

**DATA REPRESENTATION FORMATS ON
WORLD WIDE WEB USING
DECISION SUPPORT
SYSTEM**

By

ANIRUDDHA GUJRATHI

Bachelor of Engineering

University of Pune

Pune, India

1994

**Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 1999**

**DATA REPRESENTATION FORMATS ON
WORLD WIDE WEB USING
DECISION SUPPORT
SYSTEM**

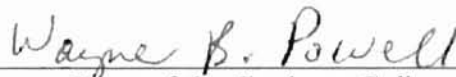
Thesis Approved:



Thesis Adviser







Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my major advisor, Dr. K.M. George for his encouragement, intelligent supervision, constructive guidance and friendly assistance throughout the span of this project.

Many thanks to Dr. D. L. Nofziger for providing me the opportunity and the resources to work on the project and his assessment and guidance during the implementation of this project.

Sincere thanks to Dr. Nick Street and Dr. Hailin Zhang for their help, encouragement and precious advice during my thesis work.

Finally, I would like to thank my parents, Mr. Ashok Gujrathi and Mrs. Prabha Gujrathi, and all my friends for their love, encouragement, patience and understanding.

TABLE OF CONTENTS

1. INTRODUCTION	1
2. LITERATURE REVIEW	3
2.1 PEET, DECISION SUPPORT SYSTEM	3
2.1.1 <i>Inputs to the System</i>	3
2.1.2 <i>Economic Impact</i>	4
2.1.3 <i>Groundwater Hazard</i>	6
2.1.4 <i>System Output</i>	6
2.2 RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS)	7
2.2.1 <i>Structured Query Language (SQL)</i>	7
2.2.1.1 <i>Data Definition Language (DDL)</i>	7
2.2.1.2 <i>Interactive Data Manipulation language (DML)</i>	8
2.2.1.3 <i>Embedded Data Manipulation Language</i>	8
2.2.1.4 <i>View Definition</i>	8
2.2.1.5 <i>Authorization</i>	8
2.2.1.6 <i>Transaction Control</i>	8
2.2.2 <i>Oracle RDBMS</i>	9
2.3 UNIX AND SHELL PROGRAMMING	10
2.4 WORLD WIDE WEB	10
2.4.1 <i>Hyper Text Transfer Protocol (HTTP)</i>	11
2.4.2 <i>Hyper Text Markup Language (HTML)</i>	11
2.4.3 <i>Common Gateway Interface (CGI)</i>	12
2.4.4 <i>JavaScript</i>	13
2.5 JAVA PROGRAMMING LANGUAGE	14
3. DESIGN	15
3.1 OVERALL SYSTEM DESIGN	15
3.2 USER INTERFACE DESIGN	16
3.2.1 <i>Area of Interest</i>	16
3.2.2 <i>Field Conditions</i>	16
3.2.3 <i>Typical Yields</i>	17
3.2.4 <i>Market Value</i>	17
3.2.5 <i>Application Type</i>	18
3.2.6 <i>Weed Densities</i>	18
3.2.7 <i>Pesticide Costs</i>	19
3.2.8 <i>Options</i>	20
3.3 CGI INTERFACE DESIGN	22
3.3.1 <i>Intermediate Server Calls</i>	22
3.3.2 <i>Final Server Call</i>	23
3.4 DATABASE DESIGN	24
3.4.1 <i>Database Schema</i>	24
3.4.2 <i>Project Tables</i>	24
3.5 PEET OUTPUT DESIGN	28
4. IMPLEMENTATION	30

4.1 ORACLE DATA MANAGEMENT.....	30
4.1.1 <i>Creating Tablespaces</i>	30
4.1.2 <i>Creating Users</i>	30
4.1.3 <i>Creating Tables</i>	31
4.1.4 <i>Loading Data into Tables</i>	32
4.2 CGI INTERFACE.....	33
4.3 ORACLE INTERFACE.....	34
4.4 CREATING GRAPHICS.....	35
4.4.1 <i>Server Side Graphics Creation</i>	35
4.4.2 <i>Client Side Graphics Creation</i>	38
5. CONCLUSION & FUTURE WORK	41
5.1 CONCLUSION	41
5.2 FUTURE WORK	42
5.2.1 <i>Decision Support System</i>	42
5.2.2 <i>User Interface Design</i>	42
5.2.3 <i>Customization</i>	43
5.2.4 <i>Security</i>	43
REFERENCES	44
APPENDIX I.....	46
APPENDIX II.....	47
APPENDIX III.....	62
APPENDIX IV.....	70

LIST OF TABLES

1.	ATTRIBUTES OF COUNTY TABLE	24
2.	ATTRIBUTES OF THE SOILNAME TABLE.....	25
3.	ATTRIBUTES OF THE HERB TABLE	25
4.	ATTRIBUTES OF WEED TABLE	25
5.	ATTRIBUTES OF COST TABLE.....	26
6.	ATTRIBUTES OF GWHAZARD TABLE.....	26
7.	ATTRIBUTES OF EFFICACY TABLE	26
8.	ATTRIBUTES OF PARAM TABLE.....	27
9.	ATTRIBUTES OF UNIT TABLE	27
10.	ATTRIBUTES OF TREATMNT TABLE	28
11.	ATTRIBUTES OF YIELD TABLE	28
12.	PRO*C PROGRAMS CREATED FOR PEET.....	34
13.	VARIABLES IN APPLLET PEETGRAPH	39
14.	OBJECTS IN APPLLET PEETGRAPH	40

LIST OF FIGURES

1.	OUTPUT GRAPH FOR THE PEET SYSTEM.	7
2.	FLOW OF DATA IN PEET SYSTEM.....	15
3.	HTML FORM TO ENTER AREA OF INTEREST.	16
4.	HTML FORM TO ENTER FIELD CONDITIONS.....	17
5.	HTML FORM TO ENTER TYPICAL YIELD VALUES.....	17
6.	HTML FORM TO ENTER MARKET VALUE OF YIELD.	17
7.	HTML FORM TO SELECT THE TYPE OF APPLICATION.	18
8.	HTML FORM TO SELECT WEED DENSITIES FOR PPI AND PE.	18
9.	HTML FORM TO ENTER WEED DENSITIES FOR POST EMERGENCE.	19
10.	HTML FORM TO ENTER PESTICIDE COSTS.	20
11.	HTML FORM TO ENTER VARIOUS SYSTEM OPTIONS.	21
12.	PEET ENTRY SCREEN.	21
13.	DATA FLOW IN GET CGI METHOD.	22
14.	DATA FLOW IN POST CGI METHOD.	23
15.	PEET OUTPUT GRAPH CREATED ON THE SERVER.....	37
16.	GRAPH TO SHOW ECONOMIC IMPACT AND GROUNDWATER HAZARD TRADEOFF.	38
17.	JAVA APPLLET TO DISPLAY PEET GRAPH OUTPUT.	40

1. Introduction

The World Wide Web (WWW) technology allows users from anywhere to access the information managed by a WWW server at any time. The WWW server can also access other systems through the Common Gateway Interface (CGI). Oracle is a very powerful relational database management system (RDBMS) which provides full capabilities of data storage, query and data management. The project **Pesticide Economic and Environmental Tradeoff, PEET**, implemented as the thesis work combines all of the above technologies to represent data on the WWW. It uses the client/server technologies like Java and JavaScript for an interactive user interface design and reduced traffic load on the network. Oracle is used to store and retrieve the analysis data since it provides powerful tools for better data management. The WWW server used is Netscape SuiteSpot. This project can also serve as a framework to many other database-related WWW applications. The thesis is organized as follows:

Chapter I (this chapter) provides an introduction to the thesis topic.

Chapter II starts with the background of the PEET system reviewing various terms and mathematical formulas used for computation and analysis work. It also reviews various computer technologies used for implementation of the project.

Chapter III deals with the details of design of the PEET system. It includes the front-end design, the interface design between client and the server and design of the database schema used to store PEET data.

Chapter IV gives the implementation details of the system. It includes various techniques used to access and maintain the Oracle database and various methods used to represent the PEET output data on the client machine.

Finally, conclusions and future work recommendations are given in Chapter V.

Several appendices are included for reference. Appendix I includes the major CGI program of the system. Appendix II is the source code of the PRO*C program that accesses Oracle RDMBS and performs calculations. Appendix III lists Java applet used to generate graphs on the client machine and Appendix IV gives the C code used to do the same on the server side.

2. Literature Review.

As described in Chapter I, **PEET, Pesticide Economic and Environmental Tradeoff**, is implemented as the thesis work. The system makes use of various computing technologies like World Wide Web, Relational Database Management Systems and object-oriented programming for its implementation. This chapter gives a brief overview of the PEET system by providing the background information regarding its purpose and use. Mathematical formulae used for computation and analysis work for the system are discussed in detail. The chapter also reviews all the above-mentioned technologies used to implement the system.

2.1 PEET, Decision Support System

Farmers use various techniques for pest management. It involves compromises between economic gain and risk to the environment [20]. Pesticide Economic and Environmental Tradeoff, PEET, is a decision support system that helps farmers to evaluate both economic and environmental impacts of different pest management practices [20]. PEET is a decision support system in the sense that it helps the farmers to select appropriate pesticides that will aid them to increase yield without harming the natural resources. It also displays the potential hazard on ground water due to the selected pesticide. Thus, this system helps make a decision for selecting a pesticide that minimizes the economic loss as well as the environmental loss [20].

2.1.1 Inputs to the System

Inputs required for this decision support system are [20].

- County in which the field is associated
- Soil Type
- Tillage Practices used
- Irrigation Practices used.
- Type of herbicide treatment.
- Low, normal and high weed free yields
- Cost of applying a pesticide per unit area
- Cost of scouting for weeds per unit area
- Expected market price of crop per unit harvested
- Purchase price and purchase units of each potential pesticide
- Density of each weed species in the field
- Soil moisture conditions at time of treatment
- Weed size at time of treatment
- Approximate date of treatment

2.1.2 Economic Impact

Economic Impact is the term used to determine the loss of yield due to pest with or without different pest management practices [20]. The product of the yield loss and unit value of the crop plus the cost of the pesticide and its application determine the economic loss [20]. PEET determines this component by calculating the loss in yield due to the weeds. The yield reduction, $Y_{LOSS-Abs}$ due to weed population and infestation is given by [20]

$$Y_{LOSS - Abs} = Y_{LOSS - Rel} * Y$$

where

$Y_{LOSS-Rel}$ = relative loss in yield

Y = projected weed-free yield.

The relative loss in yield is given by [20]

$$Y_{LOSS - Rel} = \frac{A * I * D}{A + (I * D)}$$

where,

A = maximum relative field loss,

I = yield loss per unit competitive load for low loads

D = total competitive load

The total competitive load D is the sum of competitive loads for the different weed species and is calculated by the equation [20]

$$D = \sum_{j=1}^n c_j * d_j$$

where,

n = number of weed species infesting the field

c_j = competitive index for weed j

d_j = density of weed j .

The economic loss, E_{LOSS} , due to weeds is given by [20]

$$E_{LOSS} = Y_{LOSS} - Abs * V + C_{Appl} + C_{Scout} + \sum_{Herb=1}^m C_{Herb} * R_{Herb}$$

where,

V = expected value of the crop per unit harvested

C_{Appl} = cost of herbicide application

C_{Scout} = cost of scouting weeds,

C_{Herb} = cost of herbicide Herb per unit applied

R_{Herb} = rate of application of herbicide Herb

m = number of herbicides used in the treatment

2.1.3 Groundwater Hazard.

The use of pesticides causes the degradation of the quality of ground water due to leaching and runoff. The ground water hazard is determined by simulating pesticide movement and degradation through each soil on which peanuts are grown. In PEET, the ground water hazard, GWH, associated with a particular pesticide is defined as the ratio of the estimated concentration, C , of the active ingredient in groundwater to the U.S. EPA lifetime health advisory level, HAL, or the maximum contamination load, MCL, for the active ingredient [20]. That is

$$GWH = \frac{C}{C_{critical}}$$

where,

$$C_{critical} = \text{HAL or MCL for the pesticide.}$$

2.1.4 System Output

The output of the PEET system is essentially a bar graph of economic impact of pesticides against the cost of the pesticide and the potential groundwater hazard due to the pesticide. The user can then decide by selecting the pesticide that maximizes the yield and minimizes the groundwater hazard. Data regarding the same is also presented in a tabular format to the user. The output graph of the system looks as shown in Figure 1.

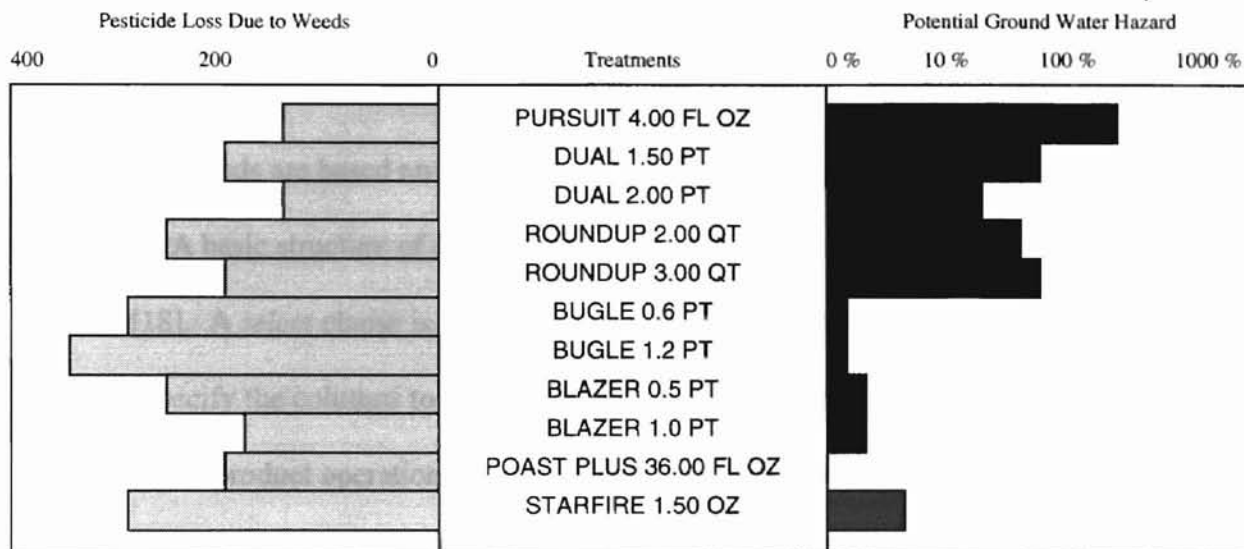


Figure 1. Output graph for the PEET system.

