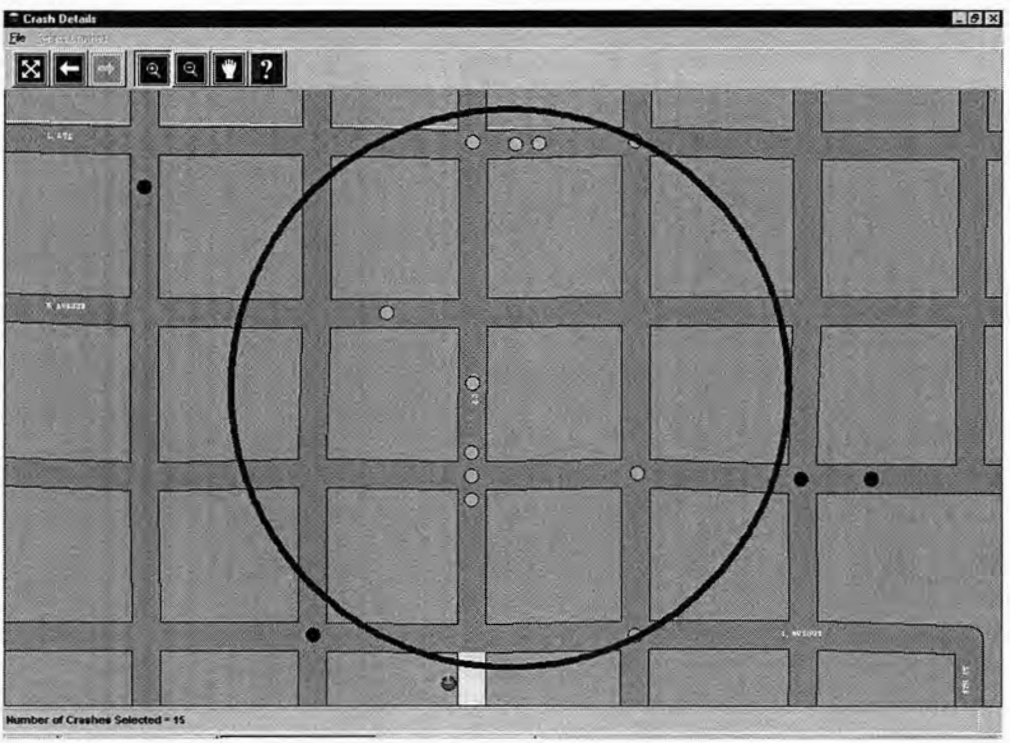


Figure 4.6. Elliptical Selection of Crashes

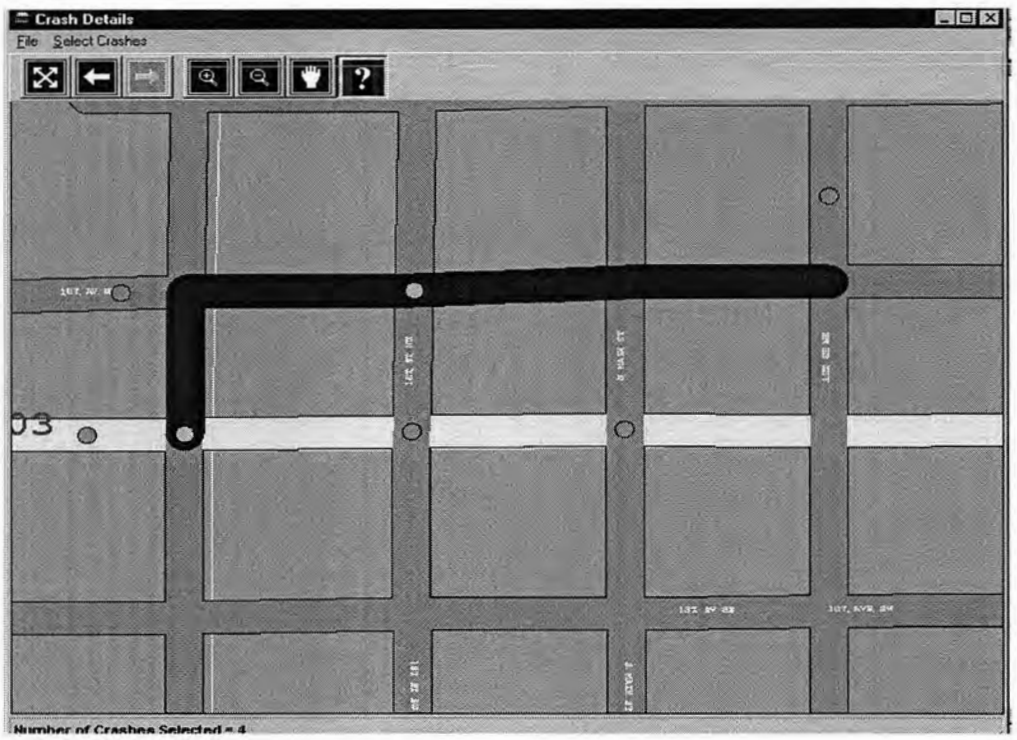


Figure 4.7 Selection of Crashes on a Route

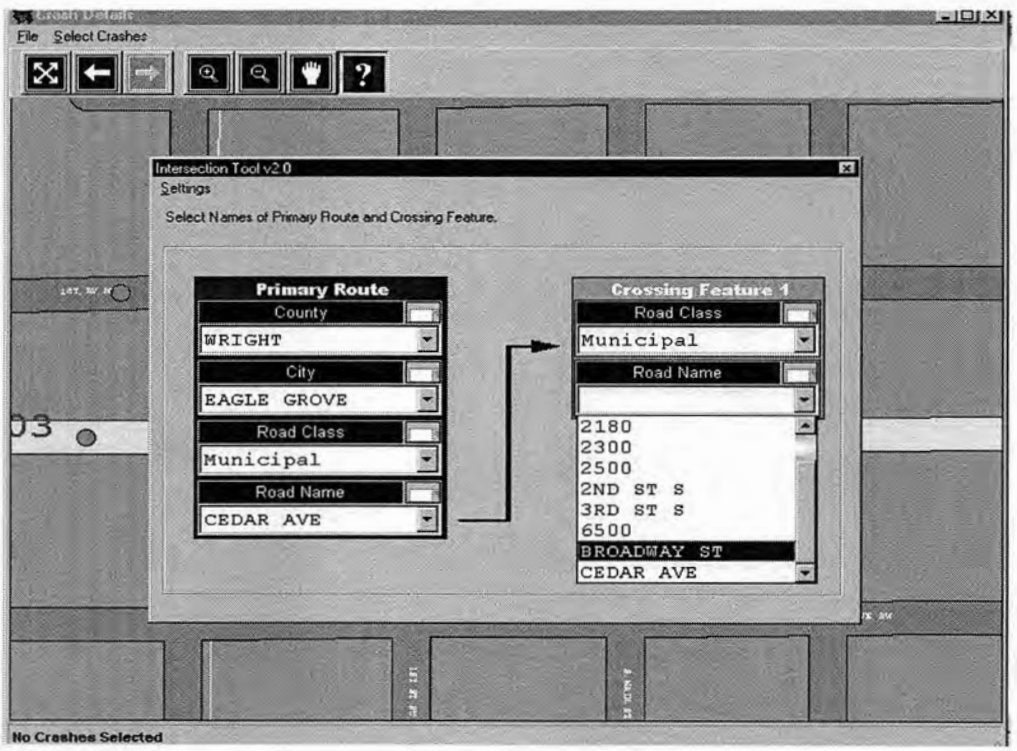


Figure 4.8. Selection of Crashes between Two Intersections

Selection of Individual Crash or Set of Crashes

Unlike the current Access-ALAS software, the user can also select an individual crash or a set of crashes using Map Interface 1.0. This feature enables the user to retrieve information about a specific crash/set of crashes facilitating site-specific crash analysis. The features of Map Interface 1.0 were designed to enable convenient, faster, reliable, more accurate and more efficient crash analysis. Therefore, a performance test was conducted to quantify how Map Interface 1.0 improved Access-ALAS. Four users were selected to run four identical queries on the same computer and analysis times of the current Access-ALAS software and the Access- ALAS with Map Interface 1.0 were noted.

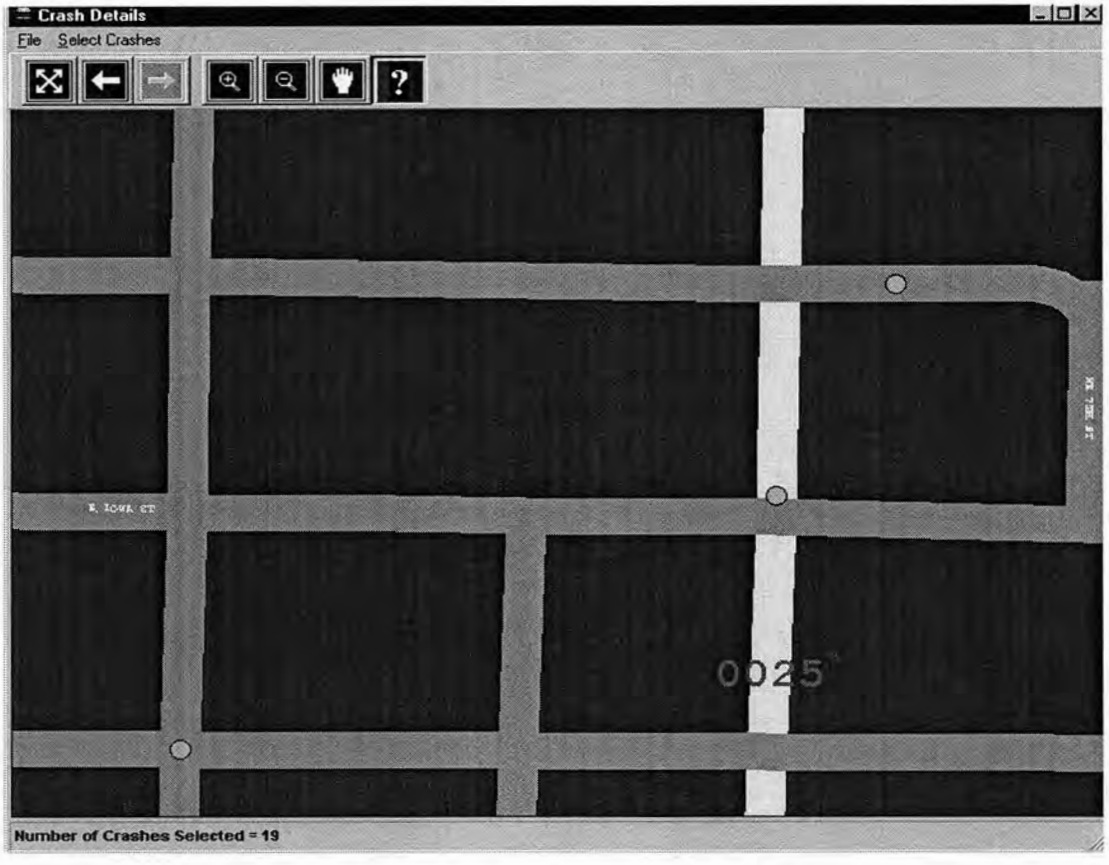


Figure 4.9. Selection of an Individual Crash or Set of Crashes

Performance Test and Results

The performance test of Map Interface 1.0 was designed to evaluate improvements in analysis speed for four different queries. These queries included:

- The selection of crashes between two adjacent intersections with no other nodes between them (Query 1).
- The selection of crashes between two intersections with other intersections/nodes in between them (Query 2).
- The selection of crashes at an intersection (Query 3).
- The selection of crashes within a half-mile radius around an intersection (Query 4).

The four test subjects selected were trained to run four queries using both Access-ALAS with and without Map Interface 1.0 (see Appendix C for the actual queries). Each of the four subjects performed the queries once on the same computer. The time to search the location of the crash, run the query, and generate reports with Access-ALAS with and without Map Interface 1.0 was recorded. The time savings and percent time savings were then calculated. Tables 4.1 through 4.4 show the results of these tests and calculations. The time needed to run these queries varied from subject to subject depending on their familiarity with the software.

Table 4.1. Performance Test Results for Selection of Crashes between Two Adjacent Intersections (Query 1)

	Time (in seconds) for Query 1 with		Time savings with Map Interface	
	Access-ALAS	Map Interface	Seconds	Percent
Subject 1	225	80	145	64
Subject 2	360	180	180	50
Subject 3	325	170	155	48
Subject 4	180	90	90	50
Average	**273**	**130**	**143**	***53***

Table 4.2. Performance Test Results for Selection of Crashes between Two Non-Adjacent Intersections (Query 2)

	Time (in seconds) for Query 2 with		Time savings with Map Interface	
	Access-ALAS	Map Interface	Seconds	Percent
Subject 1	345	165	180	52
Subject 2	360	180	180	50
Subject 3	410	190	220	54
Subject 4	220	74	146	66
Average	**334**	**152**	**182**	***56***

Table 4.3. Performance Test Results for Selection of Crashes at an Intersection (Query 3)

	Time (in seconds) for Query 3 with		Time savings with Map Interface	
	Access-ALAS	Map Interface	Seconds	Percent
Subject 1	130	25	105	81
Subject 2	360	120	240	67
Subject 3	596	60	536	90
Subject 4	78	58	20	26
Average	**291**	**66**	**225**	***66***

Table 4.4. Performance Test Results for Selection of Crashes within a Half-Mile Radius of an Intersection (Query 4)

	Time (in seconds) for Query 4 with		Time savings with Map Interface	
	Access-ALAS	Map Interface	Seconds	Percent
Subject 1	140	95	45	32
Subject 2	240	120	120	50
Subject 3	200	35	165	83
Subject 4	179	44	135	75
Average	**190**	**74**	**116**	***60***

Performance Test Result Evaluation

The results of the performance test indicate substantial time savings from the introduction of Map Interface 1.0 to Access-ALAS. A typical crash analysis performed by the Iowa DOT involves locating and querying crashes, generating reports, obtaining other information such as traffic volumes and speeds and then determining safety measures. According to an Iowa DOT safety analyst approximately 25 percent of typical crash analysis is spent in running Access-ALAS and generating reports (*23*). Therefore, for a total crash

analysis time of four hours, approximately one hour will be spent on running Access-ALAS. From the results of the performance test (see tables 4.1 to 4.4), a time saving of 60 percent was observed when similar analysis was done with Access-ALAS with Map Interface 1.0 compared to Access-ALAS without Map Interface 1.0. Thence, a 15 percent of the total crash analysis time can be saved with Map Interface 1.0. It was also observed from the test that there was no substantial difference in analyzing/query processing times indicating that a major portion of the time savings with Access-ALAS with Map Interface 1.0 were a result of additional crash selection features. Therefore, the user has more time to spend on determining solutions to potential safety problems. Figure 4.10 is a graphical representation of the performance test results.