2.2 Relational Database Management System (RDBMS)

Relational databases are the most widely used and primary data model in commercial data processing applications [18]. A relational database consists of a collection of tables. The data and the relationships between the data are both represented using tables. Every table has a unique name in the model and consists of columns.

2.2.1 Structured Query Language (SQL)

SQL is a query language used by most RDBMS systems. It was originally developed at the IBM San Jose Research Labs. It has now been standardized by American National Standards Institute (ANSI). The SQL language has several parts [18]:

2.2.1.1 Data Definition Language (DDL)

DDL provides commands for defining relational schemes, deleting relations, creating indices and modifying schemes. DDL commands specifically include *create table*, *drop table*, *alter table*, *create view* etc.

2.2.1.2 Interactive Data Manipulation language (DML)

DML commands are based on relational algebra and tuple relational calculus to query the database. A basic structure of a DML command consists of three clauses: *select*, *from* and *where* [18]. A *select* clause is based on the projection operation of relational calculus and used to specify the columns to be included in the result. The *from* clause corresponds to the cartesian product operation and contains a list of tables from which to select the data and the *where* clause corresponds to the predicate of relational algebra. It specifies the conditions used on the rows appearing in the final result.

2.2.1.3 Embedded Data Manipulation Language

This form of DML uses procedural languages like C, C++, FORTRAN, and COBOL along with SQL for general purpose programming.

2.2.1.4 View Definition

This is SQL DDL providing commands for creation of a view. Views are basically virtual tables.

2.2.1.5 Authorization

This SQL DDL contains commands dealing with access rights to the objects of the database.

2.2.1.6 Transaction Control

These sets of commands are used for specifying the beginning and end of a transaction.

2.2.2 Oracle RDBMS

Oracle is one of the most commonly used RDBMS software available commercially [5].

The version of Oracle RDBMS used for this project is Oracle7. Oracle7 supports all major operating systems including MS DOS, NetWare, Windows NT, MacOS and most flavors of UNIX [5]. The Oracle networking software SQL*Net supports most of the communication protocols using TCP/IP, SPX/IPX, Named Pipes and DECNet. Oracle7 supports client/server-computing features for reducing the network traffic and is relatively easy to administer [5]. Oracle provides various tools for efficient data management and data retrieval. These tools include:

SQL: Oracle provides a strong SQL engine to perform various database tasks including data management and retrieval. Oracle SQL is compliant to the ANSI SQL standards.

PL/SQL: PL/SQL combines the power of SQL language with the procedural language constructs like loops, conditional statements and variable declaration. This further enhances the capabilities of SQL. Also, since various SQL statements can be packed in a PL/SQL construct, network traffic is reduced and so is the load on the server.

Embedded SQL: Embedded SQL provides the user to include SQL statements in programs written in popular programming languages like C, C++, FORTRAN and COBOL. This tool is widely used to fetch data from the database and generate reports.

SQL*Loader: This tool is used to load data from flat files to database tables. The loader provides various options for conditional loading.

2.3 UNIX and Shell programming

UNIX is a well-known and widely used multi-user, time sharing operating system. It was first developed by Ken Thompson at the AT&T Bell Laboratories in Murray Hill, New Jersey in 1969. It is used in universities, government organizations and industry [23].

A UNIX shell is one of the powerful interfaces to the UNIX operating system. It also provides a very powerful programming language environment. UNIX shell supports various high-level language constructs like conditional statements, loops and functions with parameter passing, subroutine calls and interrupt handling. Three popular UNIX shells are the Bourne shell (sh), C shell (csh) and the Korn shell (ksh).

Bourne shell is used in this project for writing CGI scripts.

2.4 World Wide Web

The World Wide Web, also termed as the Web, is a distributed system on the Internet that uses hypermedia to link resources on the Internet [26]. The Web, along with all the advantages of the client-server system, offers other advantages that have made it immensely popular [26]. It offers standardization of user interface since a browser is used to access resources on the Web. A high transaction volume is possible on the Web since it uses the HTTP protocol. Finally because the Web is based on the Internet it provides the ability to work across platforms and applications. With the client machines becoming more powerful and browsers now able to handle audio and video, along with text and graphics, the Web is well suited for multimedia applications [26].

Various client/server technologies used with the Web in this thesis work are as follows

2.4.1 Hyper Text Transfer Protocol (HTTP)

HTTP (HyperText Transport Protocol) is one of the most widely used protocols on the Internet. It is an application-layer, connectionless protocol communicating over the underlying TCP/IP connections on the Internet. The HTTP protocol is connectionless in the sense that once the server responds to the client's request, the connection between the client and the server is terminated [11].

2.4.2 Hyper Text Markup Language (HTML)

HTML (HyperText Markup Language) is a markup language designed specifically for making electronic documents for delivery over the Internet and for presentation on a variety of displays. HTML is the standard markup language used on WWW clients to create Web pages. HTML is based on SGML (Standard Generalized Markup Language) which is an international standard for defining device-independent methods of representing text in electronic form [11].

An HTML document is a plain text document containing instructions (also called HTML tags) along with the contents of the document. These instructions tell the browser about the formatting of the following text. HTML tags are enclosed within the "<" and ">" bracket pairs. HTML is a structured language in the sense that it has rules for the placement of tags [11]. HTML tags can optionally contain attributes followed by their value. HTML tags and the attributes are case-insensitive. HTML allows users to create hyper-links, fill-in-forms and clickable images. Hypertext links are used to link the current document with other resources on the Internet [13]. These resources are specified through a Uniform Resource Locator (URL) that is included in the HTML tags [13].

When a user clicks these links the browser searches for the resource specified by the HTML link and fetches the document at that specified location or downloads a file using anonymous FTP etc.

2.4.3 Common Gateway Interface (CGI)

The Common Gateway Interface (CGI) is a standard protocol for communication between the HTTP servers and the server-side gateway programs [9]. CGI specifications define how the data is being passed from the server to the gateway program and back. CGI is used in PEET to pass the information selected on the HTML forms to gateway programs. These gateway programs pass this data to the programs accessing Oracle data, based on the user input.

CGI uses environment variables and standard input streams to pass the information to the gateway program, which are set when a client requests services from the server [9].

Commonly used environment variables in this project are:

CONTENT_TYPE: This variable contains the MIME type of the information passed by the HTML form in case PUT or POST method is used to transfer information.

CONTENT_LENGTH: In case CONTENT_TYPE contains valid information, this variable contains the length of the data being passed otherwise it is left blank.

REQUEST_METHOD: The method associated with the request. HTTP protocol uses GET, HEAD, POST or PUT method.

PATH_INFO: Contains any extra path information in addition to the absolute document path of the actual document being invoked.

SCRIPT_NAME: Contains the path and the name of the script being accessed as referenced in the URL.

QUERY_STRING: The query string that follows the question mark in the URL. The string is encoded while transferring it to the server.

Apart from setting these environment variables, the contents of the form are sent to the standard input stream when using the POST method. This data is encoded using the same encoding scheme used for encoding the URL information while using the GET method.

PEET uses both the GET and POST method to pass the information from input form to the Web server and back.

2.4.4 JavaScript

JavaScript is a scripting language introduced by Netscape Communications [7]. Unlike Java applets that exist independently in HTML documents, JavaScript code can be directly integrated into HTML documents. This provides a tight integration between the HTML code and JavaScript. JavaScript is used to respond to various user events, the current environment and previous surfing history. It can be used to validate form elements to ensure that the data that is sent back to the server is in the form expected by the server side gateway programs and they don't have to do any further validations. In fact, JavaScript is used to replace many CGI scripts, thus reducing the server load and the network traffic between the client and the server. JavaScript is used in this thesis work to perform validations of form elements and to create dynamic URL's depending on the user's choices in the forms.

2.5 Java Programming Language

Java is an object-oriented language introduced by Sun Microsystems in 1991 [8]. A Java compiler converts Java code into architecturally independent byte codes. These byte codes are stored on the server as a “.class” file. When an HTML document refers to the Java applet by using the <APPLET> tag, a Java-enabled browser downloads these byte codes onto the local machine along with the HTML document. The browser then verifies the integrity of the byte code and runs it by interpreting the byte code sequentially [8]. Thus, by implementing a Java-enabled browser, architecturally dissimilar operating systems like Windows 95, Windows NT, Sun Solaris and MacOS can execute the same copy of the Java byte codes making Java virtually machine independent [8].

Though Java is run locally on the client machine, it is a very secure language. This is achieved by not allowing the applet to do certain things like accessing or modifying local files or making arbitrary network connections [16]. Also the Java language does not implement pointers, thus making it impossible for the Java programs to read or write the computer memory directly. All these security features ensure that the downloaded applets do not create any havoc on the local machine on which it is downloaded or the network on which the client machine resides [16].

In PEET, Java is used to create client-side graphs displaying the economic impact and the ground water hazard due to pest management methods used.

3. Design

This chapter deals with the design issues of the PEET system. The chapter starts with the overall design of the system and then includes design of the specific modules of the system. The modules include the front-end design, interface design between client and server, the database schema design and finally the design of PEET output.

3.1 Overall System Design

PEET is a web-based decision support system. The system uses popular browsers as a front end to provide interaction between the system and the user. User inputs are accepted using HTML forms. These inputs are then passed on to the web server using Internet as medium for data transfer. The web server then uses CGI protocol to relate these user inputs to programs that will query the PEET database and perform calculations. Finally CGI protocols are used to transfer the results of the calculations back to the user. The output is displayed in the browser in various formats. The flow of data from the client to the server and back can be depicted shown in Figure 2.

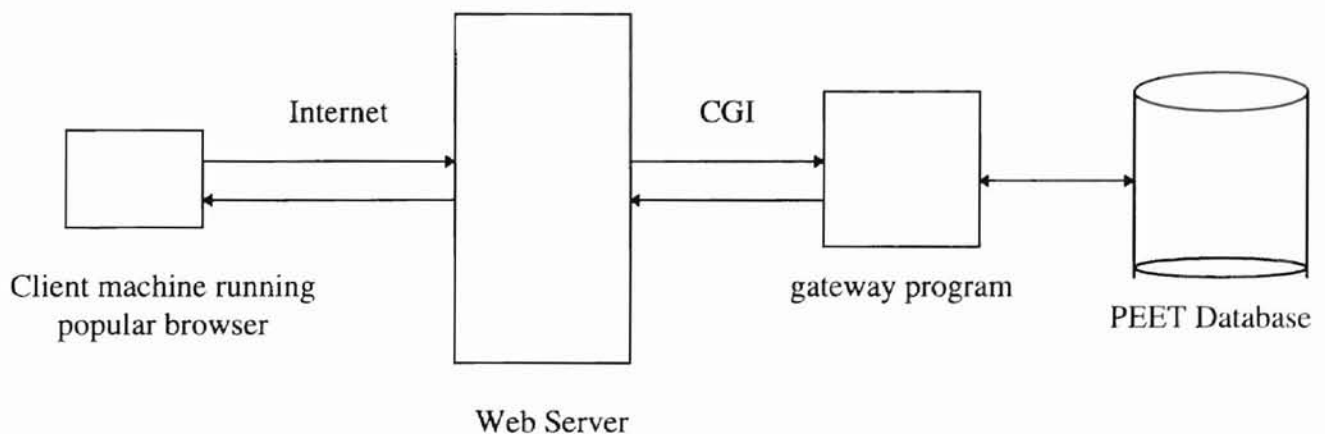


Figure 2. Flow of data in PEET system.

3.2 User Interface Design

User-interface design deals with designing HTML forms, which are easy to use and provide maximum interactivity with the user. Care is taken to perform maximum data validation at the client end so as to reduce the dependence on the server. The forms designed for accepting the user inputs are as follows:

3.2.1 Area of Interest

This form gathers information related to the area for which the data is to be simulated.

The inputs include Name of the Farmer, Name of the Field, Size of the Field and a select list to select the County in which the field is located.

The HTML form is shown in Figure 3.

Area of Interest	
Grower's Name	<input type="text"/>
Field Name	<input type="text"/>
Field Size	<input type="text"/> acre
Select County	ATOKA <input type="button" value="v"/>
<input type="button" value="Get Soils"/>	

Figure 3. HTML form to enter Area of Interest.

A default value for the county and the type of soil is set. In case the user selects a county other than the default, the current screen immediately invokes a new screen to select soil found in that particular county thus maintaining consistency.

3.2.2 Field Conditions

The form in Figure 4 has select lists to select the Tillage Type, Irrigation type and Radio buttons to select the soil moisture.

Field Conditions	
Select Tillage Type:	<input type="text" value="Straight Rows, Clean Tilled"/>
Select Irrigation Type:	<input type="text" value="None, Dryland Farming"/>
Select Soil Moisture:	<input checked="" type="radio"/> Normal <input type="radio"/> Dry

Figure 4. HTML form to enter Field Conditions.

3.2.3 Typical Yields

The form in Figure 5 displays the lowest, normal and high weed free yields for the field. Being text fields they are validated to contain valid numeric data. The initial values for these yields depend on Area of Interest and Soil Type.

Typical Yields		
Expected Weed-Free Yields		
Lowest	<input type="text" value="980"/>	LB/ACRE
Typical	<input type="text" value="1400"/>	LB/ACRE
High	<input type="text" value="1680"/>	LB/ACRE

Figure 5. HTML form to enter Typical Yield values.

3.2.4 Market Value

The user can enter the market value of the yield per ton in the form shown in Figure 6.

Market Value	
Market Price	<input type="text" value="\$ 680.00"/> /ton (peanuts)

Figure 6. HTML form to enter Market Value of yield.

3.2.5 Application Type

Form in Figure 7 gathers information regarding the type of application, which indicates when the pesticide is applied to the field. It also has text fields to enter the cost of the application per acre and Scouting cost per acre for the weeds.

Application Type / Costs	
Select Application Type:	Pre-Plant Incorporated ▾
Application Cost	4.00 /acre (one pass)
Scouting Cost	0.00 /acre (weeds only)

Figure 7. HTML form to select the Type of Application.

3.2.6 Weed Densities

Depending on the type of the application, the form in Figure 8 lets the user select the density of the weeds in the field. Weeds are fetched from the PEET database.

Enter Weed Densities							
Weed Name	Density			Weed Name	Density		
	none	high	low		none	high	low
Barnyardgrass	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Bermudagrass	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Buffalobur	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Carpetweed	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Cocklebur, Common	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Copperleaf, Hophornbeam	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Crabgrass, Large	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Crotons	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Crownbeard	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Eclipta	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Eveningprimrose, Cutleaf	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Horsenettle	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Johnsongrass	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Lambsquarters, Common	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Mallow, Venice	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Milkweed, Honeyvine	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Morningglories	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Nightshade, Silverleaf	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Nutsedge, Yellow	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Panicum, Texas	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Pigweeds	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Purslane, Common	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sicklepod	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Sida, Prickly	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Signalgrass, Broadleaf	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Spurges	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sunflowers	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	Trumpet creeper	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 8. HTML form to select Weed Densities for PPI and PE.

In case of Pre-Plant Incorporated and Pre-Emergence the user selects either low, high or no weed density. For the Post-Emergence type the number of weeds per sq. feet has to be entered. In the later case, information regarding the treatment date and size of weed is also needed. This is shown in Figure 9.

Treatment Date	15 May - 15 June		
Select Weed Size	<input checked="" type="radio"/> <2"	<input type="radio"/> 2" - 4"	<input type="radio"/> >4"

Enter Weed Densities
Enter the average number of weeds per 100 SqFt

Barnyardgrass	0	Bermudagrass	0
Buffalobur	0	Carpetweed	0
Cocklebur, Common	0	Copperleaf, Hophornbeam	0
Crabgrass, Large	0	Crotons	0
Crownbeard	0	Eclipta	0
Eveningprimrose, Cutleaf	0	Horsenettle	0
Johnsongrass	0	Lambsquarters, Common	0
Mallow, Venice	0	Milkweed, Honeyvine	0
Morningglories	0	Nightshade, Silverleaf	0
Nutsedge, Yellow	0	Panicum, Texas	0
Pigweeds	0	Purslane, Common	0
Sicklepod	0	Sida, Prickly	0
Signalgrass, Broadleaf	0	Spurges	0
Sunflowers	0	Trumpetcreeper	0

Figure 9. HTML form to Enter Weed Densities for Post Emergence.

3.2.7 Pesticide Costs

The screen in Figure 10 is used to enter the cost of various herbicides. The user is given an option to select from the choice of units in which the herbicide can be purchased.

JavaScript is used to do the data validation of all the text fields.

Enter Pesticides Costs		
Pesticide	Costs	Units
BALAN DF	82.23	12.5 LB
BALAN EC	36.55	2.5 GAL
BASAGRAN	157.50	2.5 GAL
BLAZER	56.10	GALLON
BUGLE	56.25	GALLON
BUTYRAC 175	70.00	2.5 GAL
BUTYRAC 200	73.08	2.5 GAL
DUAL	159.66	2.5 GAL
POAST PLUS	116.00	2.5 GAL
PROWL	78.26	2.5 GAL
PROWL 33EC	60.63	2.5 GAL
PURSUIT	570.00	GALLON
ROUNDUP	110.10	GALLON
SONALAN EC	75.70	2.5 GAL
STARFIRE	58.50	2.5 GAL
STORM	164.25	2.5 GAL
TREFLAN 5	34.64	40 LB
TREFLAN EC	72.25	2.5 GAL
TREFLAN MTF	75.00	2.5 GAL
TREFLAN TR-10	45.35	50 LB
TRILIN	76.92	2.5 GAL
VERNAM 7E	83.71	2.5 GAL

Figure 10. HTML form to enter Pesticide Costs.

3.2.8 Options

This window provides the options like selecting the probability at which the groundwater hazard can be exceeded, the treatment of ranks from Stochastic Dominance or Groundwater Hazard for the selected probability and the aquifer characteristics like the mixing depth of the pesticide in meters and the porosity percentage. The above two fields are validated by their respective JavaScript functions before passing them to the CGI gateway program. The screen looks as shown in Figure 11.

Options			
Aquifer Characteristics			
Mixing Depth	1.00 meters		
Porosity	25 %		
Display Groundwater Hazard Exceeded with a Probability of			
<input type="radio"/> 0.01	<input type="radio"/> 0.05	<input type="radio"/> 0.20	<input type="radio"/> 0.40
<input type="radio"/> 0.02	<input checked="" type="radio"/> 0.10	<input type="radio"/> 0.30	<input type="radio"/> 0.50
Rank Treatments Using			
<input checked="" type="radio"/> Groundwater Hazard at selected Probability			
<input type="radio"/> Stochastic Dominance			

Figure 11. HTML form to enter Various System Options.

Frames are used to design the user-input screen. The left frame consists of links to the individual input screens and a button to invoke display of the output. The user input screen is shown in Figure 12.

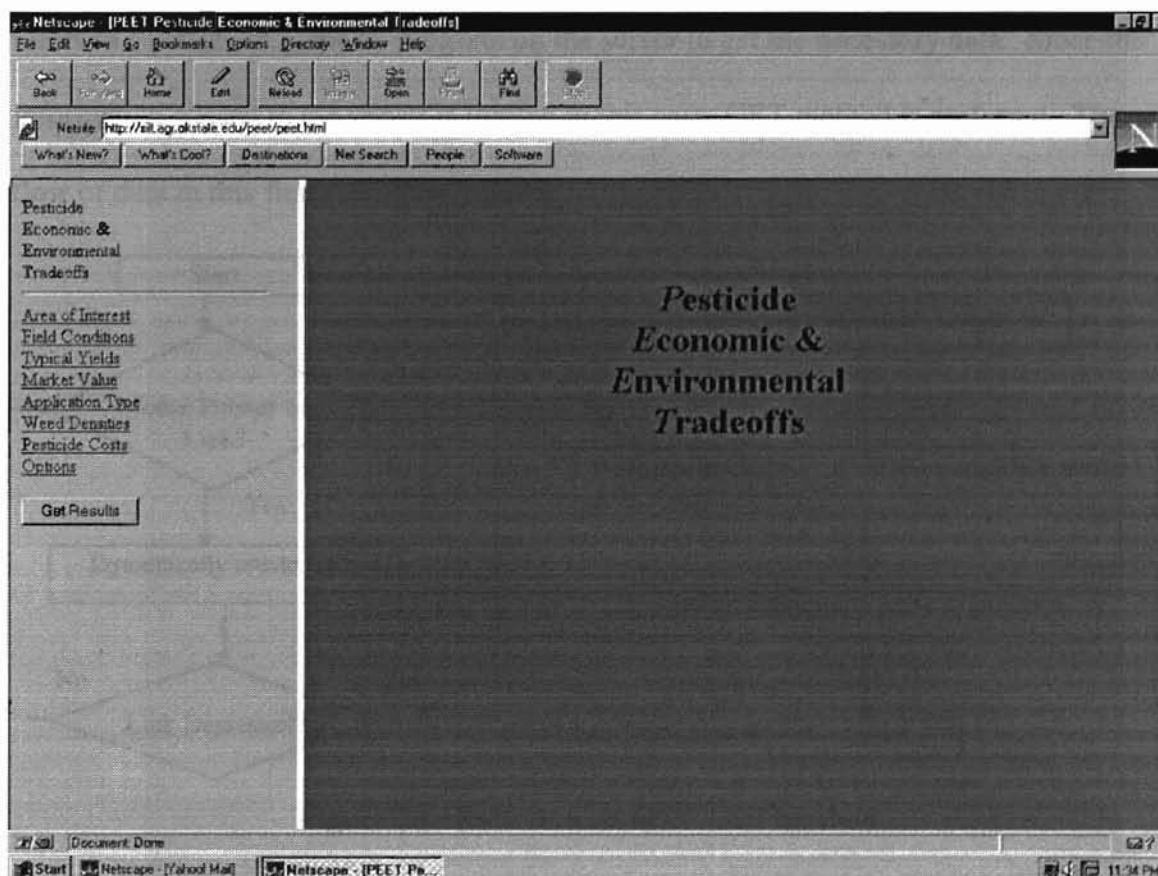


Figure 12. PEET Entry Screen.

3.3 CGI Interface Design

The client requests data from the server in two cases. In the first case data is requested from the server while accepting the user inputs. This condition typically arises when selecting a soil type in a particular county, getting default yield for a particular county and obtaining type of soil and irrigation type. In the second case, the client requests service from the server when the user presses the “Get Output” button to display the final result.

3.3.1 Intermediate Server Calls

In the first case the GET method of data posting from client to server is used. JavaScript is used to dynamically create the Uniform Resource Locator (URL) appended with necessary arguments like the name of the county. Once the link is depressed, the client refers to a particular gateway program on the server to get the necessary data. Since the information passed to the server is limited in this case, GET method is preferred. The flow of data in this first case is as follows:

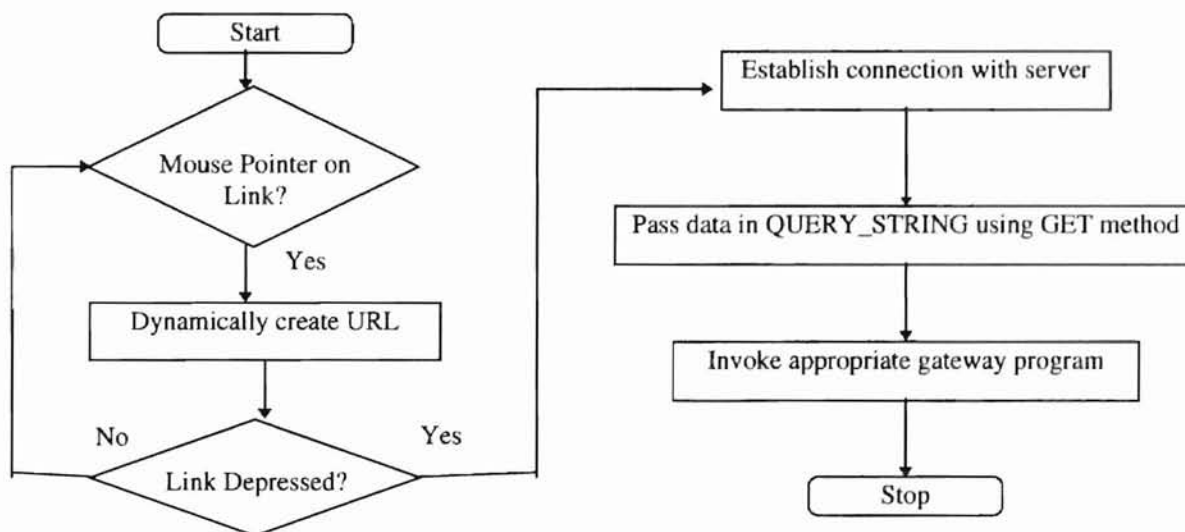


Figure 13. Data flow in GET CGI method.

3.3.2 Final Server Call

When the user has entered all the necessary information for a query, the “Get Output” button is used to display the results of the query. In this case, the form is submitted to the server using the POST method. Since the GET method has a limitation of only 255 characters, POST method is preferred in this case. The form refers to a particular gateway program on the sever in its “ACTION” attribute. This gateway program is invoked upon posting of the form. This program is responsible for reading all the input submitted to the sever, and invoking a program to query the PEET database. The flow of data from client to sever in this case is as follows:

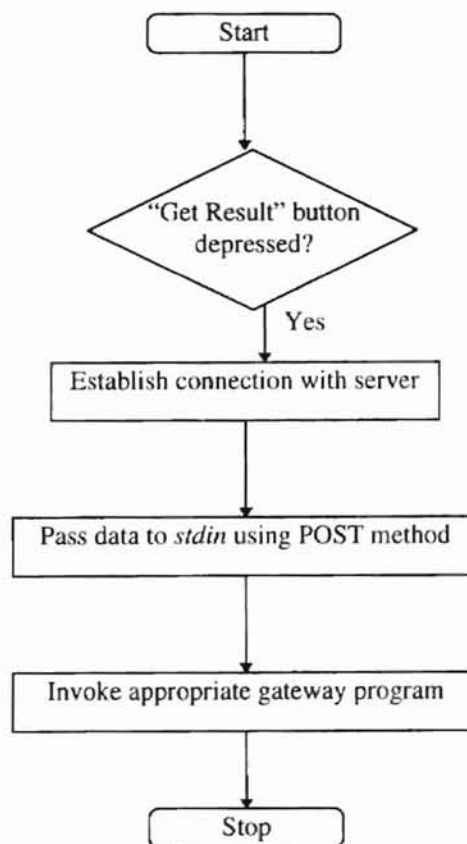


Figure 14. Data flow in POST CGI method.

3.4 Database Design

3.4.1 Database Schema

The data collected for the PEET project was analyzed and then database normalization techniques were used to design the database schema so as to avoid any data replication and ensure data integrity.

3.4.2 Project Tables

The tables created for the project can be divided between the master tables, tables containing the simulation data and other tables used to store general information for calculation purpose. Master tables contain attributes of the entities used in the project. They include the information regarding Oklahoma counties, various types of soil and weeds found in these counties and commonly applied herbicides for pest management. The Simulation Data tables contain the information collected for the purpose of analysis and calculations. Care is taken so as to maintain the consistency between the master tables and simulation data tables. Other tables used for the project contain information regarding the various units and parameters used in the calculations so that they are not hard-coded in the programs and provide for easy updates in the future.

The **Master Tables** created for the project are:

1. County: This table is the master table used to store the Oklahoma counties. The table is indexed using unique County Index. The table structure is given below in Table 1.

Table 1. Attributes of COUNTY table.

CountyName	string(12)
CountyIndex	number(6)

2. Soilname: This table is the master table used to store the soil data for the project. Soils are stored by county. The table is indexed using the unique Soil Index. The table structure is given below in Table 2.

Table 2. Attributes of the SOILNAME Table.

CountyIndex	number(6)
SoilIndex	number(6)
Soilname	string(46)

3. Herb: This table contains information regarding the herbicides commonly used by the farmers. The information stored along with the herbicide names are the amounts in which the herbicide is commonly applied and their units. The table is indexed on the unique Treatment Index field. The table structure is shown below in Table 3.

Table 3. Attributes of the HERB table.

TreatmentIndex	number(6)
TradeName	string(46)
ApplAmt	number(6)
ApplUnits	string(7)

4. Weed: This table contains information regarding the weed commonly infecting the Oklahoma fields. The table contains information regarding their competitive index, the low density and high density of the weed and the unique weed code on which the weeds are indexed. The table structure is shown in Table 4.

Table 4. Attributes of WEED table.