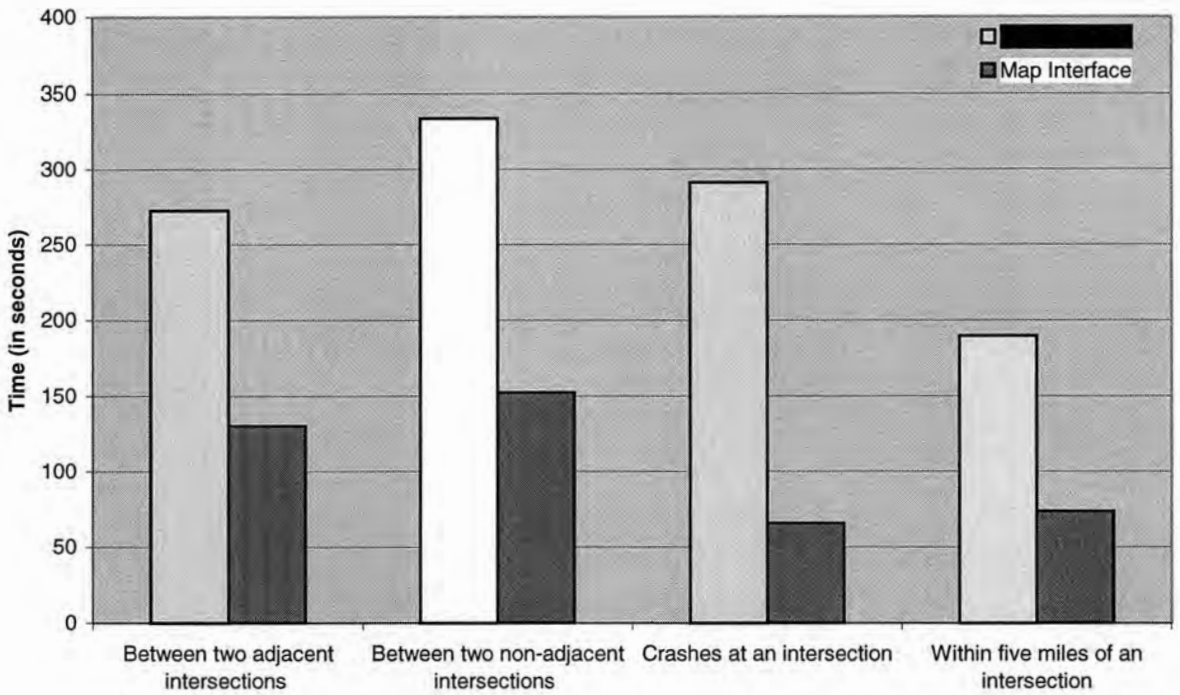


Figure 4.10. Graphical Representation of Map Interface Performance Test Results

In addition to the reduction in time needed to run the four queries the test subjects also expressed concerns about the flexibility of Access-ALAS with and without Map Interface 1.0. Test subject comments indicate that using a completely separate software (e.g., ArcView and ArcExplorer) or paper maps to obtain intersection node numbers is time-consuming and cumbersome. Also, the node maps (maps that display node locations) are not accurately digitized, meaning intersection node points on the map do not match with actual intersection locations on the ground thereby causing confusion and resulting in erroneous reports. Map Interface 1.0 is completely incorporated into Access-ALAS to eliminate the use of a different software or node maps to obtain node numbers. This software package thereby requires no training in the identification of node numbers.

CHAPTER 5. CONCLUSIONS AND RECOMMENDATIONS

The primary objective of this thesis was to design a software package to facilitate improved crash selection and allow more detailed crash analysis. An Access-ALAS user survey was done to determine user needs, and a performance test completed to determine the performance improvements by the addition of Map Interface 1.0 to Access-ALAS. During the course of this research, several conclusions were made. These conclusions, along with some recommendations, are presented in the following paragraphs.

Map Interface 1.0 was designed to work with Access-ALAS without eliminating its existing features. Therefore, users can use nodes to perform crash analysis of data from 1999 and before. For data after 1999, both the existing features of Access-ALAS and the new features provided by Map Interface 1.0. The features of the Access-ALAS with and without Map Interface 1.0 respectively are as follows:

Existing Features of Access-ALAS (without Map Interface 1.0)

- Node-based approach
- Crash Selection by Mileposts
- Countywide and Citywide crash selection
- Crash selection by route, intersection, and between intersections
- Crash data stored in an Access database
- Reports

Additional Features

- Mapping capabilities
- Digitized crash maps
- Rectangular, elliptical and polygonal selection of crashes

- Crash selection by intersection cross-street names
- Printable queries

Summary and Conclusions

Crash Data Analysis State of the Practice

Several states store crash data in databases and analyze crashes using a software that can query data. However, most states do not have a dedicated crash analysis software, which limits crash analysis to the extent that the querying software is capable of performing. States such as Idaho, Connecticut and Tennessee are planning to move towards a dedicated crash analysis system.

Access-ALAS User Survey Results

The Access-ALAS user survey determined that existing Access-ALAS users would like several features other than those required by the Iowa DOT as listed in Chapter 1. The survey results indicated that 80 percent of the users dislike all current Access-ALAS features, and 66 percent desire improved software. This could be due to the lack of proper interaction between the software users and designers.

Time and Cost Savings provided by Map Interface 1.0

The analysis of crashes with Map Interface1.0 resulted in substantial time savings (15 percent of the total crash analysis time) when compared to the time necessary to do the same task solely with Access-ALAS. The time savings provided by Map Interface 1.0 is primarily the result of eliminating node numbers to analyze crashes. With Map Interface 1.0, the user only has to point and click to select intersections or links between intersections to analyze specific crashes.

Over the past one-year, $40,000 (approximate) was spent in developing Map Interface 1.0. Assuming 250 users of Access-ALAS with Map Interface 1.0, 12 analyses per year on average, and an average time of 8 hours per analysis, a cost benefit of $180,000 (for an hourly rate of $50) is possible when Access-ALAS is used with Map Interface 1.0, a benefit-cost ratio of 4.5:1 in the first year. As the maintenance costs are usually lower than the development costs, the benefit-cost ratio would increase from the second year.

Specific Crash Analysis Impacts

The addition of Map Interface 1.0 to Access-ALAS allows the user to perform more specific (i.e., more detailed) crash analysis. In other words, the analysis can be done on several sets of crashes. For example, crashes on a roadway segment, the ends of which are not nodes, could not be analyzed with the existing Access-ALAS software package, but could be analyzed with Map Interface 1.0 added. Plus, for that same segment of roadway, the Map Interface 1.0/Access-ALAS combination allows the consideration of crashes along any roadway length, or at any point, or intersection.

In addition, for use of Access-ALAS with Map Interface 1.0 may improve the cost-effectiveness of funds allocated to improve safety along a particular roadway segment. An Access-ALAS user might suggest improvements for an entire segment (due to the limitations of the existing Access-ALAS), but with Map Interface 1.0 the same user could determine that only one portion of the segment needs safety improvements. Such an analysis could result in more cost-effective safety measures.

Training Requirements

Access-ALAS without Map Interface 1.0 required training. Approximately four hours were spent on each training session. In other words, Map Interface 1.0 requires little or no

training for users who are familiar with the Windows operating system. They can easily understand the menu-driven approach of Map Interface 1.0.

Extensibility of Map Interface 1.0

The major input requirements for the Map Interface are

- digitized maps; and
- crash database with unique crash identification numbers.

Several state DOTs maintain crash databases. Some of them include Connecticut, West Virginia, Virginia, Indiana, Idaho, Tennessee, Kentucky, Washington, Vermont, and North Carolina. Most of these states either already have or plan to have digitized crash and roadway maps. Therefore, Map Interface 1.0 can eventually be used in other states, but it would normally require modifications to database field names, the identification numbering system, and county and city listings. Robust and extensible software like Map Interface 1.0 allow it to be extended to other states and avoids designing similar software features more than once. This type of resource sharing among different states allows efficient use of safety funds.

Future of Access-ALAS

Access-ALAS with Map Interface 1.0 will be released with year 2000 crash data. In 2001, new data fields will be added to the crash database, and therefore crash reports will be modified for format and will not be recognized by the existing version of Access-ALAS with Map Interface 1.0. However, because Map Interface 1.0 is independent of the database and report formats, it will be compatible with the new format. Although Access-ALAS can be modified to read the new format, due to inadequate support to maintain it, the Iowa DOT plans to redevelop Access-ALAS using Visual Basic to make it compatible with the new

format to create VB-ALAS. By creating VB-ALAS, the Iowa DOT would be able to provide one user interface and a unique database for all their crash analysis tools, which will include VB-ALAS, GIS-ALAS, and a modified version of Intersection Magic (software that generates collision diagram sketches). Map Interface 1.0 will be used along with VB-ALAS for data years 2001 and beyond.

Recommendations

Additional study should be completed to evaluate the possibility of incorporating capability of assessing the benefit of mitigation measures (probabilistically), graphs, charts, and collision diagram sketches into crash analysis software developed as part of this research. Research to combine crash databases and analysis software to perform crash analysis automatically as soon as the database is populated should also be done. This capability will allow the users to generate reports using the most recent crash data.

Presently, in Iowa, a safety analyst either has to use crash analysis reports or query the crash database to rank high crash locations by crash rate, crash frequency and value loss. Crash analysis software capable of automating ranking locations would be a substantial improvement in high crash location determination. In addition, the ability to overlay crash locations on aerial photographs might help the user determine both feasible and infeasible safety improvements at a particular location. The crash database used by Access-ALAS with Map Interface 1.0 primarily contains data on vehicle and passengers involved in the crash, and the location of the crash. If Average Daily Traffic (ADT) information is entered into this database Map Interface 1.0 can develop vehicle miles traveled (VMT) for selected roadway links and/or analysis areas. Map Interface 1.0 could then determine crash rate for a given section or subset of roadway links.

A detailed analysis of user needs should also be performed before designing any crash analysis software. The Access-ALAS user survey for this research was performed after designing Map Interface 1.0, and it only helped in verifying the user needs with Iowa DOT's requirements. However, such a survey should be completed before designing any software. Similar to the Access-ALAS user survey, a survey of Map Interface 1.0 would help improve its features based on changing user requirements.

To keep the responses from the Access-ALAS user survey confidential the survey questionnaire did not include any provision for users to enter their personal information. However, in future to more accurately determine the characteristics/needs of the whole population from the sample responses a detailed survey including the users' personal information such as age, sex and job description should be performed. Also, for the performance test done as a part of this research the selected test subjects executed each query only once, but to more accurately determine the time savings, each query should be executed more than once.

The following list summarizes recommendations for future development of Access-ALAS with Map Interface 1.0.

- Add ability to probabilistically evaluate mitigation measures
- Add graphs and charts
- Add collision diagram sketches
- Add ability to dynamically interface with updated crash database
- Add ranking capabilities
- Add aerial photographs
- Add crash rate capability

- Redo user survey to assess/eliminate bias and determine specific user needs (after deployment of the first version of Map Interface 1.0)
- Expand performance test to strengthen the estimate of B/C ratio (to justify future development)

APPENDIX A. SURVEY INSTRUMENT

Access-ALAS User Survey

Please circle one choice for each item unless otherwise specified.

1) Please indicate whether you use Access-ALAS for any of the reasons listed below.

Reason	Yes	No
a) Improve Law Enforcement	1	2
b) Improve Engineering	1	2
c) Improve Emergency Response	1	2
d) Improve Education	1	2
e) Other (Please Specify)	1	2

2) How often do you use Access-ALAS?

1 - Once a week
2 - Once every two weeks
3 - Once a month
4 - Once every six months
5 - Once a year or less
6 - Other (please specify) _____

3) Have you used any other crash analysis software?

1 = Yes
2 = No → **skip to question 6.**

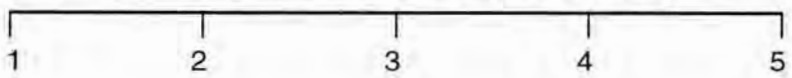
4) Do you prefer Access-ALAS software or another type of crash analysis software?

1 = Access-ALAS
2 = Other (please specify) _____

5) Which of the following descriptors of Access-ALAS do you agree with? (Circle all that apply)

1 = User-friendliness
2 = Powerful Querying capabilities
3 = Good Reporting Features
4 = Convenient or Useful Node-based approach

6) Indicate your opinion of the user-friendliness of Access-ALAS? Please use a scale where "1" is not user friendly at all and "5" is very user friendly



7)

a) Have you attended any DOT sponsored training sessions for Access-ALAS?

1 = Yes
2 = No → **skip to question 9**

b) Was the training helpful?

1 = Yes → **skip to question 9**
2 = No

c) How would you like to see the training improved?

_____

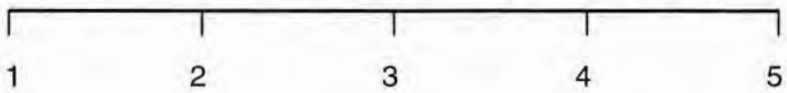
8) Please list the limitations (if any) of Access-ALAS software.

_____

_____

_____

9) How efficient is using node numbers to get information about crashes? Please use a scale where "1" means not efficient at all and "5" means very efficient.



10) Do you think you need improved software for crash analysis?

1 = Yes
2 = No

If yes, what kind of improvements would you like to have in a future version?

_____

_____

Thank you very much for filling the survey. We appreciate your time. Please return the completed survey to Iowa State University (use the business reply envelope enclosed).

APPENDIX B. VISUAL BASIC CODE

Project: prjMapInterfaceTest

Form: frmCountyorCitySelection

```
Option Explicit

Public Event cmdOkClick()
Public Event cmdCancelClick()

Private Sub cmdCancel_Click()

    RaiseEvent cmdCancelClick

End Sub

Private Sub cmdOk_Click()

    RaiseEvent cmdOkClick

End Sub

Private Sub lstCountyNames_Click()

    Dim TempCityLayer As New MapLayer
    Dim TempCityDataConnection As New MapObjects2.DataConnection
    TempCityDataConnection.Database = App.Path & "\ShapeData"
    TempCityDataConnection.Connect
    SetTempCityLayer.GeoDataset = TempCityDataConnection.FindGeoDataset("PR_Corp")

    Dim morsTempCityLayer As MapObjects2.Recordset
    Set  morsTempCityLayer  =  TempCityLayer.SearchExpression("Co_Name  =  '"  &
lstCountyNames.Text & "'")

    lstCityNames.Clear
    Do While Not morsTempCityLayer.EOF

        lstCityNames.AddItem morsTempCityLayer.Fields("Place_Name").Value
        morsTempCityLayer.MoveNext

    Loop

End Sub

Private Sub Form_Load()

    Dim TempCountyLayer As New MapLayer
    Dim TempDataConnection As New MapObjects2.DataConnection
    TempDataConnection.Database = App.Path & "\ShapeData"
    TempDataConnection.Connect
    Set TempCountyLayer.GeoDataset = TempDataConnection.FindGeoDataset("PR_Bord")

    Dim morsTempCountyLayer As MapObjects2.Recordset
    Set morsTempCountyLayer = TempCountyLayer.Records

    Do While Not morsTempCountyLayer.EOF

        lstCountyNames.AddItem morsTempCountyLayer.Fields("Co_Name").Value
```

```
        morsTempCountyLayer.MoveNext

    Loop

   lstCountyNames.Text = lstCountyNames.List(0)

   Dim TempCityLayer As New MapLayer
   Dim TempCityDataConnection As New MapObjects2.DataConnection
   TempCityDataConnection.Database = App.Path & "\ShapeData"
   TempCityDataConnection.Connect
   Set TempCityLayer.GeoDataset = TempCityDataConnection.FindGeoDataset("PR_Corp")

    Dim morsTempCityLayer As MapObjects2.Recordset
    Set  morsTempCityLayer  =  TempCityLayer.SearchExpression("Co_Name  =  '"  &
lstCountyNames.Text & "'")

    Do While Not morsTempCityLayer.EOF

        lstCityNames.AddItem morsTempCityLayer.Fields("Place_Name").Value
        morsTempCityLayer.MoveNext

    Loop

End Sub
```

Form: frmGUI

```
Option Explicit

Private WithEvents myMapTester As MapCommander.MapController
Private WithEvents myManager As SelectTools.clsManager
Private WithEvents mySaveQueryForm As frmSaveQuery
Private WithEvents myQueryNameForm As frmQueryName
Private WithEvents mySettingsForm As frmSettings
Private WithEvents myPreviousQueryForm As frmPreviousQuery
Private myProgressForm As frmProgress
Private mySelectCountyForm As frmSelectCounty
Private WithEvents myCountyOrCitySelectionForm As frmCountyOrCitySelection
Private LayerColBuilder As LayerControls.clsMapLayerBuilder
Private LayerColSaver As LayerControls.clsMapLayerSaver
Private objAdjustLayer As LayerControls.clsAdjustMapLayer
Private objLayerControl As LayerControls.clsLayerControl
Private LayerLoader As LayerControls.clsLayerLoader
Private myGeographyLoader As GeographyLoader.clsGeographyLoader
Private myTripleCollectionBuilder As RecordsetUtilities.clsTripleColBuilder
Private myGetMinBndRect As FlexibleMap.clsGetMinBndRect

Private strDatabaseLocation As String
Private rsCrashRecords As MapObjects2.Recordset
Private objShapeDrawn As Object
Private ptsCollection As MapObjects2.Points

Private dbMaster As Database
Private qdMaster As QueryDef
Private rsMaster As Recordset

Private TempLayer As New MapLayer
```

```
Private TempDataConnection As New MapObjects2.DataConnection

Private Sub childSelectionTools_Click(index As Integer)

    myManager.SelectionTools.Mode = index

    Dim X As Integer

    For X = 1 To childselectiontools.Count

        If X = index And index <> 6 Then

            childselectiontools(X).Checked = True

        Else

            childselectiontools(X).Checked = False

        End If

    Next

    If index = 1 Then

        mnuChildPopup.Item(6).Enabled = True

    Else

        mnuChildPopup.Item(6).Enabled = False

    End If

    If index = 4 Then

        myManager.SelectionTools.BuildIntersectionRecordset

    ElseIf index = 6 Then

        objLayerControl.LoadForm

    ElseIf index = 7 Then

        Set myCountyOrCitySelectionForm = New frmCountyOrCitySelection
        Load myCountyOrCitySelectionForm
        myCountyOrCitySelectionForm.Show 1, Me

    End If

End Sub

Private Sub cmdChangeSettings_Click()

    mySettingsForm.Show 1, Me

End Sub

'Private Sub cmdLayerControl_Click()
```

```
'
'    objLayerControl.LoadForm
'
'End Sub

Private Sub cmdPreviousQuery_Click()

    Me.MousePointer = 11
    Me.MousePointer = 0
    Load frmPreviousQuery
    frmPreviousQuery.Show 1, Me
    Me.Visible = True

End Sub

Private Sub cmdRestorelayer_Click()

    Dim ResetDefaultLayer As New clsResetLayerDefaults

    'Restore Database

    If                          ResetDefaultLayer.GetDefault(strDatabaseLocation,
LayerColBuilder.MapLayerCol) = True Then

        'Rebuild Collection
        Call LayerColBuilder.GetCollectionData(strDatabaseLocation)

        'Display Restored MapLayers
        Set LayerLoader.MainMap = Me.Map1
        Call    LayerLoader.LoadLayer(LayerColBuilder.MapLayerCol,    App.Path    &
"\ShapeData")

    End If

    Set ResetDefaultLayer = Nothing

End Sub

Private Sub cmdSaveLayer_Click()

    Call  LayerColSaver.SaveCollectionData(LayerColBuilder.MapLayerCol,  "C:\Burd's
Documents")

End Sub

Private Sub PrintView()

    On Error GoTo PrintErrorHandler

    Dim BeginPage, EndPage, NumCopies, PageOrient, I
    CommonDialog1.CancelError = False
    BeginPage = CommonDialog1.FromPage
    EndPage = CommonDialog1.ToPage
    NumCopies = CommonDialog1.Copies
    PageOrient = CommonDialog1.Orientation
    For I = 1 To NumCopies
      Map1.PrintMap "Mymap", "", True
```

```
    Next I

    Exit Sub

PrintErrorHandler:

    Err.Raise 1, , "Error Printing Map - Check Printer Settings"

End Sub

Private Sub Form_Load()

    frmSplashScreen.Show 1

    DoEvents

    Set myProgressForm = New frmProgress
    myProgressForm.Show

    myProgressForm.MousePointer = 11
    DoEvents

    Set myManager = New SelectTools.clsManager

    myManager.SelectionTools.GeographyChanged = False

    myProgressForm.ProgressBar1.Value = myProgressForm.ProgressBar1.Max / 5

    Call GeoLoad

     Set dbMaster = OpenDatabase(App.Path & "\Master1.mdb")
'      Set qdMaster = dbMaster.CreateQueryDef("", "select case_num from queries
where query_name= '" + frmPreviousQuery.cmbQuery.Text + "'")
'    Set rsMaster = qdMaster.OpenRecordset()

    Set qdMaster = dbMaster.CreateQueryDef("", "select Value from Settings where
Setting_Name = 'GeographyChanged'")
    Set rsMaster = qdMaster.OpenRecordset()

    myManager.SelectionTools.GeographyChanged = rsMaster.Fields(0).Value

    Set mySettingsForm = New frmSettings

    Set myMapTester = New MapCommander.MapController
    Set myMapTester.MainMap = Me.Map1

    'Menu Settings
    topSelectionTools.Enabled = False

    'Manager class settings

    myManager.DatabasePath = App.Path
    myManager.DatabaseName = "\Master1.mdb"
    myManager.SelectionTools.Mode = 2
    myManager.SelectionTools.CrashLayerName = "Crashes"

    Set myManager.CrashMap = Map1
```

```
    myManager.Enabled = False

    myProgressForm.ProgressBar1.Value = myProgressForm.ProgressBar1.Max

    strDatabaseLocation = App.Path

    'LayerControl Settings
    Set LayerLoader = New LayerControls.clsLayerLoader
    Set LayerColBuilder = New LayerControls.clsMapLayerBuilder
    Set LayerColSaver = New LayerControls.clsMapLayerSaver
    Set objAdjustLayer = New LayerControls.clsAdjustMapLayer
    Set objLayerControl = New LayerControls.clsLayerControl
    strDatabaseLocation = App.Path
    Call LayerColBuilder.GetCollectionData(strDatabaseLocation)
    Set LayerLoader.MainMap = Me.Map1
    Call      LayerLoader.LoadLayer(LayerColBuilder.MapLayerCol,      App.Path      &
"\ShapeData")
    Set objAdjustLayer.MainMap = Me.Map1
    Set objAdjustLayer.MapLayerCol = LayerColBuilder.MapLayerCol
    Set objLayerControl.MainMap = Me.Map1
    Set objLayerControl.LayerCollection = LayerColBuilder.MapLayerCol

    'set snapmanager properties
    myManager.SelectionTools.SnapLayerName = "Roads"
    myManager.SelectionTools.SearchDistance                                        =
mySettingsForm.slrSearchDistance.Value
    myManager.SelectionTools.SnapDistance = 20

    Unload myProgressForm
    Set myProgressForm = Nothing
    DoEvents

    'Adjusting the Full Extent of the map
    Dim mylayer As New MapLayer
    Dim ExtentRectangle As New MapObjects2.Rectangle

    Set mylayer = Me.Map1.Layers("County Border")
    Set myGetMinBndRect = New FlexibleMap.clsGetMinBndRect
    Set                          ExtentRectangle                                   =
myGetMinBndRect.GetMinimumBoundingRectangle(mylayer.Records)
    ExtentRectangle.ScaleRectangle 1.2
    Set Me.Map1.FullExtent = ExtentRectangle
    Set Me.Map1.Extent = Me.Map1.FullExtent
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

    Set mySaveQueryForm = New frmSaveQuery
    Set myQueryNameForm = New frmQueryName
    Set mySettingsForm = New frmSettings

    StatusBar1.Panels.Item(1).Width = 15000
    StatusBar1.Panels.Item(1).Text = "No Crashes Selected"

    Set ExtentRectangle = Nothing
    Set mylayer = Nothing

End Sub
```

```
Private Sub Form_Resize()

    On Error GoTo ErrorHandler

'-------------------------------------------------
'                   Variable Declarations
'-------------------------------------------------

Dim Border As Double, SideBorder As Double
Dim TopBorder As Double, StatusBarHeight As Double
Dim AttributeDisplayWidth, AttributeDisplayHeight As Double

'-------------------------------------------------
'                   Minimum Width Settings
'-------------------------------------------------

Border = 8
SideBorder = 8
TopBorder = 600
AttributeDisplayHeight = 0
StatusBarHeight = 380
AttributeDisplayWidth = 0

'-------------------------------------------------
'                   Instructions
'-------------------------------------------------

Map1.Top = TopBorder
Map1.Left = SideBorder
'Picture1.Top = TopBorder + 200
'Picture1.Left = SideBorder + 200

'Move GPS Control
'GPSConsole.Left = SideBorder + 200
'GPSConsole.Top = TopBorder + (MapWindow.Height - GPSConsole.Height - 200)

If ScaleHeight > TopBorder + Border + StatusBarHeight Then
        Map1.Height = ScaleHeight - TopBorder - Border - StatusBarHeight

End If

If ScaleWidth > (Border * 2) + AttributeDisplayWidth Then
        Map1.Width = ScaleWidth - (Border * 2) - AttributeDisplayWidth
End If

'*************************************************
'          Error Handling Component
'*************************************************

Exit Sub

ErrorHandler:

Unload Me
 '************************************************
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)

    Set myMapTester = Nothing
    Set myManager = Nothing
    Set mySaveQueryForm = Nothing
    Set myQueryNameForm = Nothing
    Set mySettingsForm = Nothing
    Set myPreviousQueryForm = Nothing
    Set myProgressForm = Nothing
    Set myCountyOrCitySelectionForm = Nothing
    Set LayerColBuilder = Nothing
    Set LayerColSaver = Nothing
    Set objAdjustLayer = Nothing
    Set objLayerControl = Nothing
    Set LayerLoader = Nothing
    Set myGeographyLoader = Nothing
    Set myTripleCollectionBuilder = Nothing
    Set myGetMinBndRect = Nothing
    Set rsCrashRecords = Nothing
    Set objShapeDrawn = Nothing
    Set ptsCollection = Nothing
    Set dbMaster = Nothing
    Set qdMaster = Nothing
    Set rsMaster = Nothing
    Set TempLayer = Nothing
    Set TempDataConnection = Nothing


End Sub

Private Sub Map1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)

    If      SmartMapToolBar.Buttons("ZoomIn").Value      =      tbrPressed      Or
SmartMapToolBar.Buttons("ZoomOut").Value          =          tbrPressed          Or
SmartMapToolBar.Buttons("PanMap").Value = tbrPressed Then

        myManager.Enabled = False

    End If

End Sub

Private Sub Map1_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)

    If Button = 2 Then

        PopupMenu mnuParentPopup

    End If

End Sub

Private Sub mnuChildPopup_Click(index As Integer)

    Select Case index
```

```
    Case 1  'Run Query

        Me.MousePointer = 11
        Set myManager.CollectionBuilder.MapRecordset = rsCrashRecords
        Set                    myManager.SQLQuery.CaseCollection                =
myManager.CollectionBuilder.CaseCollection
        Me.MousePointer = 0

        mySaveQueryForm.Show 1

    Case 2  'Run Old Query

        Me.MousePointer = 11
        Me.MousePointer = 0
        Set myPreviousQueryForm = New frmPreviousQuery

        myPreviousQueryForm.Show 1, Me

'        Me.Visible = True

    Case 3  'Print Map

        PrintView

    Case 5  'Layer Control

        objLayerControl.LoadForm

    Case 6  'Change Settings

        mySettingsForm.Show 1

    Case 8

        myManager.SelectionTools.ClearSelection

    Case 9  'Exit Map Interface

        Unload Me
        DoEvents

    End Select

End Sub

Private Sub myCountyOrCitySelectionForm_cmdCancelClick()

    Unload myCountyOrCitySelectionForm

End Sub

Private Sub myCountyOrCitySelectionForm_cmdOkClick()

    Dim qdCountyNameNumberCrossReference As DAO.QueryDef
    Dim rsCountyNameNumberCrossReference As DAO.Recordset
```

```
    If myCountyOrCitySelectionForm.lstCityNames.Text = "" Then

        Set qdCountyNameNumberCrossReference = dbMaster.CreateQueryDef("", "select
County_Number from CountyNameNumberCrossReference where county_name = '" &
myCountyOrCitySelectionForm.lstCountyNames.Text & "'")
        Set                    rsCountyNameNumberCrossReference                    =
qdCountyNameNumberCrossReference.OpenRecordset()
        Debug.Print rsCountyNameNumberCrossReference.Fields(0).Value
        myManager.SelectionTools.BuildCountyRecordset
CInt(rsCountyNameNumberCrossReference.Fields(0).Value)

    Else

        myManager.SelectionTools.BuildCityrecordset
myCountyOrCitySelectionForm.lstCityNames.Text

    End If

    Unload myCountyOrCitySelectionForm

End Sub

'Private Sub menuSelectionTools_Click(Index As Integer)
'
'    Select Case Index
'
'    Case 1
'
'        myManager.SelectionTools.Mode = "Point"
'
'    Case 2
'
'        myManager.SelectionTools.Mode = "Line"
'
'    Case 3
'
'        myManager.SelectionTools.Mode = "Rectangle"
'
'    Case 4
'
'        myManager.SelectionTools.Mode = "Polygon"
'
'    Case 5
'
'        myManager.SelectionTools.Mode = "Ellipse"
'
'    Case 6
'
'        myManager.SelectionTools.Mode = "SQL"
'
'    End Select
'
'End Sub

Private Sub myManager_CrashesSelected(shapedrawn As Object, CrashRecords As
MapObjects2.Recordset)
```

```
    If Not CrashRecords Is Nothing Then

        If CrashRecords.Count <> 0 Then

            Dim intRecordCounter As Integer

            CrashRecords.MoveFirst
            While Not CrashRecords.EOF

                intRecordCounter = intRecordCounter + 1
                CrashRecords.MoveNext

            Wend

            mnuChildPopup.Item(1).Enabled = True
            Set rsCrashRecords = CrashRecords
            Set objShapeDrawn = shapedrawn
            StatusBar1.Panels.Item(1).Text = "Number of Crashes Selected = " &
intRecordCounter
            mnuChildPopup.Item(8).Enabled = True

        Else

            mnuChildPopup.Item(1).Enabled = False
            StatusBar1.Panels.Item(1).Text = "No Crashes Selected"
            mnuChildPopup.Item(8).Enabled = False

        End If

    Else

        mnuChildPopup.Item(8).Enabled = False
        StatusBar1.Panels.Item(1).Text = "No Crashes Selected"

    End If

End Sub

Private Sub myManager_SegmentPointsCollectionPassed(PointsCollection As
MapObjects2.Points)

    Set ptsCollection = PointsCollection

End Sub

Private Sub myMapTester_ValidStates(StatusBlock As MapCommander.States)

    SmartMapToolBar.Buttons("ZoomIn").Enabled = StatusBlock.ZoomIn
    SmartMapToolBar.Buttons("ZoomOut").Enabled = StatusBlock.ZoomOut
    SmartMapToolBar.Buttons("PanMap").Enabled = StatusBlock.PanMap
    SmartMapToolBar.Buttons("FitAll").Enabled = StatusBlock.ViewAll
    SmartMapToolBar.Buttons("Previous").Enabled = StatusBlock.Back
    SmartMapToolBar.Buttons("Next").Enabled = StatusBlock.Forward

    Select Case myMapTester.Mode

        Case 1
```

```
        SmartMapToolBar.Buttons("ZoomIn").Value = tbrPressed

        Case 2

        SmartMapToolBar.Buttons("ZoomOut").Value = tbrPressed

        Case 3

        SmartMapToolBar.Buttons("PanMap").Value = tbrPressed

    End Select

End Sub

Private Sub myPreviousQueryForm_cmdCancelClick()

    myPreviousQueryForm.Hide
    Set myPreviousQueryForm = Nothing

End Sub

Private Sub myPreviousQueryForm_cmdQueryClick()

    Set qdMaster = dbMaster.CreateQueryDef("", "select Shape_Index from Queries
where query_name= '" + myPreviousQueryForm.cmbQuery.Text + "'")
    Set rsMaster = qdMaster.OpenRecordset()

    If myPreviousQueryForm.cmbQuery.Text <> "" Then

        myManager.SelectionTools.QueryName = myPreviousQueryForm.cmbQuery.Text
        myManager.SelectionTools.PreviousQueryEnabled = True
        myManager.SelectionTools.LoadShape      myPreviousQueryForm.cmbQuery.Text,
rsMaster.Fields(0).Value
        myPreviousQueryForm.Hide
        Set myPreviousQueryForm = Nothing

    Else

        MsgBox "Please Select a Query", , "MapInterface 1.0"

    End If

    myManager.SelectionTools.PreviousQueryEnabled = False

End Sub

Private Sub myQueryNameForm_cmdCancelClick()

    Unload myQueryNameForm

End Sub

Private Sub myQueryNameForm_cmdOkClick()

    Set TempLayer.GeoDataset = TempDataConnection.FindGeoDataset("PR_BORD")
```

```
    Dim bolQueryNameAlreadyExists As Boolean
    Dim qdQueryName As DAO.QueryDef
    Dim rsQueryName As DAO.Recordset

    Set qdQueryName = dbMaster.CreateQueryDef("", "Select Distinct(Query_Name)
from Queries")
    Set rsQueryName = qdQueryName.OpenRecordset()

    bolQueryNameAlreadyExists = False

    rsQueryName.MoveFirst

    'Check for DUPLICATE query names
    While Not rsQueryName.EOF

        If LCase(myQueryNameForm.Text1.Text) = LCase(rsQueryName.Fields(0).Value)
Then

            bolQueryNameAlreadyExists = True

        End If

        rsQueryName.MoveNext

    Wend

        If bolQueryNameAlreadyExists = True Then

            MsgBox "The query name already exists, Please select an other query
name", , "Duplicate Query Name"

        Else

            Select Case myManager.SelectionTools.Mode

            Case 1

                myManager.StoreQuery.StorePoint         myQueryNameForm.Text1.Text,
objShapeDrawn.Center, Abs(objShapeDrawn.Center.X - objShapeDrawn.Left)
                myManager.StoreQuery.StoreCaseNumbers   myQueryNameForm.Text1.Text,
rsCrashRecords, 1

            Case 2

                myManager.StoreQuery.StoreRectangle     myQueryNameForm.Text1.Text,
objShapeDrawn
                myManager.StoreQuery.StoreCaseNumbers   myQueryNameForm.Text1.Text,
rsCrashRecords, 2

            Case 3

                myManager.StoreQuery.StorePolygon       myQueryNameForm.Text1.Text,
objShapeDrawn
                myManager.StoreQuery.StoreCaseNumbers   myQueryNameForm.Text1.Text,
rsCrashRecords, 3

            Case 4
```

```
                myManager.StoreQuery.StorePoint         myQueryNameForm.Text1.Text,
objShapeDrawn.Center, Abs(objShapeDrawn.Center.X - objShapeDrawn.Left)
                myManager.StoreQuery.StoreCaseNumbers   myQueryNameForm.Text1.Text,
rsCrashRecords, 1

            Case 5

                myManager.StoreQuery.StoreSegments      myQueryNameForm.Text1.Text,
ptsCollection
                myManager.StoreQuery.StoreCaseNumbers   myQueryNameForm.Text1.Text,
rsCrashRecords, 5

            End Select

            myManager.StoreQuery.StoreLoadedCountyNumbers
myQueryNameForm.Text1.Text, TempLayer.Records
            myQueryNameForm.Hide
            MsgBox "Query Saved", , "Save Results"
            DoEvents

        End If

End Sub

Private Sub mySaveQueryForm_cmdNoClick()

    mySaveQueryForm.Hide

End Sub

Private Sub mySaveQueryForm_cmdYesClick()

    mySaveQueryForm.Hide
    myQueryNameForm.Show 1

End Sub

Private Sub mySettingsForm_SearchDistanceSelected(SearchDistance As Integer)

    myManager.SelectionTools.SearchDistance = SearchDistance

End Sub

Private Sub SmartMapToolBar_ButtonClick(ByVal Button As Comctl1Lib.Button)

Select Case Button.Key

    Case "ZoomIn"

        myMapTester.ZoomIn
        myManager.Enabled = False
'        myManager.SelectionTools.SnapManager.Enabled = False

        topSelectionTools.Enabled = False
        mnuChildPopup.Item(6).Enabled = False
```

```
        'Set a Fancy Mouse Pointer
        Map1.MousePointer = moZoomIn

    Case "ZoomOut"

        myMapTester.ZoomOut
        myManager.Enabled = False
        ' myManager.SelectionTools.SnapManager.Enabled = False

        topSelectionTools.Enabled = False
        mnuChildPopup.Item(6).Enabled = False

        'Set a Fancy Mouse Pointer
        Map1.MousePointer = moZoomOut

    Case "PanMap"

        myMapTester.Pan
        myManager.Enabled = False
        ' myManager.SelectionTools.SnapManager.Enabled = False

        topSelectionTools.Enabled = False
        mnuChildPopup.Item(6).Enabled = False

        'Set a Fancy Mouse Pointer
        Map1.MousePointer = moPan

    Case "FitAll"

        myMapTester.ViewAll
        'myManager.Enabled = False
        'myManager.SelectionTools.SnapManager.Enabled = False

    Case "Select"

        myMapTester.ExternalFunction
        myManager.Enabled = True
        ' myManager.SelectionTools.SnapManager.Enabled = True

        topSelectionTools.Enabled = True
        mnuChildPopup.Item(6).Enabled = True

        'Set a Fancy Mouse Pointer
        Map1.MousePointer = moArrowQuestion

    Case "Previous"

        myMapTester.Backward

    Case "Next"

        myMapTester.Forward

    Case "PointSelect"

        myManager.SelectionTools.ClearSelection
        myMapTester.ExternalFunction
```

```
        myManager.Enabled = False

End Select

End Sub

Private Sub Exitcmd_Click()

    Unload Me
    DoEvents

End Sub

Private Sub GeoLoad()

    DoEvents

    Dim MasterDatabase As Database
    Set MasterDatabase = OpenDatabase(App.Path & "\Master1.mdb")

    Dim RS1QueryDef As QueryDef
    Dim RS1Recordset As Recordset

    Dim ALASDataBase As Database
    Dim ALASQueryDef As QueryDef
    Dim ALASRecordSet As Recordset

    Set RS1QueryDef = MasterDatabase.CreateQueryDef("", "select Value from
settings where Setting_Name= 'ALASDatabasePath'")
    Set RS1Recordset = RS1QueryDef.OpenRecordset()

    Set ALASDataBase = OpenDatabase(RS1Recordset.Fields(0))
    Set ALASQueryDef = ALASDataBase.CreateQueryDef("", "Select Distinct County_I
from tblaccidents")
    Set ALASRecordSet = ALASQueryDef.OpenRecordset()

    'User is allowed to select one county from those in the ALAS Subset
    Set mySelectCountyForm = New frmSelectCounty

    ALASRecordSet.MoveFirst

    While Not ALASRecordSet.EOF

        mySelectCountyForm.List1.AddItem ALASRecordSet.Fields(0).Value
        ALASRecordSet.MoveNext

    Wend

'    mySelectCountyForm.Show

    myProgressForm.ProgressBar1.Value = myProgressForm.ProgressBar1.Max / 4

    Set myTripleCollectionBuilder = New RecordsetUtilities.clsTripleColBuilder

    myTripleCollectionBuilder.Field1 = "county_i"
    myTripleCollectionBuilder.Field2 = "County"
```

```
    TempDataConnection.Database = App.Path & "\ShapeData"
    TempDataConnection.Connect
    Set TempLayer.GeoDataset = TempDataConnection.FindGeoDataset("PR_Crash")

    Set myTripleCollectionBuilder.Recordset1 = ALASRecordSet
    Set myTripleCollectionBuilder.Recordset2 = TempLayer.Records

    Set TempLayer = Nothing

    myProgressForm.ProgressBar1.Value = myProgressForm.ProgressBar1.Max / 3

    Dim SourcePathQueryDef As QueryDef
    Dim SourcePathRecordset As Recordset

    If (myTripleCollectionBuilder.colInANotInB.Count > 0) Or
(myTripleCollectionBuilder.colBothInAandB.Count > 0 And
myTripleCollectionBuilder.colInBNotInA.Count > 0) Then

        Set myGeographyLoader = New GeographyLoader.clsGeographyLoader

        myProgressForm.ProgressBar1.Value = myProgressForm.ProgressBar1.Max / 2

        Dim myVariant As Variant

        For Each myVariant In myTripleCollectionBuilder.colInANotInB

            myGeographyLoader.PublicCountyCollection.Add CStr(myVariant)

        Next

        myProgressForm.ProgressBar1.Value = myProgressForm.ProgressBar1.Max / 1.7

        For Each myVariant In myTripleCollectionBuilder.colBothInAandB

            myGeographyLoader.PublicCountyCollection.Add CStr(myVariant)

        Next

        myProgressForm.ProgressBar1.Value = myProgressForm.ProgressBar1.Max / 1.5
        myGeographyLoader.DatabasePath = App.Path
        'Set myGeographyLoader.PublicCountyCollection =
myTripleCollectionBuilder.colInANotInB
        myGeographyLoader.DatabasePath = App.Path
        myGeographyLoader.DatabaseName = "Master1.mdb"
        myGeographyLoader.SavePath = App.Path & "\ShapeData"
        myGeographyLoader.TableName = "GeoLoadSettings"
     Set SourcePathQueryDef = MasterDatabase.CreateQueryDef("", "select Value from
     Settings where Setting_Name = 'ShapePath'")
        Set SourcePathRecordset = SourcePathQueryDef.OpenRecordset()

        Set myTripleCollectionBuilder = Nothing

        myGeographyLoader.SourcePath = App.Path & "\ShapeData"
'        myGeographyLoader.SourcePath = SourcePathRecordset.Fields(0).Value
        myGeographyLoader.LoadGeography
    MasterDatabase.Execute("Delete from settings where
Setting_Name='GeographyChanged'")
```

```
        MasterDatabase.Execute ("Insert into Settings Values('GeographyChanged',
'True')")

        myProgressForm.ProgressBar1.Value = myProgressForm.ProgressBar1.Max / 1.2

    End If

    myProgressForm.ProgressBar1.Value = myProgressForm.ProgressBar1.Max / 1.2

    Set TempLayer = Nothing
    Set RS1QueryDef = Nothing
    Set RS1Recordset = Nothing
    Set ALASDataBase = Nothing
    Set ALASQueryDef = Nothing
    Set ALASRecordSet = Nothing
    Set SourcePathQueryDef = Nothing
    Set SourcePathRecordset = Nothing
    Set myTripleCollectionBuilder = Nothing
    Set MasterDatabase = Nothing

End Sub

Private Sub RunPreviousQuery()

    Set myManager.CollectionBuilder.DaoRecordset = rsMaster
Set myManager.SQLQuery.CaseCollection = myManager.CollectionBuilder.CaseCollection

End Sub

Private Sub submenuSelectionTools_Click(index As Integer)

    mySettingsForm.Show 1, Me

End Sub
```