Weedname	string(29)
Weedcode	number(10)
CompetitiveIndex	number(6)
Lowdensity	number(6)
Highdensity	number(6)

The **Simulation Data Tables** created for the project are as follows:

1. Cost: Table structure shown in Table 5 stores the default cost of the herbicide per unit for every herbicide used in the system.

Table 5. Attributes of COST table.

TradeName	string(50)
PurchasePrice	number(6)
PurchaseUnit	string(50)

2. Gwhazard: The table shown in Table 6 is the main table used to compute the ground water hazard due to the pesticides. The table is organized with Irrigation type, Tillage Code, County Index, Soil Index and Treatment index as the primary key. The HoHo values indicate the probability of ground water hazard at various percentages.

Table 6. Attributes of GWHAZARD table.

IrrigationType	string(1)
TillageCode	number(4)
CountyIndex	number(4)
SoilIndex	number(4)
TreatmentIndex	number(4)
HOHO1	Number
HOHO2	Number
HOHO5	Number
HOHO10	Number
HOHO20	Number
HOHO29	Number
HOHO40	Number
HOHO50	Number
Rank	number(4)

3. Efficacy: This table stores the efficacy of every weed affecting the yield depending on the moisture in the field and the size of the weed. The table is indexed using a unique efficacy index and the weed code. The table structure is shown in Table 7.

Table 7. Attributes of EFFICACY Table.

EfficacyIndex	Number
WeedCode	Number
MoistureLevel	String(1)
WeedSize	Number
Efficacy	Number

The other tables used in the system are as follows:

1. Param: Table structure shown in table 8 stores the parameters commonly used in the calculations. They include the values of maximum loss of yield, application cost, soil moisture, various application types etc.

Table 8. Attributes of PARAM table.

Name	string(20)
Abbreviation	string(10)
Description	string(29)
Value	number(6)
OrderNo	number(6)

2. Unit: Table shown below in Table 9 is used to facilitate the conversion of values using different units in the system to the standard units. The table provides a conversion ratio between various units so that they are not hard coded in the programs and others can be added without changing the code.

Table 9. Attributes of UNIT table.

UnitType	string(1)
OldUnit	string(7)
StdUnit	string(7)
Conversion	number(6)

3. Treatmnt: Table shown in Table 10 stores the various treatments of herbicides commonly used depending on the efficacy needed. The various treatments are identified using start day for the treatment, the end day and the depth of the treatment used. The table is indexed using the unique treatment index.

Table 10. Attributes of TREATMNT table.

TreatmentIndex	number
ApplType	number
ApplDayBegin	string(1)
ApplDayEnd	number
ApplDepth	number
EfficacyIndex	number

4. Yield: Table shown in Table 11 stores the values of low yield, normal yield and the high yield of the crop for every irrigation type for soil type in each county in Oklahoma.

Table 11. Attributes of YIELD table.

CountyIndex	number
SoilIndex	number
IrrigationType	string(1)
LowYield	number
NormalYield	number
HighYield	number
YieldUnits	string(7)
AreaUnits	string(7)

3.5 PEET Output Design

PEET system output consists of a bar graph to display the economic impact and ground water hazard due to the pest management techniques applied. A graph to denote the tradeoff between these two parameters is also included. Finally this graphical data is also displayed in text format using HTML tables.

The bar graph is created either on the client side using Java applet or on the server side. The side graph is created using C and graphics libraries. The decision on how to create the bar graph is made at runtime. JavaScript functions determine whether the client browser is Java-enabled or not. In case of Java-enabled browser, the PRO*C programs create a parameter list to be passed to the applet and create an HTML output document

with an <APPLET> tag in the code. In case of a Java-disabled browser, the server will call graphics functions provided with the libraries to generate a GIF file. This file is stored in a temporary directory and referenced using an tag in the output HTML document.

The graph showing the tradeoff between the economic impact and ground water hazard is created on the server using graphics libraries and referenced in the output HTML document. Details for creating server-side or client-side bar graphs are included in the next chapter.

4. Implementation

This chapter deals with the implementation details of the PEET system. The chapter includes the details of techniques used for creating and managing the Oracle database created for the PEET system. Further it includes the implementation of the interface between the client and the Web server. This is followed by details of data retrieval from the Oracle database. Finally the implementation of server-side graph generation and Java applet for client-side graph generation are included.

4.1 Oracle Data Management

4.1.1 Creating Tablespaces

Oracle stores table data and other database management related data in data files. A *tablespace* is a group of disk files containing logically related database objects. For the above project, an independent tablespace called PEET is created using the following SQL code.

```
CREATE TABLESPACE PEET  
DATAFILE '/home/oracle/orahome/dbs/peet.dbf' SIZE 50M ONLINE
```

4.1.2 Creating Users

Users are the Oracle accounts that can have access to the Oracle objects in a particular tablespace as defined while creation of the user. For the PEET project we create two users: one as a manager account and another as a general user account. The manager account user has privileges to create, delete, access and update any objects in the PEET

tablespace whereas the user account has only a SELECT privilege for the objects in the PEET table space. These two users are created using the following SQL statements:

```
CREATE USER PEETMGR IDENTIFIED BY PASSWORD
DEFAULT TABLESPACE 'PEET' TEMPORARY TABLESPACE 'PEET'
PROFILE 'DEFAULT';
GRANT CONNECT, RESOURCE TO PEETMGR WITH ADMIN OPTION;
GRANT CREATE USER TO PEETMGR;
GRANT DROP USER TO PEETMGR;
```

```
CREATE USER PEETUSR IDENTIFIED BY PASSWORD
DEFAULT TABLESPACE 'PEET' TEMPORARY TABLESPACE 'PEET'
PROFILE 'DEFAULT';
GRANT CONNECT TO PEETUSR
```

4.1.3 Creating Tables

Oracle stores the data in the form of tables. The tables for the PEET project are created using SQL scripts. An example is shown below.

```
CREATE TABLE GwHazard (
IrrigationType      VARCHAR2(1) NOT NULL,
TillageCode         NUMBER(4)   NOT NULL,
CountyIndex         NUMBER(4)   NOT NULL,
SoilIndex           NUMBER(4)   NOT NULL,
TreatmentIndex      NUMBER(4)   NOT NULL,
HoHo01              NUMBER,
HoHo02              NUMBER,
HoHo05              NUMBER,
HoHo10              NUMBER,
HoHo20              NUMBER,
HoHo29              NUMBER,
HoHo40              NUMBER,
HoHo50              NUMBER,
Rank                NUMBER(4),
PRIMARY KEY (
IrrigationType,
TillageCode,
CountyIndex,
SoilIndex,
TreatmentIndex));
```

The name of the table to be created is followed by the key words CREATE TABLE. The columns in the table are specified with the name of the column, the data type of the

column along with the maximum precision for the column and the keyword NULL or NOT NULL, indicating whether that particular column is allowed to hold a null value or not.

The PRIMARY KEY keyword specifies which columns in the table uniquely identify a particular row in the table. Even if a table does not have a primary key, an index can be created on the table so as to make the data retrieval from the table faster.

Storage parameters for storing the table data can also be specified during creation of the table but they are ignored in the above example, in which case Oracle uses its default storage characteristics.

4.1.4 Loading Data into Tables

Data needed for the project was available in flat text files. The fields in the text file are comma-separated with character fields enclosed with quote marks. The sample data for the Chemical table is as follows:

```
"1,3-DICHLOROPROPANE",955.0000,700.0000,-95.0000
"1,3-DICHLOROPROPENE",32.0000,10.0000,0.2000
"1-NAPHTHALENE ACETAMIDE",100.0000,10.0000,-95.0000
"2,4,5-T ACID",80.0000,24.0000,70.0000
"2,4,5-T AMINE SALTS",80.0000,24.0000,70.0000
"2,4,5-T ESTERS",1000.0000,24.0000,70.0000
"2,4-D ACID",20.0000,10.0000,70.0000
"2,4-D DIMETHYLAMINE SALT",20.0000,10.0000,70.0000
"2,4-D ESTERS OR OIL-SOL.AMINES",100.0000,10.0000,70.0000
"2,4-DB BUTOXYETHYL ESTER",500.0000,7.0000,70.0000
"2,4-DB DIMETHYLAMINE SALT",20.0000,10.0000,70.0000
"3-CPA SODIUM SALT",20.0000,10.0000,-95.0000
"ABEMECTIN(AVERMECTIN)",4760.0000,28.0000,3.0000
"ACEPHATE",2.0000,3.0000,29.0000
"ACIFLUORFEN SODIUM SALT",113.0000,14.0000,1.0000
```

The flat file data is loaded into Oracle tables using the Oracle SQL*Loader utility. The SQL*Loader requires two inputs: the external data in form of disk files and the control

information in the form of a control file, which describes the nature of the input data and the table and the column information in which to load the data.

The control information needed to load the above data in the Chemod table is as follows:

```
LOAD DATA
INFILE chemod.txt
INSERT
INTO TABLE PEETMGR.CHEMOD
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
TRAILING NULLCOLS
(CommonName, Koc, HalfLife, HALEQ)
```

4.2 CGI Interface

CGI interface to the project to pass information from the client to the server and back is written using the Bourne shell programming and C programs. The CGI program starts with setting the environment for all the programs invoked by the CGI script, executes a C program to retrieve the information passed from the HTML form either as environment variables or input stream depending on the method used. These name-value pairs, which are retrieved by the C code, are used to set environment variables and then the appropriate PRO*C program is invoked to perform data retrieval from the Oracle database. The data returned by the PRO*C program is then passed back to the client.

The algorithm for a general CGI script is as follows:

- A. Determine transfer method of the data received to the web server.
- B. If GET method used go to step E.
- C. If POST method is used, read *stdin* (standard input) stream.
- D. Go to step G.
- E. Determine the length of data input from the environment variable QUERY_LENGTH.
- F. Read the environment variable QUERY_STRING.
- G. Decode the input to extract name-value pairs.
- H. Store extracted data in temporary files or environment variables.

4.3 Oracle Interface

Interface to Oracle is achieved by using SQL embedded C (PRO*C) programs. These programs are used both to provide select list data in the user input screens as well as fetching data from the server for final calculations of economic impact of pesticides on yield and potential groundwater hazard due to the same. An algorithm to extract data from Oracle RDBMS using the PRO*C is as follows:

- A. Get input parameters from the environment.
- B. Connect to Oracle RDBMS user account.
- C. Use the input parameters to dynamically create SQL query in the form of Oracle *cursor*.
- D. Use *cursor fetch* to extract data from Oracle tables.
- E. Perform calculations to determine ground water hazard and economic impact.
- F. Create temporary HTML file to contain results of calculations.
- G. Disconnect from Oracle.

Error checking is done at every step of Oracle interface. The list of PRO*C programs created and their usage is summarized in table 12.

Table 12. PRO*C programs created for PEET.

PRO*C Program	Usage
ppi.pc	Compute GWH and Economic impact for the Pre-Plant Incorporated Application type.
pre.pc	Compute GWH and Economic impact for the Pre-Emergence Application type.
pot.pc	Compute GWH and Economic impact for the Post-Emergence Application type.
GetSoilName.pc	Get soil-types for the county selected.
GetYield.pc	Get typical yields for the county, soil-type and field conditions selected.
GetCost.pc	Get typical pesticide costs for the various pesticides being used.

4.4 Creating Graphics

Graphs are created in PEET as bar graphs and can be created on the server and then passed on to the client in the output HTML document using the tag or can be created on the client side using the Java applet.

4.4.1 Server Side Graphics Creation

Server side graphs are created using C programs and graphics libraries which provide functions to first define an image in the memory, tools to draw common graphic objects like lines, rectangles, circles, include strings and fill rectangles in the image and then convert the memory into a disk file with one or more popular graphic formats available. “gd” is such a library available freely on the Internet, and is used to generate graphs. The graphs are stored as temporary GIF files and are referenced by the HTML documents in the output. The function used to create the image and the graphic objects are as follows:

gdImageLine(): This function is used to add a line to the graphics image. The arguments to the function are pointer to the image, the two coordinates of the line and the color of the line to be added to the image.

gdImageString(): This function is used to add a string to the graphics image. The arguments to the function are the image pointer, the initial x and y location, the string to be printed and the color of the string to be used.

gdImageCreate(): This function creates an image in the memory and returns the pointer to the image. It creates the image with a rectangular area with lengths of the sides passed to the function as parameters.

gdImageGif(): Converts the image pointed by the image pointer to a GIF file. The GIF file pointer is passed to the function along with the image pointer.

gdImageDestroy(): This function destroys the image pointed by the image pointer and frees the memory occupied by the image.

Figure 15 shows the bar graph created on the server side for the PEET system using functions provided by the library. The decision to create the graph on the server side depends on the type of the browser the client system is using. If the browser is a non-Java browser or the client system is 16-bit, JavaScript methods used in the PEET input screen, inform the server about the browser type during form submission. Depending on this information, the graph is created on the server-side or Java applet is used.