Form: frmPreviousQuery

```
Option Explicit

Public Event cmdQueryClick()
Public Event cmdCancelClick()

Private qdCase As QueryDef
Private rsCase As Recordset
Private dbqueries As Database

Private myTripleCollectionBuilder As RecordsetUtilities.clsTripleColBuilder

Private Sub cmdCancel_Click()

    RaiseEvent cmdCancelClick

End Sub

Private Sub cmdQuery_Click()
```

```
    RaiseEvent cmdQueryClick

End Sub

Private Sub Form_Load()

    Dim TempLayer As New MapLayer
    Dim TempDataConnection As New MapObjects2.DataConnection
    TempDataConnection.Database = App.Path & "\ShapeData"
    TempDataConnection.Connect
    Set TempLayer.GeoDataset = TempDataConnection.FindGeoDataset("PR_Bord")

    Dim qdCompare As QueryDef
    Dim rsCompare1 As Recordset
    Dim rsCompare2 As Recordset

    Dim qdqueries As QueryDef
    Dim rsqueries As DAO.Recordset

    Set dbqueries = OpenDatabase(App.Path & "\Master1.mdb")

    Set qdCompare = dbqueries.CreateQueryDef("", "Select County_no from
QueriedCounties")
    Set rsCompare1 = qdCompare.OpenRecordset()

    Set myTripleCollectionBuilder = New RecordsetUtilities.clsTripleColBuilder

    'Using the TCB to build collections and compare

    dbqueries.Execute ("Delete * from TempCounty")
    Set rsCompare2 = dbqueries.OpenRecordset("TempCounty")

    Dim rsTempMORecordset As MapObjects2.Recordset

    Set rsTempMORecordset = TempLayer.Records

    rsTempMORecordset.MoveFirst

    Do While Not rsTempMORecordset.EOF

        rsCompare2.AddNew
        If rsTempMORecordset.Fields("Co_Number").Value <> "" Then

            Debug.Print rsTempMORecordset.Fields("Co_Number").Value
            rsCompare2!County_Number = rsTempMORecordset.Fields("Co_Number").Value

        End If

        rsTempMORecordset.MoveNext
        rsCompare2.Update

    Loop

    'Loading legal queries only

    If Not rsCompare1 Is Nothing And Not rsCompare2 Is Nothing Then
```

```
        If rsCompare1.RecordCount <> 0 And rsCompare2.RecordCount <> 0 Then

            Set myTripleCollectionBuilder.Recordset1 = rsCompare1
            Set myTripleCollectionBuilder.Recordset2 = rsCompare2

            If myTripleCollectionBuilder.colBothInAandB.Count <> 0 Then

                Dim intConcatColInt1 As Integer

                Dim colConcatenatedCollection As New Collection

                For intConcatColInt1 = 1 To
myTripleCollectionBuilder.colBothInAandB.Count

                    colConcatenatedCollection.Add
myTripleCollectionBuilder.colBothInAandB(intConcatColInt1)

                Next intConcatColInt1

                Dim intConcatColInt2 As Integer

                For intConcatColInt2 = 1 To
myTripleCollectionBuilder.colInBNotInA.Count

                    colConcatenatedCollection.Add
myTripleCollectionBuilder.colInBNotInA(intConcatColInt2)

                Next intConcatColInt2

                Dim colTempCollection As New Collection
                Dim strCountyNumbers As String
                Dim intI As Integer

                strCountyNumbers = "ID='"

                Debug.Print colConcatenatedCollection.Count

                For intI = 1 To colConcatenatedCollection.Count

                    Debug.Print colConcatenatedCollection.Count
                    Debug.Print intI
                    If intI <> colConcatenatedCollection.Count Then

                        strCountyNumbers = strCountyNumbers &
colConcatenatedCollection(intI) & "_"

                    Else

                        strCountyNumbers = strCountyNumbers &
colConcatenatedCollection(intI) & "'"

                    End If

                Next
```

```
                Debug.Print "select Query_name from QueriedCounties where " &
strCountyNumbers & ""
                Set qdqueries = dbqueries.CreateQueryDef("", "select Query_name
from QueriedCounties where " & strCountyNumbers & "")
                Set rsqueries = qdqueries.OpenRecordset()

'               rsqueries.Sort = "County_No"
'               rsqueries.MoveFirst

                Do While Not rsqueries.EOF

                    On Error Resume Next
                    colTempCollection.Add              rsqueries.Fields(0).Value,
rsqueries.Fields(0).Value
                    rsqueries.MoveNext

                Loop


                If Not colTempCollection Is Nothing Then

                    Dim I As Integer

                    For I = 1 To colTempCollection.Count

                        Debug.Print colTempCollection.Item(I)
                        cmbQuery.AddItem (colTempCollection.Item(I))

                    Next I

                End If

            Else

                MsgBox ("No Queries stored"), , "MapInterface 1.0"

            End If

        End If

    End If

    Set rsqueries = Nothing
    Set dbqueries = Nothing
    Set colTempCollection = Nothing

End Sub

Private Sub Form_Unload(Cancel As Integer)

    Set qdCase = Nothing
    Set rsCase = Nothing
    Set dbqueries = Nothing
    Set myTripleCollectionBuilder = Nothing

End Sub
```

Form: frmQueryName

```
Public Event cmdOkClick()
Public Event cmdCancelClick()

Private Sub cmdCancel_Click()
    RaiseEvent cmdCancelClick
End Sub

Private Sub cmdOk_Click()
    RaiseEvent cmdOkClick
End Sub

Private Sub Form_Load()
    cmdOk.Enabled = False
End Sub

Private Sub Text1_KeyUp(KeyCode As Integer, Shift As Integer)
    If Text1.Text <> "" Then
        cmdOk.Enabled = True
    Else
        cmdOk.Enabled = False
    End If
End Sub
```

Form: frmSaveQuery

```
Public Event cmdOkClick()
Public Event cmdCancelClick()

Private Sub cmdCancel_Click()
    RaiseEvent cmdCancelClick
End Sub

Private Sub cmdOk_Click()
    RaiseEvent cmdOkClick
End Sub

Private Sub Form_Load()
    cmdOk.Enabled = False
End Sub

Private Sub Text1_KeyUp(KeyCode As Integer, Shift As Integer)
    If Text1.Text <> "" Then
        cmdOk.Enabled = True
    Else
        cmdOk.Enabled = False
    End If
End Sub
```

Form: frmSettings

```
Option Explicit

Public Event SearchDistanceSelected(SearchDistance As Integer)

Private myConvertUnits As New ConvertUnits.clsConvertUnits

Private dbSearchDistanceSettings As Database
Private qdSearchDistanceSettings As QueryDef
Private rsSearchDistanceSettings As Recordset
Private intSliderValue As Integer

Private intVolatileValue As Integer

Private Sub cmdSSTabCancel_Click()

    slrSearchDistance.Value = intSliderValue
    Unload Me

End Sub

Private Sub cmdSSTabOK_Click()

   DoEvents
   RaiseEvent SearchDistanceSelected(CDbl(slrSearchDistance.Value))
   Unload Me

End Sub

Private Sub Form_Load()

    Me.Height = SSTab1(1).Height
    Me.Width = SSTab1(1).Width

    Set dbSearchDistanceSettings = OpenDatabase(App.Path & "\Master1.mdb")
    Set  qdSearchDistanceSettings  =  dbSearchDistanceSettings.CreateQueryDef("",
"Select * from searchdistancesettings")
    Set rsSearchDistanceSettings = qdSearchDistanceSettings.OpenRecordset()

    slrSearchDistance.Min = rsSearchDistanceSettings.Fields(0).Value
    slrSearchDistance.Max = rsSearchDistanceSettings.Fields(1).Value
    lblSearchDistancelow.Caption = rsSearchDistanceSettings.Fields(2).Value
    lblSearchDistanceHigh.Caption = rsSearchDistanceSettings.Fields(3).Value
    lblSearchDistanceUnits.Caption = rsSearchDistanceSettings.Fields(4).Value

    If lblSearchDistanceUnits.Caption = "meters" Then

        optMeters.Value = True
        slrSearchDistance.TickFrequency = 25
        slrSearchDistance.Value = CDbl(rsSearchDistanceSettings.Fields(5).Value)
        txtDistanceSelected.Text = slrSearchDistance.Value

    ElseIf lblSearchDistanceUnits.Caption = "feet" Then

        optFeet.Value = True
        slrSearchDistance.TickFrequency = 15
```

```
        slrSearchDistance.Value  =  CDbl((rsSearchDistanceSettings.Fields(5).Value)
/ 3.280839895)
        txtDistanceSelected.Text = Round(slrSearchDistance.Value * 3.280839895, 0)

    ElseIf lblSearchDistanceUnits.Caption = "kilometers" Then

         optKilometers.Value = True
         slrSearchDistance.TickFrequency = 500
         slrSearchDistance.Value  =  CDbl(rsSearchDistanceSettings.Fields(5).Value *
1000)
         txtDistanceSelected.Text = Round((slrSearchDistance.Value) / 1000, 2)

    ElseIf lblSearchDistanceUnits.Caption = "miles" Then

        optMiles.Value = True
        slrSearchDistance.TickFrequency = 500
        slrSearchDistance.Value  =  CDbl(rsSearchDistanceSettings.Fields(5).Value  *
1609)
        txtDistanceSelected.Text = Round((slrSearchDistance.Value) / 1609, 2)

    ElseIf lblSearchDistanceUnits.Caption = "yards" Then

        optYards.Value = True
        slrSearchDistance.TickFrequency = 25
        slrSearchDistance.Value  =  CDbl(rsSearchDistanceSettings.Fields(5).Value  *
1.093613298)
        txtDistanceSelected.Text  =  Round((slrSearchDistance.Value)  /  1.093613298,
0)

    End If

    intVolatileValue = slrSearchDistance.Value

End Sub

Private Sub Form_Unload(Cancel As Integer)

    dbSearchDistanceSettings.Execute ("delete * from SearchDistanceSettings")
    dbSearchDistanceSettings.Execute ("Insert into SearchDistanceSettings values("
&    slrSearchDistance.Min    &    ","    &    slrSearchDistance.Max    &    ",'"    &
lblSearchDistancelow.Caption  &  "','"  &  lblSearchDistanceHigh.Caption  &  "','"  &
lblSearchDistanceUnits & "'," & CDbl(txtDistanceSelected.Text) & ")")

    Set myConvertUnits = Nothing
    Set dbSearchDistanceSettings = Nothing
    Set qdSearchDistanceSettings = Nothing
    Set rsSearchDistanceSettings = Nothing

End Sub

Private Sub optFeet_Click()

    If optFeet.Value = True Then

        lblSearchDistanceUnits.Caption = "feet"
        lblSearchDistancelow.Caption = 1
        lblSearchDistanceHigh.Caption = 1000
```

```
        slrSearchDistance.Min   =   CInt(myConvertUnits.Convert_All_Units(1,   Feet,
Meters))
        slrSearchDistance.Max  =  CInt(myConvertUnits.Convert_All_Units(1000,  Feet,
Meters))
        slrSearchDistance.Value = 1

        slrSearchDistance.TickFrequency = 15
        txtDistanceSelected.Text = slrSearchDistance.Value

    End If

End Sub

Private Sub optKilometers_Click()

    If optKilometers.Value = True Then

        lblSearchDistanceUnits.Caption = "kilometers"

        lblSearchDistancelow.Caption = 1
        lblSearchDistanceHigh.Caption = 10
        slrSearchDistance.Max      =      CInt(myConvertUnits.Convert_All_Units(10,
Kilometers, Meters))
        slrSearchDistance.Min      =      CInt(myConvertUnits.Convert_All_Units(1,
Kilometers, Meters))

        slrSearchDistance.Value = 1000
        slrSearchDistance.TickFrequency = 500
        txtDistanceSelected.Text = slrSearchDistance.Value / 1000

    End If

End Sub

Private Sub optMeters_Click()

    If optMeters.Value = True Then

        lblSearchDistanceUnits.Caption = "meters"

        lblSearchDistancelow.Caption = 1
        lblSearchDistanceHigh.Caption = 1000
        slrSearchDistance.Min  =  CInt(myConvertUnits.Convert_All_Units(1,  Meters,
Meters))
        slrSearchDistance.Max     =     CInt(myConvertUnits.Convert_All_Units(1000,
Meters, Meters))
        slrSearchDistance.Value = 1

        slrSearchDistance.TickFrequency = 25
        txtDistanceSelected.Text = slrSearchDistance.Value

    End If

End Sub

Private Sub optMiles_Click()
```

```
    If optMiles.Value = True Then

        lblSearchDistanceUnits.Caption = "miles"

        lblSearchDistancelow.Caption = 1
        lblSearchDistanceHigh.Caption = 10
        slrSearchDistance.Max  =  CInt(myConvertUnits.Convert_All_Units(10,  Miles,
Meters))
        slrSearchDistance.Min  =  CInt(myConvertUnits.Convert_All_Units(1,  Miles,
Meters))
        slrSearchDistance.Value = 1

        slrSearchDistance.TickFrequency = 500

        txtDistanceSelected.Text = slrSearchDistance.Value / 1609

    End If

End Sub

Private Sub optYards_Click()

    If optYards.Value = True Then

        lblSearchDistanceUnits.Caption = "yards"

        lblSearchDistancelow.Caption = 1
        lblSearchDistanceHigh.Caption = 1000
        slrSearchDistance.Min  =  CInt(myConvertUnits.Convert_All_Units(1,  Yards,
Meters))
        slrSearchDistance.Max = CInt(myConvertUnits.Convert_All_Units(1000, Yards,
Meters))
        slrSearchDistance.Value = 1

        slrSearchDistance.TickFrequency = 25
        txtDistanceSelected.Text = slrSearchDistance.Value

    End If

End Sub

Private   Sub   slrSearchDistance_OLEStartDrag(Data   As   ComctlLib.DataObject,
AllowedEffects As Long)

    intSliderValue = slrSearchDistance.Value

End Sub

Private Sub slrSearchDistance_Scroll()

    If lblSearchDistanceUnits.Caption = "meters" Then

        txtDistanceSelected.Text = slrSearchDistance.Value

    ElseIf lblSearchDistanceUnits.Caption = "feet" Then

        txtDistanceSelected.Text = Round(slrSearchDistance.Value * 3.280839895, 0)
```

```
    ElseIf lblSearchDistanceUnits.Caption = "kilometers" Then

        txtDistanceSelected.Text = Round(slrSearchDistance.Value / 1000, 2)

    ElseIf lblSearchDistanceUnits.Caption = "miles" Then

        txtDistanceSelected.Text = Round(slrSearchDistance.Value / 1609, 2)

    ElseIf lblSearchDistanceUnits.Caption = "yards" Then

        txtDistanceSelected.Text = Round(slrSearchDistance.Value * 1.093613298, 0)

    End If

End Sub

Private Sub txtDistanceSelected_LostFocus()

    slrSearchDistance.Value = txtDistanceSelected.Text

End Sub
```

Form: frmSplashScreen

```
Private Sub Form_activate()

    Dim n As Single
    Dim o As Single

    Me.Refresh
    n = Timer
newtime:     o = Timer
    If o - n > 2.25 Then
    Else: GoTo newtime
    End If

    Unload Me

End Sub

Private Sub Form_Load()

    Me.Width = Picture1.Width
    Me.Height = Picture1.Height
    Me.Refresh

End Sub
```

Project: prjInterfaceObj
Class: clsCollection

```
Option Explicit
Private mvarCrashCollection As New Collection

Public Sub Add(Item As clsTypes)
    mvarCrashCollection.Add Item
End Sub

Public Sub Remove(Item As clsTypes)
    mvarCrashCollection.Remove Item
End Sub

Public Property Get NewEnum() As IUnknown
    'this property allows you to enumerate
    'this collection with the For...Each syntax
    Set NewEnum = mvarCrashCollection.[_NewEnum]
End Property

Private Sub Class_Terminate()
    Set mvarCrashCollection = Nothing
End Sub

Public Property Get Item(vntIndexKey As Variant) As clsTypes
    'used when referencing an element in the collection
    'vntIndexKey contains either the Index or Key to the collection,
    'this is why it is declared as a Variant
    'Syntax: Set foo = x.Item(xyz) or Set foo = x.Item(5)
  Set Item = mvarCrashCollection(vntIndexKey)
End Property

Public Property Get Count() As Long
    'used when retrieving the number of elements in the
    'collection. Syntax: Debug.Print x.Count
    Count = mvarCrashCollection.Count
End Property



'Private CrashCollection As New Collection
'Private mvarCrashNum As String
'Private MyCrashCollection As New Map.clsQuery
'Private mvarRecSet As MapObjects2.Recordset
'
'Property Set Recset(mdata As MapObjects2.Recordset)
'    Set mvarRecSet = mdata
'    mvarRecSet.MoveFirst
'   While Not mvarRecSet.EOF
'    CrashCollection.Add (mvarRecSet.Fields("case_num"))
'    mvarRecSet.MoveNext
'   Wend
'
'    Set MyCrashCollection.CrashCollection = CrashCollection
'
'End Property
```

```
'
'Property Let CrashNum(mdata As String)
'    mvarCrashNum = mdata
'End Property
'Property Get CrashNum() As String
'    CrashNum = mvarCrashNum
'    MyCrashCollection.CrashCode = mvarCrashNum
'End Property
'
''Private Sub Class_initialize()
''    mvarRecSet.MoveFirst
''   While Not mvarRecSet.EOF
''    CrashCollection.Add (mvarRecSet.Fields("case_num"))
''    mvarRecSet.MoveNext
''   Wend
''    Set MyCrashCollection.CrashCollection = CrashCollection
''    MyCrashCollection.CrashCode = mvarCrashNum
```