Herbicides	Potential Loss due to weeds			RANK	Potential Groundwater Hazard			
	Yield lb / acre				BETTER	10%	100%	WORSE
	980	1400	1680		1%	10%	100%	1000%
NO TREATMENT VALUES	(\$ / acre)			NA				
Pre-Plant Incorporated	0	0	0					
BALAN EC 3.00 QT	15	15	15	1				<1
BALAN DF 2.00 LB	17	17	17	1				<1
BALAN EC 4.00 QT	19	19	19	1				<1
BALAN DF 2.50 LB	20	20	20	1				<1
TREFLAN 5 0.80 PT	15	15	15	1				<1
TREFLAN EC 1.00 PT	8	8	8	1				<1
TREFLAN MTF 1.00 PT	8	8	8	1				<1
TREFLAN TR-10 5.00 LB	9	9	9	1				<1
TRILIN 1.00 PT	8	8	8	1				<1
SONALAN EC 1.50 PT	10	10	10	1				<1
SONALAN EC 2.00 PT	12	12	12	1				<1
SONALAN EC 2.50 PT	13	13	13	1				<1
SONALAN EC 3.00 PT	15	15	15	1				<1
PURSUIT 4.00 FL OZ	22	22	22	1				<1
BALAN EC 3.00 QT	25	25	25	21				<1
+VERNAM 7E 2.33 PT								
BALAN DF 2.00 LB	27	27	27	21				<1
+VERNAM 7E 2.33 PT								
VERNAM 7E 2.33 PT	14	14	14	21				<1
VERNAM 7E 3.50 PT	19	19	19	24				<1
TRILIN 1.00 PT	18	18	18	25				<1
+VERNAM 7E 2.33 PT								
TREFLAN 5 0.80 PT	25	25	25	25				<1
+VERNAM 7E 2.33 PT								
TREFLAN EC 1.00 PT	17	17	17	25				<1
+VERNAM 7E 2.33 PT								
TREFLAN MTF 1.00 PT	18	18	18	25				<1
+VERNAM 7E 2.33 PT								
PROWL 1.00 PT	18	18	18	38				<1
+VERNAM 7E 2.33 PT								
PROWL 3.3EC 1.20 PT	17	17	17	38				<1
+VERNAM 7E 2.33 PT								
PROWL 3.3EC 1.20 PT	8	8	8	47				17
PROWL 1.00 PT	8	8	8	48				17
PROWL 3.3EC 1.80 PT	9	9	9	49				25
PROWL 1.50 PT	10	10	10	50				25
PROWL 3.3EC 2.40 PT	11	11	11	51				33
PROWL 2.00 PT	12	12	12	52				34
DUAL 1.50 PT	16	16	16	55				149
DUAL 2.00 PT	20	20	20	57				198
Pre-Emergence								
PURSUIT 4.00 FL OZ	22	22	22	1				<1
DUAL 1.50 PT	16	16	16	54				147
DUAL 2.00 PT	20	20	20	56				196
Spot								
ROUNDUP 2.00 QT	59	59	59	1				<1
ROUNDUP 3.00 QT	87	87	87	1				<1
Post-Emergence								
BUGLE 0.60 PT	8	8	8	1				<1
BUGLE 1.20 PT	12	12	12	1				<1
PURSUIT 4.00 FL OZ	22	22	22	1				<1
BLAZER 0.50 PT	8	8	8	29				<1
BLAZER 1.00 PT	11	11	11	30				<1
BLAZER 1.50 PT	15	15	15	31				<1
BLAZER 1.50 PT	18	18	18	32				<1
+BUTYRAC 200 1.00 PT								
BLAZER 2.00 PT	22	22	22	32				<1
+BUTYRAC 200 1.00 PT								
BUTYRAC 175 1.80 PT	10	10	10	34				<1
BUTYRAC 200 1.60 PT	10	10	10	35				<1
POAST PLUS 36.00 FL OZ	17	17	17	36				<1
POAST PLUS 48.00 FL OZ	21	21	21	36				<1
BUTYRAC 200 16.00 FL O	20	20	20	40				<1
+STORM 1.50 PT								
STORM 1.50 PT	16	16	16	40				<1
BUTYRAC 200 8.00 FL OZ	18	18	18	40				<1
+STORM 1.50 PT								
BASAGRAN 1.50 PT	18	18	18	43				<1
+BUTYRAC 200 8.00 FL OZ								
BASAGRAN 1.50 PT	16	16	16	43				<1
BASAGRAN 2.00 PT	22	22	22	45				<1
+BUTYRAC 200 8.00 FL OZ								
BASAGRAN 2.00 PT	20	20	20	45				<1
STARFIRE 11.00 FL OZ	6	6	6	53				51

Figure 15. PEET output graph created on the Server.

The graph to display the tradeoff between the economical impact of pests and the ground water hazard due to pest management techniques in the average case is displayed using a graph created on the sever side. The graph created is shown in Figure 16.

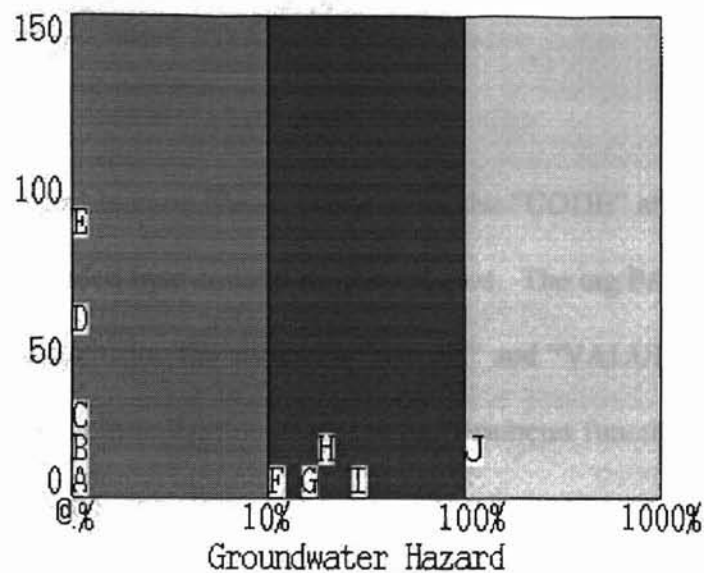


Figure 16. Graph to show economic impact and groundwater hazard tradeoff.

4.4.2 Client Side Graphics Creation

Client-side graphs are created using an applet written in the Java language. The applet `printGraph.java` is used to create the bar graphs on the client machine. The applet is transferred to the client machine along with the output and a parameter list. The list includes values for the economic impact and the groundwater hazard due to every pesticide used in the field. The sample HTML script containing the applet tag and the parameter list is as follows:

```
<html>
<head>
<title>PEET</title>
</head>
<body>
<center>
<h2>Cost and Potential Groundwater Hazard by Herbicide</h2>
<applet code=peetGraph.class width=610 height=290>
<PARAM NAME="SOIL0" VALUE="NO TREATEMENT VALUES">
<PARAM NAME="LOW0" VALUE="0">
```

```

<PARAM NAME="TYPICAL0" VALUE="0">
<PARAM NAME="HIGH0" VALUE="0">
...
...
<PARAM NAME="TYPICAL75" VALUE="6">
<PARAM NAME="HIGH75" VALUE="6">
<PARAM NAME="LOSS75" VALUE="51">
<PARAM NAME="TOTALSOILS" VALUE="76">
<h1> Your Browser does not support JAVA </h1>
</applet>
</center>
</body>
</html>

```

HTML tag applet refers to class file specified using the "CODE" attribute. This is essentially a pre-compiled byte code of the Java applet. The tag PARAM is used to pass parameters to the applet using the attributes "NAME" and "VALUE". These parameters are read by the applet at initialization using the getParameter function. The class peetGraph contains following class.

class peetGraph: This is the entry point for the applet. This class reads the parameters passed to the applet and stores them in arrays. It defines and initializes objects for the graph class, scrollbars and check boxes. It also includes an event-handler to switch the economic tradeoff graph between normal yield, high yield and low yield values. Tables 13 and 14 list the variables, objects and methods of the peetGraph class.

Variables:

Table 13. Variables in applet peetGraph.

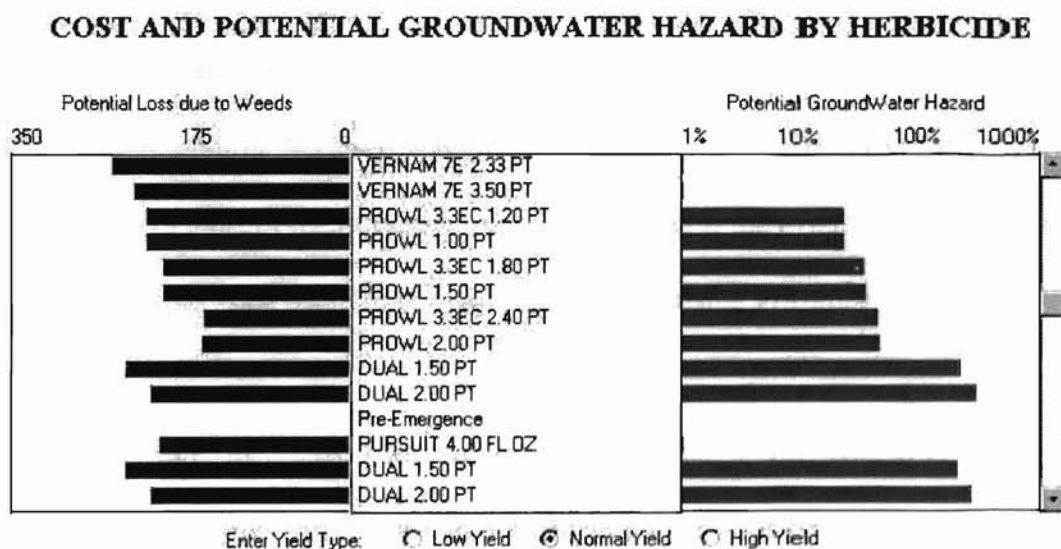
Variable name	Type	Comments
chemName	String[]	Stores the pesticides used in the field.
normalYield	int[]	Stores the values of normal yields for a specific pesticide
highYield	int[]	Stores the values of normal yields for a specific pesticide
lowYield	int[]	Stores the values of normal yields for a specific pesticide
hazard	int[]	Stores the groundwater hazard due to the pesticide
maximum	int	Stores the maximum economic impact due to the pesticides.
maxHazard	int	Stores the maximum groundwater hazard due to pesticides.

Objects:

Table 14. Objects in applet peetGraph.

Object Name	Type	Comments
soilCanvas	newCanvas	Defines a canvas to display the pesticides
barCanvas	graphCanvas	Defines a canvas to display the horizontal bars indicating economic impact due to pesticides.
hzbars	graphCanvas	Defines a canvas to display the horizontal bars indicating economic impact due to pesticides.
key	keyCanvas	Defines a canvas to display the checkbox
slider	Scrollbar	Defines the scrollbar
box1, box2, box3	Checkbox	Radio buttons to select between the normal, high and low yield values.

The graph created on the server side is shown in Figure 17.

**Figure 17. Java Applet to display PEET graph output.**

5. Conclusion & Future Work

5.1 Conclusion

The Pesticide Economic & Environmental Tradeoff, PEET, system is a very helpful system for farmers and EPA agents. It provides users with information regarding which pesticides to use for pest management to maximize their yield without harming the environment. Since the system is available on the WWW, it can be accessed from anywhere regardless of geographic location. The only system that is required is a client machine with Internet connection and a browser supporting graphics and/or Java. These browsers are popularly available at minimal cost or even no cost.

The PEET system requires Netscape Navigator 2.0 or above or Internet Explorer 3.0 or above. This covers most of the browsers used today. Even if the client system does not support Java, care has been taken to display PEET output using graphics created at server-side.

Since Java and JavaScript are used for the user interface design, the system is very interactive and easy to use. As the system uses client power for most of the data validations, it is fast and requires minimum bandwidth.

Use of a powerful RDBMS system like Oracle eases the task of storing, accessing and updating data. Use of Oracle allows the use of powerful tools provided by Oracle for better data management.

Care has been taken to minimize coding constants and parameters used for calculations directly into the programs. Instead, parameter tables are used to access these variables from the database. Most of the HTML input forms are also generated dynamically. Care

has been taken to eliminate any hard coding in HTML forms. Hence the project is very portable and can be expanded easily. For example, data for states other than Oklahoma can be collected and inserted into Oracle tables to make the PEET system available for those states. None of the PRO*C code or the HTML code need to be modified or recompiled.

5.2 Future Work

Though PEET is a very fast and a powerful decision support system, following improvements and future work are suggested to further enhance the system. These improvements are suggested in the following areas:

5.2.1 Decision Support System

Currently the analysis is made for one particular soil type. This is impractical sometimes. The system can be enhanced to perform the analysis on more than one soil type.

5.2.2 User Interface Design

Currently HTML forms with select fields, text fields and radio buttons are used to select the area of interest and the soil type. This can be replaced by a system which allows the user to select the area of interest using a series of arial photographs and finally selecting the field boundary he/she is interested in. The system would then determine the soil type(s) in the marked field and run the analysis algorithm for those soil types and produce the necessary output.

5.2.3 Customization

Currently the user is allowed to select from a list of commonly used pesticides for pest management. The weeds affecting the yield are also the ones regularly found in the area. The user can be given a choice of entering a pesticide or weed other than those provided by the system. The system would then be made to provide analysis for these customized user inputs.

5.2.4 Security

Though the current application is quite robust and secure from network hackers, it relies on the security provided by the UNIX system and Oracle Database Security features to protect its data. A firewall can be implemented to further enhance the security of the system.

References

- [1] Ben, Adido. "Securing the Web", IEEE Internet Computing, July-August 1997
- [2] Mirosav, Benda. "Applications on the Global Computer", IEEE Internet Computing, May-June 1997
- [3] Alex, Berson. "Client/Server Architecture", The McGraw Hill Companies, Inc., USA 1995
- [4] Ulysses, Black. "TCP/IP and Related Protocols", 2nd Edition, McGraw Hill, Inc., USA 1995
- [5] Steve, Bobrowski. "Mastering Oracle 7 & Client/Server Computing", 2nd edition, Sybex Inc., Alameda, CA 1995
- [6] Martin, Colby and David Jackson. "Special Edition Using SGML", Que Corporation, USA 1996
- [7] Arman, Danesh. "JavaScript 1.1 Developer's Guide", 1st edition, Sams.net Publishing, Indianapolis, IN 1996
- [8] Deital, Paul and Harvey Deital. "Java How to Program", Prentice Hall, Inc., Upper Saddle River, NJ 1995
- [9] Mark, Felton. "CGI: Internet Programming with C++ and C", Prentice Hall, Upper Saddle River, NJ 1995
- [10] Jerry, Fitzgerald. and Alan Dennis. "Business Data Communications and Networking", 5th Edition, John Wiley & Sons Inc., USA 1995
- [11] Ian, Graham. "The HTML Sourcebook : A complete guide to HTML 3.0", 2nd Edition, John Wiley & Sons Inc., USA 1996
- [12] Gupta, Ajay, Emilia Stoica, Ehab Al-Shaer. and Kurt Maly. Overstreet, Micheal, "Interactive Distance Learning", IEEE Internet Computing, January-February 1997
- [13] Hahn, Harley and Rick Stout. "The Internet Complete Reference", Osborne McGraw-Hill, Berkeley, CA 1997
- [14] Harrod and Elltiote. "Brewing Java: A Tutorial", URL <http://sunsite.unc.edu/javafaq/javatutorial.html>, 1995-1997