Class: clsCollectionBuilder

```
Option Explicit
Private mvarCrashCollection As New Collection

Public Sub Add(Item As clsTypes)
    mvarCrashCollection.Add Item
End Sub

Public Sub Remove(Item As clsTypes)
    mvarCrashCollection.Remove Item
End Sub

Public Property Get NewEnum() As IUnknown
    'this property allows you to enumerate
    'this collection with the For...Each syntax
    Set NewEnum = mvarCrashCollection.[_NewEnum]
End Property

Private Sub Class_Terminate()
    Set mvarCrashCollection = Nothing
End Sub

Public Property Get Item(vntIndexKey As Variant) As clsTypes
    'used when referencing an element in the collection
    'vntIndexKey contains either the Index or Key to the collection,
    'this is why it is declared as a Variant
    'Syntax: Set foo = x.Item(xyz) or Set foo = x.Item(5)
  Set Item = mvarCrashCollection(vntIndexKey)
End Property

Public Property Get Count() As Long
    'used when retrieving the number of elements in the
    'collection. Syntax: Debug.Print x.Count
    Count = mvarCrashCollection.Count
End Property
```

Class: clsSQLQuery

```
Option Explicit
Private mvarCrashCode As String
Private MyCrashCollection As clsCollection
Private d As Database
Private DBSettings As Database
Private QDSettings As QueryDef
Private RSSettings As Recordset
Private temp As String

Private mstrDatabasePath As String
Private mstrDatabaseName As String

Property Let DatabasePath(mdata As String)
    mstrDatabasePath = mdata
End Property

Property Get DatabasePath() As String
    DatabasePath = mstrDatabasePath
End Property

Property Let DatabaseName(mdata As String)
    mstrDatabaseName = mdata
End Property

Property Get DatabaseName() As String
    DatabaseName = mstrDatabaseName
End Property

Property Set CaseCollection(mdata As clsCollection)
    Set MyCrashCollection = mdata
    Call RunQuery
End Property

Property Let CrashCode(mdata As String)
    mvarCrashCode = mdata
End Property

Property Get CrashCode() As String
    CrashCode = mvarCrashCode
End Property

Private Sub RunQuery()

    Set DBSettings = OpenDatabase(mstrDatabasePath & mstrDatabaseName)
    Set QDSettings = DBSettings.CreateQueryDef("", "select value from settings
where setting_name='ALASDatabasePath'")
    Set RSSettings = QDSettings.OpenRecordset()
    temp = RSSettings.Fields(0)
    Set d = OpenDatabase(temp)

    Dim CrashStr As String
    Dim intInnerLoop As Integer
    Dim intOuterLoop As Integer
    Dim intStartItem As Integer
    Dim intEndItem As Integer
```

```
    Dim intX As Integer

    If (MyCrashCollection.Count / 200 - CInt(MyCrashCollection.Count / 200)) > 0
Then

        intX = CInt(MyCrashCollection.Count / 200 + 1)

    Else

        intX = CInt(MyCrashCollection.Count / 200)

    End If

    d.Execute ("Delete * from tblaccidentslocform")

    For intOuterLoop = 1 To intX

        intStartItem = (intOuterLoop * 200) - 199

        If MyCrashCollection.Count > intStartItem + 199 Then

            intEndItem = intStartItem + 199

        Else

            intEndItem = MyCrashCollection.Count

        End If

        intInnerLoop = intStartItem
        CrashStr = MyCrashCollection.Item(intInnerLoop).CaseNumber

        For intInnerLoop = intStartItem To intEndItem

            CrashStr = CrashStr + " or " + "case_year & '0' & case_k=" +
MyCrashCollection.Item(intInnerLoop).CaseNumber

        Next intInnerLoop

        d.Execute ("Insert into tblaccidentslocform select " & _
        "Case_year , Case_k, County_I, City_I, Accident_D, Event_time,
Loc_of_accident_I, Total_Vehs_q, Total_damage_m, Type_of_accident_I,
Roadway_Geometric_I, Charactr_of_rdwy_I, Locality_I, Light_conditions_I,
Weather_cond1_I, Collision_type_I, Rural_urban_I, Locator_route_i,
Locator_rd_class_i, Intrsct_class_i, Intersection_i, Reference_node_i,
Dist_ind_miles_q, Direction_node_i, Weekday, Major_cause, Surface_condition,
Fatal_injury, Major_injury, Minor_injury, Possible_injury, Severity,
Rural_Municipal, tblVehicle_case_year_k, tblVehicle_case_k, Veh_unit_k, Veh_year,
Veh_type_i, Special_use_i, Total_occupants_q, Attachment_i, Fire_explosion_i,
Hit_and_run_i, Init_point_impct_i, Damage_area1_veh_i, Damage_severity_i,
Veh_defect_i, Init_dir_of_trav_i, Speed_mph_i, Rdwy_env_ctr_cir_i,
Driver_condition_i, Drv_veh_ctr_cir1_i, Traffic_controls_i, Type_of_trffcway_i,
Traffic_flow_i, Type_of_surface_i, Veh_action_i, Fixed_obj_strck_i,
Loc_fix_obj_strk_i, Surface_cond1_i, " & _
        "vision_obscured_i, Gender_i, Lic_restr1_i, Lic_endor1_i,
Res_endor_cmpll_f, Drvr_charged_f, Drvr_test_result_i, tblaccidents.[Driver
```

```
Age],tblinjuredtable_case_year_k,tblinjuredtable_case_k,tblinjuredtable_injured_nu
mber_k,Age_i,Pos_of_inj_person_1,tblInjuryDetails_case_year,case_number,injury_num
ber,injury_severity,tblInjuredOccupant_case_year_k,tblInjuredOccupant_case_k,tblIn
juredOccupant_injured_number_k,protective_device1_i    from    tblaccidents    where
case_year & '0' & case_k = " & CrashStr)

    Next intOuterLoop

    Set MyCrashCollection = Nothing

End Sub


Private Sub Class_Terminate()

    Set MyCrashCollection = Nothing
    Set d = Nothing
    Set DBSettings = Nothing
    Set QDSettings = Nothing
    Set RSSettings = Nothing

End Sub
```

Class: clsTypes

```
Option Explicit
Private mvarCaseNumber As String

Property Let CaseNumber(mdata As String)
    mvarCaseNumber = mdata
End Property

Property Get CaseNumber() As String
    CaseNumber = mvarCaseNumber
End Property
```

Project: SelectTools
Class: clsLoadSavedQuery

```
Option Explicit
Private mvarCaseNumber As String

Property Let CaseNumber(mdata As String)
    mvarCaseNumber = mdata
End Property

Property Get CaseNumber() As String
    CaseNumber = mvarCaseNumber
End Property
```

Class: clsManager

```
Option Explicit

Public Event CrashesSelected(ShapeDrawn As Object, CrashRecords As
MapObjects2.Recordset)
Public Event SegmentPointsCollectionPassed(PointsCollection As MapObjects2.Points)
```

```
Private WithEvents mySelectionTools As clsSelectionTools
Private myCollectionBuilder As New prjInterfaceObj.clsCollectionBuilder
Private mySQLQuery As New prjInterfaceObj.clsSQLQuery
Private myStoreQuery As New clsStoreQuery

Private mstrDatabaseName As String
Private mstrDatabasePath As String
Private mstrGeoTableName As String
Private mstrCaseTableName As String
Private WithEvents momapCrashMap As MapObjects2.Map
Private mbolEnabled As Boolean

Property Get SelectionTools() As clsSelectionTools
    Set SelectionTools = mySelectionTools
End Property

Property Get CollectionBuilder() As prjInterfaceObj.clsCollectionBuilder
    Set CollectionBuilder = myCollectionBuilder
End Property

Property Get SQLQuery() As prjInterfaceObj.clsSQLQuery
    Set SQLQuery = mySQLQuery
End Property

Property Get StoreQuery() As clsStoreQuery
    Set StoreQuery = myStoreQuery
End Property

Property Let DatabaseName(mdata As String)

    mstrDatabaseName = mdata
    mySelectionTools.DatabaseName = mstrDatabaseName
    mySQLQuery.DatabaseName = mstrDatabaseName
    myStoreQuery.DatabaseName = mstrDatabaseName

End Property

Property Get DatabaseName() As String

    DatabaseName = mstrDatabaseName

End Property

Property Let DatabasePath(mdata As String)

    mstrDatabasePath = mdata
    mySelectionTools.DatabasePath = mstrDatabasePath
    mySQLQuery.DatabasePath = mstrDatabasePath
    myStoreQuery.DatabasePath = mstrDatabasePath

End Property

Property Get DatabasePath() As String

   DatabasePath = mstrDatabasePath
```

```
End Property

Property Let GeoTableName(mdata As String)

    mstrGeoTableName = mdata

End Property

Property Get GeoTableName() As String

    GeoTableName = mstrGeoTableName

End Property

Property Let CaseTableName(mdata As String)

    mstrCaseTableName = mdata

End Property

Property Get CaseTableName() As String

    CaseTableName = mstrCaseTableName

End Property

Property Set CrashMap(mdata As Object)

    Set momapCrashMap = mdata
    Set mySelectionTools.CrashMap = momapCrashMap

End Property

Property Let Enabled(mdata As Boolean)

    mbolEnabled = mdata
    mySelectionTools.Enabled = mdata

End Property

Property Get Enabled() As Boolean

    Enabled = mbolEnabled

End Property

Private Sub Class_Initialize()

    Set mySelectionTools = New clsSelectionTools

End Sub

Private Sub Class_Terminate()

    Set mySelectionTools = Nothing
    Set myCollectionBuilder = Nothing
    Set mySQLQuery = Nothing
```

```
    Set myStoreQuery = Nothing
    Set momapCrashMap = Nothing

End Sub

Private Sub mySelectionTools_RecordsetFound(ShapeDrawn As Object, IsValid As
Boolean, rsSelectedCrashes As MapObjects2.Recordset)

    If IsValid = True Then

        RaiseEvent CrashesSelected(ShapeDrawn, rsSelectedCrashes)
'        Set myCollectionBuilder.MapRecordset = rsSelectedCrashes

    End If

End Sub

Private Sub mySelectionTools_SegmentPointsCollection(PointsCollection As
MapObjects2.Points)

    RaiseEvent SegmentPointsCollectionPassed(PointsCollection)

End Sub
```

Class: clsSelectionTools

```
Option Explicit

Public Event RecordsetFound(ShapeDrawn As Object, IsValid As Boolean,
rsSelectedCrashes As MapObjects2.Recordset)
Public Event SegmentPointsCollection(PointsCollection As MapObjects2.Points)

Private mySelectFeatures As SelectFeatures.clsSelectFeatures
Private WithEvents mySnapManager As SnapFunctions.clsSnapManager
Private WithEvents myIntersectionFinderForm As frmIntFinder
Private myLoadSavedQuery As clsLoadSavedQuery

Private WithEvents momapCrashMap As MapObjects2.Map
Private mbolEnabled As Boolean
Private mintBufferDistance As Integer

Public Enum stMode

    stPoint = 1
    stRectangle = 2
    stPolygon = 3
    stIntersection = 4
    stRouteSelection = 5
    stCheckStackedCrashes = 7

End Enum

Private ShapeMode As stMode

Private mbolGeographyChanged As Boolean
Private mstrDatabaseName As String
```

```
Private mstrDatabasePath As String
Private mstrSQL As String
Private rsSelectedCrashes As MapObjects2.Recordset
Private MySymbol As New MapObjects2.Symbol
Private objCrashShape As Object
Private mstrSnapLayerName As String ' Move upto to manager if common layer to all
classes
Private mstrCrashLayerName As String
Private mintSearchDistance As Integer
Private mstrQueryName As String
Private mbolPreviousQueryEnabled As Boolean

Private ptMousePoint As New MapObjects2.Point
Private bolPointQuery As Boolean
Private bolRouteSelectionPreviousQuery As Boolean
Private rsSelectedSegments As MapObjects2.Recordset
Private ptsCollection As New MapObjects2.Points
Private bolClearSelection As Boolean
Private rsCrashesStacked As MapObjects2.Recordset

Property Let GeographyChanged(mdata As Boolean)

    mbolGeographyChanged = mdata

End Property

Property Get GeographyChanged() As Boolean

    GeographyChanged = mbolGeographyChanged

End Property

Property Let DatabaseName(mdata As String)

    mstrDatabaseName = mdata
    myLoadSavedQuery.DatabaseName = mstrDatabaseName

End Property

Property Get DatabaseName() As String

    DatabaseName = mstrDatabaseName

End Property

Property Let DatabasePath(mdata As String)

    mstrDatabasePath = mdata
    myLoadSavedQuery.DatabasePath = mstrDatabasePath

End Property

Property Get DatabasePath() As String

    DatabasePath = mstrDatabasePath

End Property
```

```
Property Get SnapManager() As clsSnapManager
    Set SnapManager = mySnapManager
End Property
Property Let KeyField(mdata As String)
    mySnapManager.KeyField = mdata
End Property
Property Get KeyField() As String
    KeyField = mySnapManager.KeyField
End Property
Property Let SnapDistance(mdata As Integer)
'    mySnapManager.SnapDistance = mdata
End Property
Property Get SnapDistance() As Integer
    SnapDistance = mySnapManager.SnapDistance
End Property
Property Let SearchDistance(mdata As Integer)
    mintSearchDistance = mdata
    If ShapeMode = stPoint Then
        BuildEllipseRecordset
    End If
End Property
Property Get SearchDistance() As Integer
    SearchDistance = mintSearchDistance
End Property
Property Let SQL(mdata As String)
    mstrSQL = mdata
    mySnapManager.SQL = mdata
End Property
Property Get SQL() As String
```

```
    SQL = mstrSQL

End Property

Property Let SnapLayerName(mdata As String)

    mstrSnapLayerName = mdata
'    mySnapManager.LayerName = mdata

End Property

Property Get SnapLayerName() As String

    SnapLayerName = mstrSnapLayerName

End Property

Property Let CrashLayerName(mdata As String)

    mstrCrashLayerName = mdata

End Property

Property Get CrashLayerName() As String

    CrashLayerName = mstrCrashLayerName

End Property

Property Set CrashMap(mdata As Object)

    Set momapCrashMap = mdata
'    Set mySnapManager.Map = mdata

End Property

Property Let Enabled(mdata As Boolean)

    mbolEnabled = mdata

    If mbolEnabled = False Then

        mySelectFeatures.Enabled = False

    End If

End Property

Property Get Enabled() As Boolean

    Enabled = mbolEnabled

End Property

Property Get Mode() As stMode
```

```
    Mode = ShapeMode

End Property

Property Let Mode(mdata As stMode)

    ShapeMode = mdata

    If Mode = stRouteSelection Then

        BuildSegmentsRecordset

    End If

End Property

Property Let QueryName(mdata As String)

    mstrQueryName = mdata

End Property

Property Get QueryName() As String

    QueryName = mstrQueryName

End Property

Property Let PreviousQueryEnabled(mdata As Boolean)

    mbolPreviousQueryEnabled = mdata

End Property

Property Get PreviousQueryEnabled() As Boolean

    PreviousQueryEnabled = mbolPreviousQueryEnabled

End Property

Private Sub BuildEllipseRecordset()

    If (mbolEnabled = True Or Mode = stIntersection) And mbolPreviousQueryEnabled
= False Then

        If Not ptMousePoint Is Nothing Then

            Dim moEllipse As New MapObjects2.Ellipse
            moEllipse.Left = ptMousePoint.x - mintSearchDistance
            moEllipse.Bottom = ptMousePoint.y - mintSearchDistance
            moEllipse.Right = ptMousePoint.x + mintSearchDistance
            moEllipse.Top = ptMousePoint.y + mintSearchDistance

            Set objCrashShape = moEllipse

            FindSelectedCrashes
            momapCrashMap.Refresh
```

```
            Set moEllipse = Nothing

        End If

        If Mode = stIntersection Then

            Dim rectLocalRectangle1 As New MapObjects2.Rectangle

            Set rectLocalRectangle1 = objCrashShape.Extent
            rectLocalRectangle1.ScaleRectangle 1.2
            Set momapCrashMap.Extent = rectLocalRectangle1

'            FindSelectedCrashes
'            momapCrashMap.Refresh

            Set rectLocalRectangle1 = Nothing

        End If

    ElseIf mbolPreviousQueryEnabled = True Then

        Dim rectLocalRectangle As New MapObjects2.Rectangle

'        Set rectLocalRectangle = objCrashShape.Extent
'        rectLocalRectangle.ScaleRectangle 1.2
'        Set momapCrashMap.Extent = rectLocalRectangle

        FindSelectedCrashes
        momapCrashMap.Refresh

        Set rectLocalRectangle = Nothing

    End If

End Sub

Private Sub BuildLineRecordset()

End Sub

Public Sub BuildIntersectionRecordset()

    Set myIntersectionFinderForm = New frmIntFinder

    Set myIntersectionFinderForm.ctlIntersectionX1.SourceMapLayer_1 =
momapCrashMap.Layers("ROADS")
    Set myIntersectionFinderForm.ctlIntersectionX1.SourceMapLayer_2 =
momapCrashMap.Layers("ROADS")
    Set myIntersectionFinderForm.ctlIntersectionX1.Map = momapCrashMap

    myIntersectionFinderForm.ctlIntersectionX1.SettingsDatabaseName =
mstrDatabasePath & mstrDatabaseName
    myIntersectionFinderForm.ctlIntersectionX1.SchemaName = "Roadway"

    myIntersectionFinderForm.ctlIntersectionX1.Init
```

```
    Set myIntersectionFinderForm.ctlIntersectionX1.SourceRecordset =
momapCrashMap.Layers("ROADS").Records

    If mbolGeographyChanged = True Then

        myIntersectionFinderForm.ctlIntersectionX1.Refresh 1, 1, True
        myIntersectionFinderForm.ctlIntersectionX1.WriteListToDatabase 1

        mbolGeographyChanged = False
'        Master_Array(ReturnCounter("GeographyChanged")).Setting = False

    Else

        myIntersectionFinderForm.ctlIntersectionX1.LoadListFromSaveData 1

    End If

    myIntersectionFinderForm.Show 1
    Unload myIntersectionFinderForm
    Set myIntersectionFinderForm = Nothing

End Sub

Private Sub BuildRectangleRecordset()

    If mbolEnabled = True And mbolPreviousQueryEnabled = False Then

        Set objCrashShape = momapCrashMap.TrackRectangle
        FindSelectedCrashes
        momapCrashMap.Refresh

    ElseIf mbolPreviousQueryEnabled = True Then

        Dim rectLocalRectangle As New Rectangle

        Set momapCrashMap.Extent = objCrashShape
        momapCrashMap.Extent.ScaleRectangle 1.2

        FindSelectedCrashes
        momapCrashMap.Refresh

        Set rectLocalRectangle = Nothing

    End If

End Sub

Private Sub BuildPolygonRecordset()

    If mbolEnabled = True And mbolPreviousQueryEnabled = False Then

        Set objCrashShape = momapCrashMap.TrackPolygon
        FindSelectedCrashes

    ElseIf mbolPreviousQueryEnabled = True Then

        Dim rectLocalRectangle As New Rectangle
```

```
        Set rectLocalRectangle = objCrashShape.Extent
        rectLocalRectangle.ScaleRectangle 1.2
        Set momapCrashMap.Extent = rectLocalRectangle

        FindSelectedCrashes

        Set rectLocalRectangle = Nothing

    End If

    momapCrashMap.Refresh

End Sub

Public Sub BuildCountyRecordset(CountyNumber As Integer)

    Dim myCountyLayer As New MapLayer
    Dim myCrashLayer As New MapLayer

    Set myCountyLayer = momapCrashMap.Layers("County")
    Set myCrashLayer = momapCrashMap.Layers("Crashes")

    Set objCrashShape = myCountyLayer.SearchExpression("County = " & CountyNumber)
    Set rsSelectedCrashes = myCrashLayer.SearchShape(objCrashShape, moContaining,
"")
    FindSelectedCrashes

    Dim rectLocalRectangle As MapObjects2.Rectangle

'     Set rectLocalRectangle = objCrashShape.Fields("Shape").Value.Extent
'     rectLocalRectangle.ScaleRectangle 1.2
'     Set momapCrashMap.Extent = rectLocalRectangle
'
'     Set myCountyLayer = Nothing
'     Set myCrashLayer = Nothing
'     Set rectLocalRectangle = Nothing

End Sub

Public Sub BuildCityRecordset(CityName As String)

    Dim myCityLayer As New MapLayer
    Dim myCrashLayer As New MapLayer

    Set myCityLayer = momapCrashMap.Layers("Cities")
    Set myCrashLayer = momapCrashMap.Layers("Crashes")

    Set objCrashShape = myCityLayer.SearchExpression("Place_Name = '" & CityName &
"'")
    Set rsSelectedCrashes = myCrashLayer.SearchShape(objCrashShape, moContaining,
"")
    FindSelectedCrashes

    Dim rectLocalRectangle As MapObjects2.Rectangle

    Set rectLocalRectangle = objCrashShape.Fields("Shape").Value.Extent
```

```
    rectLocalRectangle.ScaleRectangle 1.2
    Set momapCrashMap.Extent = rectLocalRectangle

    Set myCityLayer = Nothing
    Set myCrashLayer = Nothing
    Set rectLocalRectangle = Nothing

End Sub
Private Sub Class_Initialize()

    'Set mySnapManager = New SnapFunctions.clsSnapManager
    'mySnapManager.Visible = True

    Set myLoadSavedQuery = New clsLoadSavedQuery
    Set mySelectFeatures = New SelectFeatures.clsSelectFeatures

End Sub

Private Sub Class_Terminate()

    Set mySelectFeatures = Nothing
    Set mySnapManager = Nothing
    Set myIntersectionFinderForm = Nothing
    Set myLoadSavedQuery = Nothing
    Set momapCrashMap = Nothing

    Set rsSelectedCrashes = Nothing
    Set MySymbol = Nothing
    Set objCrashShape = Nothing
    Set ptMousePoint = Nothing
    Set rsSelectedSegments = Nothing
    Set ptsCollection = Nothing
    Set rsCrashesStacked = Nothing

End Sub

Private Sub momapCrashMap_AfterLayerDraw(ByVal Index As Integer, ByVal canceled As
Boolean, ByVal hDC As stdole.OLE_HANDLE)

    If Mode = 7 Then

        Dim moSym As New MapObjects2.Symbol

            moSym.Color = moBlue
            moSym.SymbolType = moFillSymbol
            moSym.Style = moSolidFill
            moSym.SymbolType = moLineSymbol
            moSym.Size = momapCrashMap.Layers("Roads").Symbol.Size

            Set rsSelectedSegments = mySelectFeatures.SelectedShapes

            Call ShowSelectedFeatures(rsSelectedCrashes, moCyan)

            Set moSym = Nothing

        End If
```

```
If bolClearSelection = False Then

    If Mode = stRouteSelection Then

        Dim sym As New MapObjects2.Symbol

        sym.Color = moBlue
        sym.SymbolType = moFillSymbol
        sym.Style = moSolidFill
        sym.SymbolType = moLineSymbol
        sym.Size = momapCrashMap.Layers("Roads").Symbol.Size

        Set rsSelectedSegments = mySelectFeatures.SelectedShapes
        momapCrashMap.DrawShape rsSelectedSegments, sym
        FindSelectedCrashes
'        momapCrashMap.Refresh

        Call ShowSelectedFeatures(rsSelectedCrashes, moCyan)

        Set sym = Nothing

    End If

    If bolRouteSelectionPreviousQuery = True Then

        Dim moSymbol As New MapObjects2.Symbol

        moSymbol.Color = moBlue
        moSymbol.SymbolType = moFillSymbol
        moSymbol.Style = moSolidFill
        moSymbol.SymbolType = moLineSymbol
        moSymbol.Size = momapCrashMap.Layers("Roads").Symbol.Size

        momapCrashMap.DrawShape rsSelectedSegments, moSymbol
        FindSelectedCrashes
        momapCrashMap.Refresh

        Call ShowSelectedFeatures(rsSelectedCrashes, moCyan)

        Set moSymbol = Nothing

    End If

    If Not rsSelectedSegments Is Nothing Then

        Dim symShape As New MapObjects2.Symbol
        symShape.Color = moBlue
        symShape.SymbolType = moFillSymbol
        symShape.Style = moSolidFill
        symShape.SymbolType = moLineSymbol
        symShape.Size = momapCrashMap.Layers("Roads").Symbol.Size

        momapCrashMap.DrawShape rsSelectedSegments, symShape

        Set symShape = Nothing
```

```
        End If

'        If Not rsCrashesStacked Is Nothing Then
'
'            If rsCrashesStacked.Count <> 0 Then
'
'                Dim textsym As New MapObjects2.TextSymbol
'                textsym.Color = moBlack
'
'    '            momapCrashMap.DrawText rsCrashesStacked.Count, ptMousePoint,
textsym
'
'            End If
'
'        End If

    End If

End Sub

Private Sub momapCrashmap_AfterTrackingLayerDraw(ByVal hDC As stdole.OLE_HANDLE)

    If Not rsSelectedCrashes Is Nothing Then

        If rsSelectedCrashes.Count > 0 Then

            Call ShowSelectedFeatures(rsSelectedCrashes, moCyan)

        End If

    End If

    If Not objCrashShape Is Nothing Then

        Dim symShape As New MapObjects2.Symbol
        symShape.SymbolType = moFillSymbol
        symShape.Style = moTransparentFill
        symShape.OutlineColor = moBlue
        symShape.Size = 7
        momapCrashMap.DrawShape objCrashShape, symShape

        Set symShape = Nothing

    End If

End Sub

Private Sub momapCrashmap_MouseDown(Button As Integer, Shift As Integer, x As
Single, y As Single)

    If Button = 1 Then

        Select Case ShapeMode

        Case 1  'Ellipse Selection

'                If Not mySnapManager.SnapPoint Is Nothing Then
```

```
'                    Set ptMousePoint = mySnapManager.SnapPoint.Point

'                Else

                    Set ptMousePoint = momapCrashMap.ToMapPoint(x, y)

'                End If
                mySelectFeatures.Enabled = False
                Set rsSelectedSegments = Nothing

                BuildEllipseRecordset

        Case 2  'Rectangle Selection

            mySelectFeatures.Enabled = False
            Set ptMousePoint = Nothing
            Set rsSelectedSegments = Nothing

'            mySnapManager.Enabled = False
            BuildRectangleRecordset

        Case 3  'Polygon Selection

            mySelectFeatures.Enabled = False
            Set ptMousePoint = Nothing
            Set rsSelectedSegments = Nothing

'            mySnapManager.Enabled = False
            BuildPolygonRecordset

        Case 5  'Route Selection

            mySelectFeatures.Enabled = True
            mySelectFeatures.LayerName = "Roads"
            Set mySelectFeatures.MainMap = momapCrashMap
            mySelectFeatures.UniqueFieldName = "PLINK"

            If Not objCrashShape Is Nothing Then

                Set objCrashShape = Nothing
                momapCrashMap.Refresh

            End If

        Case 7  ' Countywide Selection of Crashes

        Case 8  ' Citywide Selection of Crashes

        Case 9  'Not a selection mechanism but just a check for number of crashes
stacked

            Set rsSelectedSegments = Nothing
            mySelectFeatures.Enabled = False
            Set ptMousePoint = momapCrashMap.ToMapPoint(x, y)
            CheckStackedCrashes
```

```
        End Select

    End If

End Sub

Private Sub CheckStackedCrashes()

    If mbolEnabled = True Then

        Set rsCrashesStacked =
momapCrashMap.Layers("Crashes").SearchByDistance(ptMousePoint,
momapCrashMap.Layers("Crashes").Symbol.Size, "")

        If rsCrashesStacked.Count <> 0 Then

            MsgBox "Number of Crashes Stacked = " & rsCrashesStacked.Count, , "Map
Interface 1.0"

        End If

    End If

End Sub

Private Sub ShowSelectedFeatures(Recset As MapObjects2.Recordset, Color)

    MySymbol.SymbolType = moPointSymbol
    MySymbol.Style = moSolidFill
    MySymbol.Size = 10
    MySymbol.Color = moCyan

    If Not Recset Is Nothing Then

        momapCrashMap.DrawShape Recset, MySymbol

    End If

End Sub

Private Sub FindSelectedCrashes()

    Dim myLayer As MapLayer

    Set myLayer = momapCrashMap.Layers(mstrCrashLayerName)

    If (Mode = stPoint Or Mode = stIntersection) And mbolPreviousQueryEnabled =
False Then

        Set rsSelectedCrashes = myLayer.SearchByDistance(ptMousePoint,
SearchDistance, mstrSQL)

    ElseIf bolPointQuery = True Then

        Dim dbDatabase As DAO.Database
        Dim qdQueryDef As DAO.QueryDef
        Dim rsRecordset As DAO.Recordset
```

```
        Set dbDatabase = OpenDatabase(mstrDatabasePath & mstrDatabaseName)
        Set qdQueryDef = dbDatabase.CreateQueryDef("", "Select * from SavedPoint
where Query_Name= '" & mstrQueryName & "'")
        Set rsRecordset = qdQueryDef.OpenRecordset()

        ptMousePoint.x = rsRecordset.Fields("X").Value
        ptMousePoint.y = rsRecordset.Fields("Y").Value

        Set rsSelectedCrashes = myLayer.SearchByDistance(ptMousePoint,
rsRecordset.Fields("Radius").Value, mstrSQL)

        Dim LocalEllipse As New MapObjects2.Ellipse

        LocalEllipse.Left = ptMousePoint.x - rsRecordset.Fields("Radius").Value
        LocalEllipse.Bottom = ptMousePoint.y - rsRecordset.Fields("Radius").Value
        LocalEllipse.Right = ptMousePoint.x + rsRecordset.Fields("Radius").Value
        LocalEllipse.Top = ptMousePoint.y + rsRecordset.Fields("Radius").Value

        Set objCrashShape = LocalEllipse

        Set dbDatabase = Nothing
        Set qdQueryDef = Nothing
        Set rsRecordset = Nothing
        Set LocalEllipse = Nothing

    ElseIf Mode = stRouteSelection Then

        Set rsSelectedCrashes = myLayer.SearchByDistance(rsSelectedSegments, 5,
"")

        RaiseEvent
SegmentPointsCollection(mySelectFeatures.ClickedPointsCollection)

    ElseIf bolRouteSelectionPreviousQuery = True Then

        Set rsSelectedCrashes = myLayer.SearchByDistance(rsSelectedSegments, 5,
"")

    ElseIf Mode = 7 Then

        momapCrashMap.Refresh

    Else

        Set rsSelectedCrashes = myLayer.SearchShape(objCrashShape, moContaining,
"")

    End If

'    momapCrashMap.Refresh

'    If rsSelectedCrashes.Count <> 0 Then

    RaiseEvent RecordsetFound(objCrashShape, True, rsSelectedCrashes)

'    End If
```

```
    Set myLayer = Nothing

End Sub

Public Sub ClearSelection()

    Set rsSelectedCrashes = Nothing
    RaiseEvent RecordsetFound(objCrashShape, True, rsSelectedCrashes)
    Set objCrashShape = Nothing
    Set rsSelectedCrashes = Nothing
    Set rsSelectedSegments = Nothing
    bolClearSelection = True
    momapCrashMap.Refresh
    bolClearSelection = False

End Sub

Private Sub myIntersectionFinderForm_ShapeFound(Point As MapObjects2.Point)

' CLient Code -- Respond to Point being found

'    Dim TempRectangle As Rectangle
'
'    Set TempRectangle = momapCrashMap.FullExtent
'
'    TempRectangle.ScaleRectangle 0.0001
'
'    Set momapCrashMap.Extent = TempRectangle
'
'    momapCrashMap.CenterAt Point.x, Point.y
'    momapCrashMap.Refresh

    Set ptMousePoint = Point

    BuildEllipseRecordset

End Sub

Private Sub BuildSegmentsRecordset()

    If mbolEnabled = True And mbolPreviousQueryEnabled = False Then

        mySelectFeatures.Enabled = True
        mySelectFeatures.LayerName = "Roads"
        Set mySelectFeatures.MainMap = momapCrashMap
        mySelectFeatures.UniqueFieldName = "PLINK"

    ElseIf mbolPreviousQueryEnabled = True Then

        Dim rectLocalRectangle As New Rectangle

'        Set rectLocalRectangle = objCrashShape.Extent
        rectLocalRectangle.ScaleRectangle 1.2
'        Set momapCrashMap.Extent = rectLocalRectangle
```

```
        Set rsSelectedSegments =
momapCrashMap.Layers("Roads").SearchByDistance(ptsCollection, 5, "")
        Set objCrashShape = rsSelectedSegments

        bolRouteSelectionPreviousQuery = True
        FindSelectedCrashes
        momapCrashMap.Refresh

        Set rectLocalRectangle = Nothing

    End If

    bolRouteSelectionPreviousQuery = False

End Sub

Public Sub LoadShape(LoadQueryName As String, ShapeIndex As Integer)

    Select Case ShapeIndex

    Case 1

        Set objCrashShape = myLoadSavedQuery.LoadPoint(LoadQueryName)
        bolPointQuery = True
        BuildEllipseRecordset

    Case 2

        Set objCrashShape = myLoadSavedQuery.LoadRectangle(LoadQueryName)
        BuildRectangleRecordset

    Case 3

        Set objCrashShape = myLoadSavedQuery.LoadPolygon(LoadQueryName)
        BuildPolygonRecordset

    Case 4

        Set objCrashShape = myLoadSavedQuery.LoadLine(LoadQueryName)
        BuildLineRecordset

    Case 5

        Set ptsCollection = myLoadSavedQuery.LoadSegments(LoadQueryName)
        BuildSegmentsRecordset

    End Select

End Sub
```