- [15] Herwijnen, Eric. "Practical SGML", 2nd edition, Kluwer Academic Publishers, Norwell, MA 1994
- [16] Joshua, Marketos. "The Java Developer's Toolkit, Techniques and Technologies for Web Programmers", New York, NY 1997
- [17] Koch, George and Kevin Loney. "Oracle : The Complete Reference", Osborne McGraw Hill, Berkeley, CA 1995
- [18] Korth, Henry and Albert Silberschatz. "Database System Concepts", 2nd edition, McGraw Hill Inc., NY 1994
- [19] Lynch, Daniel and Marshall Rose. "Internet System Handbook", Addison-Wesley Publishing Company, UK 1993
- [20] Nofziger, David, Arthur Hornsby and Dana Hoag. "Pesticide Economic and Environmental Tradeoffs: Developers Perspective", First International Conference on MODSS for Agriculture and Environment, Honolulu, HI, July 1995
- [21] "Programmers Guide to Oracle PRO*C Precompiler", Release 2.0, Oracle Corporation, Redwood City, CA 1996
- [22] Ritchey, Tim. "Java!", New Riders Publishing , Indianapolis, IN 1995.
- [23] Rosen, Kenneth, Richard Rosinski, James Farber and Douglas Host. "Unix System V Release 4: An Introduction", 2nd edition, Osborne McGraw Hill, Berkeley, CA 1996
- [24] "Selected Tax Policy Implications of Global Electronic Commerce", <ftp://ftp.fedworld.gov/pub/tel/internet.txt>
- [25] Smythe, Colin. "Internetworking: Designing the Right Architectures", Addison-Wesley Publishing Company, UK 1995
- [26] Swank, Mark and Drew Kittel. "World Wide Web database Developer's Guide", Sams.net Publishing, Indianapolis, IN 1996
- [27] Tanenbaum, Andrew. "Distributed Operating Systems", Prentice Hall Inc., Englewood Cliffs, NJ 1995

Appendix I

CGI Program (query.cgi)

```
#!/bin/sh
#-----
# query.cgi:
# Written by: Aniruddha Gujrathi
# Written on: 11 Sep, 1997
#-----
#
# Setting up environment
#
PROG_PATH=/var/http/demo/cgi-bin/PEET
PROC_PATH=/usr/DISK3/PROC/PEET
GIF_PATH=/var/http/demo/public/PEET
TMP_PATH=/usr/DISK2/tmpdir
PID=$$
export PROG_PATH
export PROC_PATH
export GIF_PATH
export TMP_PATH
export PID
#
# Setting up Oracle Environment
#
ORACLE_HOME=/usr/DISK2/oracle/product/732
ORACLE_SID=sid1
ORACLE_BASE=/usr/DISK2/oracle
ORACLE_PATH=/usr/DISK2/oracle/product/732/bin
LD_LIBRARY_PATH=/usr/DISK2/oracle/product/732/lib
export ORACLE_HOME
export ORACLE_SID
export ORACLE_BASE
export ORACLE_PATH
export LD_LIBRARY_PATH
/bin/rm $GIF_PATH/report.gif.*
/bin/rm $GIF_PATH/rank.gif.*
echo 'Content-type: text/html'
echo
#
# Fetching information from FORM
#
$PROG_PATH/aniquery
echo '</CENTER>'
echo '</BODY>'
echo '</HTML>'
#
#-----Delete the temporary files
#/bin/rm $TMP_PATH/*$PID*
#
```

Appendix II

PRO*C Program (ppi.pc)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/*-- Define constants for VARCHAR lengths. --*/
#define UNAME_LEN 20
#define PWD_LEN 40
#define ARRAY_LENGTH 10
#define MAX_REC 100
#define GRAPH_HEIGHT 15
#define GRAPH_WIDTH 36
/*-- Declare variables. No declare section is needed if MODE=ORACLE. --*/
char *username = "PEETMGR";
char *password = "PEETMGR";

EXEC SQL INCLUDE sqlca.h;

/* Declare error handling function. */
void sql_error();
void blank2null();

EXEC SQL BEGIN DECLARE SECTION;
short var_CountyIndex;
VARCHAR sqlstmt[1000];
EXEC SQL END DECLARE SECTION;

VARCHAR var_CountyName[12], var_SoilName[47], var_CountyIndexc[25];
char var_SoilIndexc[25];
int var_SoilIndex;

short treatmnt_EfficacyIndex[100], treatmnt_Index[100], var_StartDay, herb_TreatmentIndex[122];
char herb_TradeName[122][47], treatmnt_ApplType[100][5];
float herb_ApplAmt[122], cost_PurchasePrice[24], unit_Conversion[15];
char herb_ApplUnits[122][8], herb_TradeNameAll[122][57], cost_Tradename[24][47];
char cost_PurchaseUnit[24][8], unit_UnitType[15][2], unit_OldUnits[15][8], unit_StdUnits[15][8];

char weed_WeedName[50][31], weed_WeedCode[50][11];
float weed_CompetitiveIndex[50], weed_LowDensity[50], weed_HighDensity[50], weed_Density[50];
char weed_density[50][2], MoistureLevel[2], WeedSize[4], eff_WeedCode[6552][11];
short eff_EfficacyIndex[6552], weed, p, pot, gwTreatmnt;
float eff_Efficacy[6552], I,A,Y1, Y2, Y3, V, gw_HoHo20[1000];
char var_IrrigationType, RankingScheme[2], var_weed[10], var_density[5], PotList_ApplType[150][5];
short TillageCode, CountyIndex, gw_TreatmentIndex[1000], gw_Rank[1000];
float D, Y_Loss_Rel, NY1_Loss, Y1_Loss_Abs, NY2_Loss, Y2_Loss_Abs, NY3_Loss, Y3_Loss_Abs;
short PotList_TreatmentIndex[150], PotList_Rank[150], PotList_EfficacyIndex[150], PotList_Exist[150];
float PotList_HoHo[150];
float PotList_LossLow[150], PotList_LossTypical[150], PotList_LossHigh[150], PotList_Position[150];
```

```

int paramNo=0;

char *substr(char *str,int x,int y) {
    int i,j;
    char v_str[25];

    for (i=x-1,j=0; (i<y-1) && (*(str+i)!=0x00); i++,j++)
        v_str[j] = *(str+i);
    v_str[j] = 0x00;
    return(v_str);
}

void main(int argc, char *argv[]) {
    int i,j,k,m,n,p,q,r,t,ty,w,x,y,z, flag, num, herb, cost, both;
    int unit, efficacy, treatmnt;
    char tmpAppl[5], APPL[5];
    FILE *weedfile,*rptfile, *rankfile, *costfile,*javaFile;
    short tmpIndex, tmpRank, tmpEff;
    float tmpHoHo, tmp_HoHo, tmp_Loss, tmp;
    float Loss_low, Loss_typical, Loss_high, loss_max, AppCost, ScoutCost, Price, E;
    char CostUnit[8], var_cost[80], var_cost2[80], tmp_hoho[8];
    float Conv1, Conv2,prevHoHo;
    short prevRank, prevValue, tmpLeap;
    short MAX_Y, graph, Graph_X[36], Graph_Y[15], Graph_Rank[150];
    char herb_applamt[25], tillageBuffer[2], irrgBuffer[4], probBuffer[7];

    EXEC SQL WHENEVER SQLERROR DO sql_error("ORACLE error--\n");
    EXEC SQL CONNECT :username IDENTIFIED BY :password;

    strcpy(RankingScheme, argv[1]);
    AppCost = atoi(argv[2]);
    ScoutCost = atoi(argv[3]);
    Y1 = atof(argv[4]);
    Y2 = atof(argv[5]);
    Y3 = atof(argv[6]);
    V = atof(argv[7]);
    if ( (weedfile = fopen(argv[8],"rt")) == NULL ) {
        printf("error opening file. %s\n", argv[8]);
        exit(0);
    }
    if ( (rptfile = fopen(argv[9],"wt")) == NULL ) {
        printf("Unable to open output file.\n");
        exit(0);
    }
    if ( (rankfile = fopen(argv[10],"wt")) == NULL ) {
        printf("Unable to open output file.\n");
        exit(0);
    }
    javaFile=fopen(argv[16],"w");
    if(!javaFile) {
        printf("Unable to open java output file\n");
        exit(0);
    }
    strcpy((char *)var_CountyName.arr, argv[11]);
    var_CountyName.len = strlen( (char *)var_CountyName.arr );

```

```

strcpy((char *)var_SoilName.arr, argv[12]);
var_SoilName.len = strlen( (char *)var_SoilName.arr);
for (i=0; i<strlen((char*)var_SoilName.arr); i++) {
    if (var_SoilName.arr[i] == '_')
        var_SoilName.arr[i] = ' ';
}
var_IrrigationType = argv[13][0];
if ( (costfile = fopen(argv[14], "rt")) == NULL ) {
    printf("Error opening cost file.\n");
    exit(0);
}
strcpy(probBuffer, argv[15]);
TillageCode = 2;
var_StartDay = 135;

for (i=0; i<100; i++)
    treatmnt_Index[i] = 0;
for (i=0; i<122; i++)
    herb_AplAmt[i] = 0;
for (i=0; i<6552; i++)
    eff_Efficacy[i] = -1;
for (i=0; i<24; i++)
    cost_PurchasePrice[i] = 0;
for (i=0; i<15; i++)
    unit_Conversion[i] = 0;
for (i=0; i<50; i++)
    weed_HighDensity[i] = 0;

strcpy(MoistureLevel, "N");
strcpy(WeedSize, "<2");
for (i=0; i<6552; i++)
    eff_EfficacyIndex[i] = 0;
TillageCode = 2;
for (i=0; i<1000; i++)
    gw_HoHo20[i] = -1;

/*Call stored function to get CountyIndex */
EXEC SQL EXECUTE
    BEGIN
        :var_CountyIndex := GETCOUNTYINDEX( :var_CountyName );
    END;
END-EXEC;

sprintf((char*)var_CountyIndexc.arr, "%d", var_CountyIndex);
var_CountyIndexc.len = strlen((char*)var_CountyIndexc.arr);
EXEC SQL
SELECT SoilIndex INTO :var_SoilIndexc
    FROM peetmgr.SOILNAME
WHERE CountyIndex = :var_CountyIndexc
    AND SoilName = :var_SoilName;

var_SoilIndex = atoi(var_SoilIndexc);
EXEC SQL
SELECT ApplType, TreatmentIndex, EfficacyIndex
    INTO :treatmnt_ApplType, :treatmnt_Index, :treatmnt_EfficacyIndex

```

```

FROM peetmgr.TREATMNT
WHERE ApplType != 'POT' OR (ApplType = 'POT' AND ApplDayBegin = 135);

for (treatmnt=0; treatmnt_Index[treatmnt] != 0; treatmnt++) {
    treatmnt_ApplType[treatmnt][4] = 0x00; /* null out at end of string */
    blank2null(treatmnt_ApplType[treatmnt]);
}
EXEC SQL
SELECT *
    INTO :herb_TreatmentIndex, :herb_TradeName, :herb_ApplAmt, :herb_ApplUnits
    FROM peetmgr.HERB;

for (herb=0; herb_ApplAmt[herb]!=0; herb++) {
    herb_TradeName[herb][46] = 0x00; /* null out at end of string */
    blank2null(herb_TradeName[herb]);
    herb_ApplUnits[herb][7] = 0x00;
    blank2null(herb_ApplUnits[herb]);
    sprintf(herb_applamt, "%f", herb_ApplAmt[herb]);
    sprintf(herb_TradeNameAll[herb], "%s %s %s", herb_TradeName[herb], substr(herb_applamt, 1,
        strlen(herb_applamt)-3), herb_ApplUnits[herb]);
}
fgets(var_cost,80,costfile);
cost = 0;
while ( !feof(costfile) ) {
    i = strlen(var_cost);
    var_cost[i-1] = 0x00;
    strcpy(cost_TradeName[cost],strtok(var_cost,"t"));
    strcpy(var_cost2,strtok("\0","t"));
    cost_PurchasePrice[cost] = atof(var_cost2);
    strcpy(cost_PurchaseUnit[cost],strtok("\0","t"));
    cost++;
    fgets(var_cost,80,costfile);
}
fclose(costfile);
EXEC SQL
SELECT *
    INTO :unit_UnitType, :unit_OldUnits, :unit_StdUnits, :unit_Conversion
    FROM peetmgr.UNIT;

for (unit=0; unit_Conversion[unit] != 0; unit++) {
    unit_UnitType[unit][1] = 0x00;
    blank2null(unit_UnitType[unit]);
    unit_OldUnits[unit][7] = 0x00; /* null out at end of string */
    blank2null(unit_OldUnits[unit]);
    unit_StdUnits[unit][7] = 0x00;
    blank2null(unit_StdUnits[unit]);
}
EXEC SQL
SELECT *
    INTO :weed_WeedName, :weed_WeedCode, :weed_CompetitiveIndex, :weed_LowDensity,
        :weed_HighDensity
    FROM peetmgr.WEED;

for (weed=0; weed_HighDensity[weed]!=0; weed++) {
    weed_WeedCode[weed][10] = 0x00;

```

```

    blank2null(weed_WeedCode[weed]);
    weed_WeedName[weed][30] = 0x00;
    blank2null(weed_WeedName[weed]);
    strcpy(weed_density[weed],"n");
}
EXEC SQL
SELECT Value
  INTO :A
  FROM peetmgr.PARAM
 WHERE Name = 'Maxyieldloss';

EXEC SQL
SELECT Value
  INTO :I
  FROM peetmgr.PARAM
 WHERE Name = 'Lossperci';

EXEC SQL
SELECT EfficacyIndex, WeedCode, Efficacy
  INTO :eff_EfficacyIndex, :eff_WeedCode, :eff_Efficacy
  FROM peetmgr.EFFICACY
 WHERE MoistureLevel = 'N' AND WeedSize = '<2';

for (efficacy=0; eff_Efficacy[efficacy]!=-1; efficacy++) {
    eff_WeedCode[efficacy][10] = 0x00;
    blank2null(eff_WeedCode[efficacy]);
}
strcpy((char*)sqlstmt.arr,"SELECT TreatmentIndex, Rank, ");
strcat((char*)sqlstmt.arr,probBuffer);
strcat((char*)sqlstmt.arr," FROM peetmgr.GWHAZARD ");
strcat((char*)sqlstmt.arr," WHERE IrrigationType = ");
sprintf(irrgBuffer,"%c",var_IrrigationType);
strcat((char*)sqlstmt.arr,irrgBuffer);
sprintf(tillageBuffer,"%d",TillageCode);
strcat((char*)sqlstmt.arr," AND TillageCode = ");
strcat((char*)sqlstmt.arr,tillageBuffer);
strcat((char*)sqlstmt.arr," AND CountyIndex = ");
strcat((char*)sqlstmt.arr,(char*)var_CountyIndexc.arr);
strcat((char*)sqlstmt.arr," AND SoilIndex = ");
strcat((char*)sqlstmt.arr,var_SoilIndexc);
sqlstmt.len=strlen((char*)sqlstmt.arr);
EXEC SQL WHENEVER SQLERROR DO sql_error("Unable to prepare cursor\n");
EXEC SQL PREPARE S FROM :sqlstmt;
EXEC SQL WHENEVER SQLERROR DO sql_error("Unable to declare cursor\n");
EXEC SQL DECLARE C CURSOR FOR S;
EXEC SQL WHENEVER SQLERROR DO sql_error("Unable to open cursor\n");
EXEC SQL OPEN C ;
EXEC SQL WHENEVER SQLERROR DO sql_error("Unable to fetch cursor\n");
EXEC SQL FETCH C INTO :gw_TreatmentIndex, :gw_Rank, :gw_HoHo20;
for (gwTreatmnt=0; gw_HoHo20[gwTreatmnt] != -1; gwTreatmnt++);
EXEC SQL COMMIT WORK RELEASE;

while ( fscanf(weedfile,"%s\t%s",var_weed,var_density) != EOF )
    for (i=0; i<weed; i++)
        if ( strcmp(weed_WeedCode[i],var_weed) == 0 )

```

```

        strcpy(weed_density[i],var_density);
fclose(weedfile);
fprintf(rptfile, "\n===== \n");
fprintf(rptfile, "    Herbicides    | Potential Loss | RANK | Potential\n");
fprintf(rptfile, "                | due to weeds  | | Groundwater Hazard\n");
fprintf(rptfile, "                |            | | BETTER    WORSE\n");
fprintf(rptfile, "                | Yield lb / acre | |1%% 10%% 100%% 1000%%\n");
fprintf(rptfile, "                | %4.0f %4.0f %4.0f | \n", Y1, Y2, Y3 );
fprintf(rptfile, "\n===== \n");
fprintf(rptfile, "                | ( $ / acre ) | \n");

D = 0;
for ( p = 0; p < weed; p++ ) {
    if ( strcmp(weed_density[p], "n") == 0 ) {
        weed_Density[p] = 0;
        continue;
    }
    if ( strcmp(weed_density[p], "l") == 0 )
        weed_Density[p] = weed_LowDensity[p];
    if ( strcmp(weed_density[p], "h") == 0 )
        weed_Density[p] = weed_HighDensity[p];
    D = D + weed_CompetitiveIndex[p] * weed_Density[p];
}
Y_Loss_Rel = 0.01 * A * I * D / (A + I * D );
NY1_Loss = Y_Loss_Rel * Y1 * V / 2000;
NY2_Loss = Y_Loss_Rel * Y2 * V / 2000;
NY3_Loss = Y_Loss_Rel * Y3 * V / 2000;
fprintf(rptfile, "NO TREATMENT VALUES    | %4.0f %4.0f %4.0f | NA \n", NY1_Loss,
        NY2_Loss, NY3_Loss);
fprintf(javaFile, "<PARAM NAME='SOIL%d' VALUE='NO TREATMENT VALUES'>\n",
        paramNo);
fprintf(javaFile, "<PARAM NAME='LOW%d' VALUE='%0f'>\n", paramNo, NY1_Loss);
fprintf(javaFile, "<PARAM NAME='TYPICAL%d' VALUE='%0f'>\n", paramNo, NY2_Loss);
fprintf(javaFile, "<PARAM NAME='HIGH%d' VALUE='%0f'>\n", paramNo, NY3_Loss);
fprintf(javaFile, "<PARAM NAME='LOSS%d' VALUE='-1'>\n", paramNo);
paramNo++;
pot = 0;
for ( i=0; i < gwTreatmnt; i++ ) {
    for ( j=0; j < treatmnt; j++ ) {
        if ( gw_TreatmentIndex[i] == treatmnt_Index[j] ) {
            strcpy(PotList_ApplType[pot], treatmnt_ApplType[j]);
            PotList_TreatmentIndex[pot] = gw_TreatmentIndex[i];
            PotList_HoHo[pot] = gw_HoHo20[i];
            PotList_Rank[pot] = gw_Rank[i];
            PotList_EfficacyIndex[pot] = treatmnt_EfficacyIndex[j];
            PotList_Exist[pot] = 1;
            pot++;
            break;
        }
    }
}
for ( i=0; i < pot; i++ ) {
    m = i;
    for ( j=i+1; j < pot; j++ )
        if ( strcmp(RankingScheme, "S") == 0 )

```

```

    if ( PotList_Rank[m] > PotList_Rank[j] )
        m = j;
else
    if ( PotList_HoHo[m] > PotList_HoHo[j] )
        m = j;

if ( m != i ) {
    strcpy(tmpAppl,PotList_ApplType[i]);
    tmpIndex = PotList_TreatmentIndex[i];
    tmpRank = PotList_Rank[i];
    tmpHoHo = PotList_HoHo[i];
    tmpEff = PotList_EfficacyIndex[i];
    strcpy(PotList_ApplType[i],PotList_ApplType[m]);
    PotList_TreatmentIndex[i] = PotList_TreatmentIndex[m];
    PotList_Rank[i] = PotList_Rank[m];
    PotList_HoHo[i] = PotList_HoHo[m];
    PotList_EfficacyIndex[i] = PotList_EfficacyIndex[m];
    strcpy(PotList_ApplType[m],tmpAppl);
    PotList_TreatmentIndex[m] = tmpIndex;
    PotList_Rank[m] = tmpRank;
    PotList_HoHo[m] = tmpHoHo;
    PotList_EfficacyIndex[m] = tmpEff;
}
}
loss_max = 0;
for( i = 0; i < pot ; i++ ) {
    Loss_low = 0;
    Loss_typical = 0;
    Loss_high = 0;
    flag = 0;
    num = 0;
    for ( j = 0; j < herb ; j++ ) {
        if ( herb_TreatmentIndex[j] == PotList_TreatmentIndex[i] ) {
            flag = 1; /* # Change flag */
            for ( z = 0; z < cost; z++ )
                if ( strcmp(cost_Tradename[z],herb_TradeName[j]) == 0 ) {
                    Price = cost_PurchasePrice[z];
                    strcpy(CostUnit,cost_PurchaseUnit[z]);
                    break;
                }
            both = 0;
            for ( z = 0; z < unit; z++ ) {
                if ( strcmp(unit_OldUnits[z],herb_ApplUnits[j]) == 0 ) {
                    Conv1 = unit_Conversion[z];
                    both ++;
                }
                if ( strcmp(unit_OldUnits[z],CostUnit) == 0 ) {
                    Conv2 = unit_Conversion[z];
                    both ++;
                }
            }
            Price = (herb_ApplAmt[j] * Conv1) * ( Price / Conv2 );
            Loss_low = Loss_low + Price;
            Loss_typical = Loss_typical + Price;
            Loss_high = Loss_high + Price;

```



```

    }
    else {
        if ( flag == 1 ) {
            flag = 0;
            break;
        }
    }
}
Loss_low = Loss_low + AppCost + ScoutCost;
Loss_typical = Loss_typical + AppCost + ScoutCost;
Loss_high = Loss_high + AppCost + ScoutCost;
D = 0;
for ( p = 0; p < weed; p++ ) {
    if ( weed_Density[p] == 0 )
        continue;
    E = -1;
    for ( t=0; t<efficacy; t++ ) {
        if ( (eff_EfficacyIndex[t] == PotList_EfficacyIndex[i] &&
            (strcmp(eff_WeedCode[t],weed_WeedCode[p]) == 0) ) ) {
            E = eff_Efficacy[t];
            break;
        }
    }
    if ( E == -1 ) {
        PotList_Exist[i] = 0;
        D = 0;
    }
    else
        D = D + weed_CompetitiveIndex[p] * weed_Density[p] * ( 1 - E );
}
Y_Loss_Rel = 0.01 * A * I * D / ( A + I * D );
Y1_Loss_Abs = Y_Loss_Rel * Y1 / 2000;
Y2_Loss_Abs = Y_Loss_Rel * Y2 / 2000;
Y3_Loss_Abs = Y_Loss_Rel * Y3 / 2000;
Loss_low = Loss_low + Y1_Loss_Abs * V;
Loss_typical = Loss_typical + Y2_Loss_Abs * V;
Loss_high = Loss_high + Y3_Loss_Abs * V;
PotList_LossLow[i] = Loss_low;
PotList_LossTypical[i] = Loss_typical;
PotList_LossHigh[i] = Loss_high;
if ( Loss_typical > loss_max )
    loss_max = Loss_typical;
}
prevRank = 0;
tmpLeap = 0;
prevHoHo = -1;
prevValue = 0;
for ( q=0; q<pot; q++ ) {
    if ( PotList_Exist[q] == 0 )
        continue;
    if ( strcmp(RankingScheme,"S") == 0 ) {
        if ( PotList_Rank[q] != prevValue ) {
            prevValue = PotList_Rank[q];
            tmpLeap ++;
            prevRank += tmpLeap;
        }
    }
}

```

```

        PotList_Rank[q] = prevRank;
        tmpLeap = 0;
    }
    else {
        tmpLeap ++;
        PotList_Rank[q] = prevRank;
    }
}
else {
    if ( PotList_HoHo[q] != prevHoHo ) {
        prevHoHo = PotList_HoHo[q];
        tmpLeap ++;
        prevRank += tmpLeap;
        PotList_Rank[q] = prevRank;
        tmpLeap = 0;
    }
    else {
        tmpLeap ++;
        PotList_Rank[q] = prevRank;
    }
}
}
if ( NY2_Loss > loss_max )
    loss_max = NY2_Loss;
if ( loss_max <= 150 )
    MAX_Y = 150;
if ( loss_max > 150 && loss_max <= 300 )
    MAX_Y = 300;
if ( loss_max > 300 && loss_max <= 450 )
    MAX_Y = 450;
if ( loss_max > 450 && loss_max <= 600 )
    MAX_Y = 600;
if ( loss_max > 600 && loss_max <= 900 )
    MAX_Y = 900;
if ( loss_max > 900 )
    MAX_Y = 1200;
graph=0;
for(q=0; q<pot; q++) {
    if ( PotList_Exist[q] == 0 )
        continue;
    tmp_HoHo = PotList_HoHo[q] * 100;
    tmp_Loss = PotList_LossTypical[q];
    if ( tmp_Loss > MAX_Y )
        y = GRAPH_HEIGHT;
    else {
        tmp = tmp_Loss * GRAPH_HEIGHT / MAX_Y;
        if ( tmp != (int)( tmp ) )
            y = (int)(tmp) + 1;
        else
            y = (int)(tmp);
    }
}
if ( tmp_HoHo > 1000 )
    x = GRAPH_WIDTH;
else
    if ( tmp_HoHo < 1 )

```

```

    x = 1;
else {
    if ( tmp_HoHo < 10 ) {
        tmp = (tmp_HoHo / 10) * ( GRAPH_WIDTH / 3 );
        if ( tmp != (int)(tmp) )
            x = (int)(tmp) + 1;
        else
            x = (int)(tmp);
    }
    else
        if ( tmp_HoHo < 100 ) {
            tmp = ((tmp_HoHo - 10) / 90) * (GRAPH_WIDTH / 3);
            if ( tmp != (int)(tmp) )
                x = (int)(tmp) + 1 + (int)(GRAPH_WIDTH/3);
            else
                x = (int)(tmp) + (int)(GRAPH_WIDTH/3);
        }
        else {
            tmp = ((tmp_HoHo - 100) / 900) * ( GRAPH_WIDTH / 3);
            if ( tmp != (int)(tmp) )
                x = (int)(tmp) + 1 + (int)(GRAPH_WIDTH * 2 / 3);
            else
                x = (int)(tmp) + (int)(GRAPH_WIDTH * 2 / 3);
        }
    }
    if ( graph == 0 ) {
        graph ++;
        Graph_X[graph] = x;
        Graph_Y[graph] = y;
        PotList_Position[q] = graph;
        continue;
    }
    for( j=0; j<=graph; j++ )
        if ( (Graph_X[j] == x) && (Graph_Y[j] == y) )
            break;
    if ( j <= graph )
        PotList_Position[q] = j;
    else {
        graph++;
        Graph_X[graph] = x;
        Graph_Y[graph] = y;
        PotList_Position[q] = graph;
    }
}
for ( m=0; m<graph; m++ )
    Graph_Rank[m] = m;
for ( m=0; m<graph-1; m++ ) {
    r = m;
    for(n=m+1; n<graph; n++ )
        if ( Graph_X[Graph_Rank[n]] < Graph_X[Graph_Rank[r]] )
            r = n;
    else
        if ( (Graph_X[Graph_Rank[n]] == Graph_X[Graph_Rank[r]]) &&
            (Graph_Y[Graph_Rank[n]] < Graph_Y[Graph_Rank[r]]) )
            r = n;
}

```

```

if ( r != m ) {
    tmp = Graph_Rank[m];
    Graph_Rank[m] = Graph_Rank[r];
    Graph_Rank[r] = tmp;
}
}
fprintf(rankfile,"%4d %f\n", MAX_Y, NY2_Loss);
for( m=0; m<graph; m++ )
    fprintf(rankfile,"%4d %4d %4d \n", m, Graph_X[Graph_Rank[m]], Graph_Y[Graph_Rank[m]]);
fprintf(rankfile,"#\n");
for ( ty=0; ty<4; ty++ ) {
    if ( ty == 0 ) {
        strcpy(APPL,"PPI");
        fprintf(rankfile,"Pre-Plant Incorporated =====\n");
    }
    if ( ty == 1 ) {
        strcpy(APPL,"PRE");
        fprintf(rankfile,"$Pre-Emergence =====\n");
    }
    if ( ty == 2 ) {
        strcpy(APPL,"SPOT");
        fprintf(rankfile,"$Spot =====\n");
    }
    if ( ty == 3 ) {
        strcpy(APPL,"POT");
        fprintf(rankfile,"$Post-Emergence =====\n");
    }
}
for( n=0; n<graph; n++ ) {
    for( m=0; m<pot; m++ ) {
        if ( PotList_Exist[m] == 0 )
            continue;
        if ( strcmp(PotList_ApplType[m],APPL) != 0 )
            continue;
        if ( PotList_Position[m] != Graph_Rank[n] )
            continue;
        flag = 0;
        for ( k=0; k<herb; k++ ) {
            if ( PotList_TreatmentIndex[m] == herb_TreatmentIndex[k] ) {
                if ( flag == 0 ) {
                    fprintf(rankfile,"%c -- %-4s -- %-24s\n", n+65-1, APPL, herb_TradeNameAll[k]);
                    flag = 1;
                }
                else
                    fprintf(rankfile,"----- + %-24s\n", herb_TradeNameAll[k]);
            }
            else {
                if ( flag == 1 ) {
                    flag = 0;
                    break;
                }
            }
        }
    }
}
}
}
}

```