Class: clsStoreQuery

```
Option Explicit

Private mstrDatabaseName As String
Private mstrDatabasePath As String
Private mstrGeoTableName As String
Private mstrCaseTableName As String
Private mstrCaseNumberField As String
Private mvarQueryName As String
Private dbMaster As Database

Property Let DatabaseName(mdata As String)
    mstrDatabaseName = mdata
End Property

Property Get DatabaseName() As String
    DatabaseName = mstrDatabaseName
End Property

Property Let DatabasePath(mdata As String)
    mstrDatabasePath = mdata
End Property

Property Get DatabasePath() As String
    DatabasePath = mstrDatabasePath
End Property

Property Let GeoTableName(mdata As String)
    mstrGeoTableName = mdata
End Property

Property Get GeoTableName() As String
    GeoTableName = mstrGeoTableName
End Property

Property Let CaseTableName(mdata As String)
    mstrCaseTableName = mdata
End Property

Property Get CaseTableName() As String
    CaseTableName = mstrCaseTableName
End Property

Property Let CaseNumberField(mdata As String)
    mstrCaseNumberField = mdata
End Property

Property Get CaseNumberField() As String
    CaseNumberField = mstrCaseNumberField
End Property

Public Sub StorePoint(StoreQueryName As String, Shape As MapObjects2.Point, Radius
As Double)

    Set dbMaster = OpenDatabase(mstrDatabasePath & mstrDatabaseName)
```

```
    dbMaster.Execute ("Insert into SavedPoint values('" & StoreQueryName & "', " &
Shape.x & ", " & Shape.y & ", " & Radius & ")")

End Sub

Public Sub StoreRectangle(StoreQueryName As String, Shape As
MapObjects2.Rectangle)

    Set dbMaster = OpenDatabase(mstrDatabasePath & mstrDatabaseName)
    dbMaster.Execute ("Insert into SavedRectangle values('" & StoreQueryName & "',
" & Shape.Top & ", " & Shape.Left & ", " & Shape.Bottom & ", " & Shape.Right &
")")

End Sub

Public Sub StorePolygon(StoreQueryName As String, Shape As MapObjects2.Polygon)

    Dim moPolygon As MapObjects2.Polygon
    Dim moPoints As MapObjects2.Points

    Dim I As Integer
    Dim ptsVertices As MapObjects2.Points

    Set dbMaster = OpenDatabase(mstrDatabasePath & mstrDatabaseName)

    For Each ptsVertices In Shape.Parts

        For I = 0 To ptsVertices.Count - 1

            dbMaster.Execute ("Insert into SavedPolygonPoints values('" &
StoreQueryName & "', " & ptsVertices.Item(I).x & ", " & ptsVertices.Item(I).y &
")")

        Next I

    Next

End Sub

Public Sub StoreEllipse(StoreQueryName As String, Shape As MapObjects2.Ellipse)

    Set dbMaster = OpenDatabase(mstrDatabasePath & mstrDatabaseName)
    dbMaster.Execute ("Insert into Savedrectangle values('" & StoreQueryName & "',
" & Shape.Top & ", " & Shape.Left & ", " & Shape.Bottom & ", " & Shape.Right &
")")

End Sub

Public Sub StoreCaseNumbers(StoreQueryName As String, CrashRecords As
MapObjects2.Recordset, ShapeIndex As Integer)

    CrashRecords.MoveFirst

    Set dbMaster = OpenDatabase(mstrDatabasePath & mstrDatabaseName)

    While Not CrashRecords.EOF
```

```
        dbMaster.Execute ("Insert into Queries values('" + StoreQueryName + "','"
+ CrashRecords.Fields("Case_Num") + "'," & ShapeIndex & ")")
        CrashRecords.MoveNext

    Wend

End Sub

Public Sub StoreLoadedCountyNumbers(StoreQueryName As String, CountyRecords As
MapObjects2.Recordset)

    CountyRecords.MoveFirst
    Dim strLoadedCountyNumbers As String

    Dim I As Integer

    For I = 1 To CountyRecords.Count

        If I <> CountyRecords.Count Then

            strLoadedCountyNumbers = strLoadedCountyNumbers &
CountyRecords.Fields("Co_Number").Value & "_"

        Else

            strLoadedCountyNumbers = strLoadedCountyNumbers &
CountyRecords.Fields("Co_Number").Value

        End If

        CountyRecords.MoveNext

   Next I

    CountyRecords.MoveFirst

    While Not CountyRecords.EOF

        Set dbMaster = OpenDatabase(mstrDatabasePath & mstrDatabaseName)
        dbMaster.Execute ("Insert into QueriedCounties values('" & StoreQueryName
& "','" & CountyRecords.Fields("Co_Number").ValueAsString & "','" &
strLoadedCountyNumbers & "')")
        CountyRecords.MoveNext

    Wend

End Sub

Public Sub StoreSegments(StoreQueryName As String, PointsCollection As
MapObjects2.Points)

    Set dbMaster = OpenDatabase(mstrDatabasePath & mstrDatabaseName)

    Dim ptLocalPoint As New MapObjects2.Point

    For Each ptLocalPoint In PointsCollection
```

```
        dbMaster.Execute ("Insert into SavedPolygonPoints values('" &
StoreQueryName & "', " & ptLocalPoint.x & ", " & ptLocalPoint.y & ")")

    Next

End Sub

Private Sub Class_Terminate()

    Set dbMaster = Nothing

End Sub
```

APPENDIX C. PERFORMANCE TEST QUERIES

Map Interface Performance Test

Your Name: _____

Query 1

Select crashes between the intersections of ***Lincoln Way and Welch Avenue***, and ***Lincoln Way and Hayward Avenue***.

Time with Access-ALAS = _____

Time with Map Interface = _____

Query 2

Select crashes between the intersections of ***Lincoln Way and Elwood Drive***, and ***Lincoln Way and Grand Avenue***.

Time with Access-ALAS = _____

Time with Map Interface = _____

Query 3

Select crashes at the intersection of ***Thirteenth Street and Grand Avenue.***
(Access-ALAS query name - "Intersection node")

Time with Access-ALAS = _____

Time with Map Interface = _____

Query 4

Select Crashes within 0.5 miles of the intersection of ***S.Duff Avenue and Airport Road***.
(Access-ALAS query name - "Intersection links")

Time with Access-ALAS = _____

Time with Map Interface = _____

Comments (very essential) on time, aesthetics, ease of use etc. (use the flip side if necessary).

REFERENCES

1. Sherry L. Murphy. *Deaths: Final Data for 1998.* Report (PHS) 2000-1120. National Vital Statistics Reports, NCHS, 2000.

2. FARS. Total Fatalities by Posted Speed Limit and Rural vs. Urban. 1998. http://www-fars.nhtsa.dot.gov/www/cfm/wizard/frame.cfm?qryID=PER1. Accessed Feb. 20, 2000

3. United States Department of Transportation. *The Changing Face of Transportation.* US DOT, Washington D.C., 2000

4. Warren E. Hughes "New and Emerging Technologies for Improving Accident Data Collection". Report FHWA-RD-92-097. Federal Highway Administration, United States Department of Transportation, 1993.

5. Warren E. Hughes. *New and Emerging Technologies for Improving Accident Data Collection.* (Report FHWA-RD-92-097. FHWA, US DOT, 1993), quoting Highway Traffic Safety Act of 1970, Standard No.9, "Identification and Surveillance of Accident Locations".

6. Mirza R. Baig, Melvyn Wasserman, David Caruth. GIS-based Accident Analysis and Mitigation System. *Proceedings of the 26th International Traffic Records Forum* (CD-ROM), Portland, Oregon, July 30 – August 3, 2000

7. Brian Limotti. Assistant Systems Manager, Roadway Systems Branch, Washington State Department of Transportation. "Washington State Collision Analysis Systems." Personal E-mail (30 March, 2001).

8. Washington Department of Transportation. Collision Records System. *Proceedings of the 26th International Traffic Records Forum* (CD-ROM), Portland, Oregon, July 30 – August 3, 2000

9. Craig M. Hanchey, Randy Thompson, and Devin Doyle. Automated Collision Database and Reporting System for Nashville. *ITE Journal.* April 2000, pp.24-27.

10. Steve Rich. Research Analyst Principal, Office of Highway Safety, Idaho Transportation Department. "Idaho's Crash Analysis Reporting System" Personal E-mail (13 March 2001).

11. Mary C.S Godin. Highway Research Supervisor, Vermont Agency of Transportation. "Re: Vermont Crash Analysis Systems". Personal E-mail (04 April 2001).

12. Larry Caldwell. Assistant State Traffic Engineer, Virginia Department of Transportation. "Crash Analysis Systems". Personal E-mail (16 April 2001).

13. Sebastian Puglisi. Office of Planning Inventory and Data, Connecticut Department of Transportation. "Crash Analysis Systems". Personal E-mail (17 April 2001).

14. Ray Lewis. Planning and Research Engineer, Traffic Engineering Division, West Virginia Department of Transportation. "RE: Crash Analysis Systems". Personal E-mail (05 April 2001).

15. John L.Nagle. Congestion and Safety Management Engineer, Indiana Department of Transportation. "Re: Crash Analysis Systems" Personal E-mail (10 April 2001).

16. Federal Highway Administration, United States Department of Transportation, GIS Safety Analysis Tools V2.0 (CD-ROM).

17. Iowa Department of Transportation. Traffic and Criminal Software. *National Model* (CD-ROM), 2000.

18. Bob Schultz. Education Consultant for the Iowa DOT "Access-ALAS Overview for Access-ALAS User Survey". Accessed May 02, 2001.

19. Webopedia. Graphical User Interface. Sept. 01, 1996. http://webopedia.internet.com/TERM/G/Graphical_User_Interface_GUI.html. Accessed Apr. 05, 2001.

20. Neal Peterson. Fundamentals of Software. Oct. 05, 1999. http://www.dbu.edu/peterson/softpres/tsld009.htm. Accessed Apr. 05, 2001.

21. Fred Gault. The Future of the Industry. http://www.fgault.com/articles/future.htm. Accessed Apr. 05, 2001.

22. Grady Booch, Ivar Jacobson, James Rumbaugh. *The Unified Modeling Language User Guide*. Addison Wesley Longman, Inc., Massachussets, 1999.

23. Phone interview with Timothy Simodynes, Engineer Intern, Office of Traffic and Safety, Iowa Department of Transportation (15 May 2001).