```

for ( ty = 0; ty < 4; ty++ ) {
  if ( ty == 0 ) {
    strcpy(APPL,"PPI");
    fprintf(rptfile,"Pre-Plant Incorporated |-----|-----\n");
    fprintf(javaFile,"<PARAM NAME=\\"SOIL%d\\" VALUE=\\"Pre-Plant Incorporated\\">\n",
      paramNo++);
  }
  if ( ty == 1 ) {
    strcpy(APPL,"PRE");
    fprintf(rptfile,"Pre-Emergence |-----|-----\n");
    fprintf(javaFile,"<PARAM NAME=\\"SOIL%d\\" VALUE=\\"Pre-Emergence\\">\n",paramNo++);
  }
  if ( ty == 2 ) {
    strcpy(APPL,"SPOT");
    fprintf(rptfile,"Spot |-----|-----\n");
    fprintf(javaFile,"<PARAM NAME=\\"SOIL%d\\" VALUE=\\"Spot\\">\n",paramNo++);
  }
  if ( ty == 3 ) {
    strcpy(APPL,"POT");
    fprintf(rptfile,"Post-Emergence |-----|-----\n");
    fprintf(javaFile,"<PARAM NAME=\\"SOIL%d\\" VALUE=\\"Post-Emergence\\">\n",paramNo++);
  }
  for ( m=0; m<pot; m++ ) {
    if ( PotList_Exist[m] == 0 )
      continue;
    if ( strcmp(PotList_ApplType[m],APPL) != 0 )
      continue;
    flag = 0;
    for ( k=0; k<herb; k++ ) {
      if ( PotList_TreatmentIndex[m] == herb_TreatmentIndex[k] ) {
        if ( flag == 0 ) {
          if ( PotList_HoHo[m]*100 < 1 ) {
            strcpy(tmp_hoho,"<1");
            fprintf(javaFile,"<PARAM NAME=\\"LOSS%d\\" VALUE=\\"<1\\">\n",paramNo);
          }
          else {
            sprintf(tmp_hoho,"%3.0f", PotList_HoHo[m]*100);
            fprintf(javaFile,"<PARAM NAME=\\"LOSS%d\\" VALUE=\\"%.0f\\">\n",paramNo,
              PotList_HoHo[m]*100);
          }
          fprintf(rptfile," %-22.22s |",herb_TradeNameAll[k]);
          fprintf(javaFile,"<PARAM NAME=\\"SOIL%d\\" VALUE=\\"%.22s\\">\n",
            paramNo,herb_TradeNameAll[k]);
          fprintf(rptfile," %4.0f",PotList_LossLow[m]);
          fprintf(javaFile,"<PARAM NAME=\\"LOW%d\\" VALUE=\\"%.0f\\">\n",
            paramNo,PotList_LossLow[m]);
          fprintf(rptfile," %4.0f",PotList_LossTypical[m]);
          fprintf(javaFile,"<PARAM NAME=\\"TYPICAL%d\\" VALUE=\\"%.0f\\">\n",
            paramNo,PotList_LossTypical[m]);
          fprintf(rptfile," %4.0f |",PotList_LossHigh[m]);
          fprintf(javaFile,"<PARAM NAME=\\"HIGH%d\\" VALUE=\\"%.0f\\">\n",
            paramNo,PotList_LossHigh[m]);
          fprintf(rptfile," %3d |",PotList_Rank[m]);
          fprintf(rptfile," %s\n",tmp_hoho );
          flag = 1;
        }
      }
    }
  }
}

```

```

        paramNo++;
    }
    else {
        fprintf(rptfile, "+%-22s |          | \n", herb_TradeNameAll[k]);
        fprintf(javaFile, "<PARAM NAME=\\"SOIL%d\\" VALUE=\\"+%.22s\\"">\n",
                paramNo, herb_TradeNameAll[k]);
        paramNo++;
    }
}
else {
    if ( flag == 1 ) {
        flag = 0;
        break;
    }
}
}
}
}
fprintf(javaFile, "<PARAM NAME=\\"TOTALSOILS\\" VALUE=\\"%d\\"">\n", paramNo);
fprintf(rptfile, "===== \n");
fprintf(rptfile, "#\n");
for( w = 0; w < weed; w++ ) {
    if ( weed_Density[w] == 0 )
        continue;
    fprintf(rptfile, "\n\n");
    fprintf(rptfile, "<TABLE BORDER=3>\n");
    fprintf(rptfile, "<TR ALIGN=CENTER VALIGN=CENTER>\n");
    fprintf(rptfile, "<TD COLSPAN=3><B>\n");
    fprintf(rptfile, "Cost and Potential Groundwater Hazard by Herbicide\n");
    fprintf(rptfile, "</B></TD>\n</TR>\n");
    fprintf(rptfile, "<TR ALIGN=CENTER VALIGN=CENTER>\n");
    fprintf(rptfile, "<TD>Herbicides</TD>\n");
    fprintf(rptfile, "<TD COLSPAN=2>Weed Densities\n");
    fprintf(rptfile, "<BR><BR>Competative Load</TD>\n</TR>\n");
    fprintf(rptfile, "<TR>\n<TD>\n");
    fprintf(rptfile, "<TD ALIGN=CENTER VALIGN=CENTER COLSPAN=2>\n");
    fprintf(rptfile, "<B><I>%s</B></I></TD>\n</TR>\n", weed_WeedName[w]);
    fprintf(rptfile, "<TR>\n<TD>\n");
    fprintf(rptfile, "<TD ALIGN=CENTER VALIGN=CENTER>Density</TD>\n");
    fprintf(rptfile, "<TD ALIGN=CENTER VALIGN=CENTER>Weed Load</TD>\n");
    fprintf(rptfile, "</TR>\n");
    fprintf(rptfile, "<TR>\n");
    fprintf(rptfile, "<TD ALIGN=LEFT>No Treatment Values</TD>\n");
    fprintf(rptfile, "<TD ALIGN=CENTER>%5.0f</TD>\n", weed_Density[w]);
    fprintf(rptfile, "<TD ALIGN=CENTER>%5.0f</TD>\n",
            weed_Density[w]*weed_CompetitiveIndex[w]);
    fprintf(rptfile, "</TR>\n");
    for ( ty = 0; ty < 4; ty++ ) {
        if ( ty == 0 ) {
            strcpy(APPL, "PPI");
            fprintf(rptfile, "<TR>\n");
            fprintf(rptfile, "<TD ALIGN=CENTER><B>Pre-Plant Incorporated</B></TD>\n");
            fprintf(rptfile, "<TD></TD>\n<TD></TD>\n");
            fprintf(rptfile, "</TR>\n");
        }
    }
}

```

```

if ( ty == 1 ) {
    strcpy(APPL,"PRE");
    fprintf(rptfile,"<TR>\n");
    fprintf(rptfile,"<TD ALIGN=CENTER><B>Pre-Emergence</B></TD>\n");
    fprintf(rptfile,"<TD></TD>\n<TD></TD>\n");
    fprintf(rptfile,"</TR>\n");
}
if ( ty == 2 ) {
    strcpy(APPL,"SPOT");
    fprintf(rptfile,"<TR>\n");
    fprintf(rptfile,"<TD ALIGN=CENTER><B>Spot</B></TD>\n");
    fprintf(rptfile,"<TD></TD>\n<TD></TD>\n");
    fprintf(rptfile,"</TR>\n");
}
if ( ty == 3 ) {
    strcpy(APPL,"POT");
    fprintf(rptfile,"<TR>\n");
    fprintf(rptfile,"<TD ALIGN=CENTER><B>Post-Emergence</B></TD>\n");
    fprintf(rptfile,"<TD></TD>\n<TD></TD>\n");
    fprintf(rptfile,"</TR>\n");
}
for( m=0; m<pot; m++ ) {
    if ( PotList_Exist[m] == 0 )
        continue;
    if ( strcmp(PotList_ApplType[m],APPL) != 0 )
        continue;
    for ( t=0; t<efficacy; t++ ) {
        if ( (eff_EfficacyIndex[t] == PotList_EfficacyIndex[m]) &&
            (strcmp(weed_WeedCode[t],weed_WeedCode[w]) == 0) ) {
            E = eff_Efficacy[t];
            break;
        }
    }
    flag = 0;
    for ( k=0; k<herb; k++ ) {
        if ( PotList_TreatmentIndex[m] == herb_TreatmentIndex[k] ) {
            if ( flag == 0 ) {
                fprintf(rptfile,"<TR>\n");
                fprintf(rptfile,"<TD ALIGN=LEFT>");
                fprintf(rptfile,"%-24s</TD>",herb_TradeNameAll[k]);
                fprintf(rptfile,"<TD ALIGN=CENTER>\n");
                fprintf(rptfile,"%5.0f</TD>",weed_Density[w]*(1-E));
                fprintf(rptfile,"<TD ALIGN=CENTER>\n");
                fprintf(rptfile,"%5.0f",weed_CompetitiveIndex[w]*weed_Density[w]*(1-E));
                fprintf(rptfile,"</TD>\n</TR>\n");
                flag = 1;
            }
        }
        else {
            fprintf(rptfile,"<TR>\n");
            fprintf(rptfile,"<TD ALIGN=LEFT>");
            fprintf(rptfile,"%-24s",herb_TradeNameAll[k]);
            fprintf(rptfile,"</TD>\n<TD></TD>\n<TD></TD>\n");
            fprintf(rptfile,"</TR>\n");
        }
    }
}
}

```


Appendix III

Java Applet for Graph Generation

```
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.net.*;
import java.util.StringTokenizer;
import java.lang.Math;

class headCanvas extends Canvas {
    int maxValue, align;
    String titleLeft = "Potential Loss due to Weeds";
    String titleRight = "Potential GroundWater Hazard";

    headCanvas(int max, int alignment) {
        maxValue = max;
        align = alignment;
    }

    public void reset(int max, int alignment) {
        maxValue = max;
        align = alignment;
        repaint();
    }

    public Dimension minimumSize() {
        if(align == 1)
            return new Dimension(200,45);
        else
            return new Dimension(215,45);
    }

    public Dimension preferredSize() {
        if(align == 1)
            return new Dimension(200,45);
        else
            return new Dimension(215,45);
    }

    public void paint(Graphics g) {
        int cx;
        Font font = new Font("Arial",Font.BOLD,10);
        FontMetrics fm = getFontMetrics(font);
        cx=size().width;
        if(align == 1) {
            g.drawString(titleLeft,(cx-fm.stringWidth(titleLeft))/2,20);
            g.drawString(""+maxValue,0,40);
            g.drawString(""+maxValue/2,cx/2,40);
            g.drawString("0",cx-fm.stringWidth("0"),40);
        }
        else {
            cx = 200;
        }
    }
}
```

```

        g.drawString(titleRight,(cx-fm.stringWidth(titleRight))/2,20);
        g.drawString("1000%",cx-fm.stringWidth("1000%"),40);
        g.drawString("100%",2*cx/3-fm.stringWidth("100%")/2,40);
        g.drawString("10%",cx/3-fm.stringWidth("10%")/2,40);
        g.drawString("1%",0,40);
    }
}

class keyCanvas extends Canvas {
    String yieldLabel;

    keyCanvas(String yieldLabel) {
        this.yieldLabel = "Potential loss for " +yieldLabel;
    }

    public void reset(String newYield) {
        yieldLabel = "Potential loss for " +newYield;
        repaint();
    }

    public void paint(Graphics g) {
        Font font = new Font("Arial", Font.PLAIN, 10);
        FontMetrics fm = getFontMetrics(font);
        g.setColor(Color.blue);
        g.fillRect(0,5,10,10);
        g.setColor(Color.black);
        g.drawString(yieldLabel, 12, 15);
        g.setColor(Color.red);
        g.fillRect(0,17,10,10);
        g.setColor(Color.black);
        g.drawString("GroundWater Hazard", 12, 25);
    }

    public Dimension preferredSize() {
        return new Dimension(180,30);
    }

    public Dimension minimumSize() {
        return new Dimension(180,30);
    }
}

class graphCanvas extends Canvas {
    int[] array;
    int count;
    int maxValue=0;
    Font textFont;
    FontMetrics fm;
    int cx, cy;
    int maxWidth;
    int yPos;
    int height;
    Dimension mySize;
    Color graphColor;
}

```

```

int alignment;

graphCanvas(int[] iArray, int top, int maxWidth, int maxValue, int alignment, Color color) {
    count = top;
    array = new int[top];
    for(int i=0;i<count;i++)
        array[i] = iArray[i];
    this.maxWidth = maxWidth-1;
    yPos = 0;
    this.alignment = alignment;
    this.maxValue = maxValue;
    graphColor = color;
}

public void init() {
    textFont = new Font("Arial",Font.BOLD,12);
    fm = getFontMetrics(textFont);
    height = fm.getHeight();
    mySize = new Dimension(this.maxWidth, size().height);
}

public void addNotify() {
    super.addNotify();
    init();
}

public Dimension minimumSize() {
    return new Dimension(200,size().height);
}

public Dimension preferredSize() {
    return new Dimension(200,size().height);
}

void redraw(int yPos) {
    this.yPos = (fm.getAscent() + fm.getDescent() + fm.getLeading())*yPos;
    if(this.yPos < 0 )
        return;
    repaint();
}

public void reInit(int[] iArray, int top, int maxWidth, int maxValue, int alignment, Color color) {
    this.maxWidth = maxWidth;
    this.maxValue = maxValue;
    this.alignment = alignment;
    graphColor = color;
    count = top;
    array = new int[top];
    for(int i=0;i<count;i++)
        array[i] = iArray[i];
    yPos = 0;
    repaint();
}

void reset() {

```

```

        yPos = 0;
        repaint();
    }

    public void paint(Graphics g) {
        int midValue;
        int myWidth;
        float xx;

        g.setColor(graphColor);
        for(int i=0;i<count;i++) {
            if(array[i] < 0)
                continue;
            cy = (fm.getAscent() + fm.getDescent() + fm.getLeading())*i+3;
            if (alignment == 1) {
                cx = Math.max(array[i]*maxWidth / maxVale, 1);
                g.fillRect(maxWidth-cx,cy-yPos,maxWidth,10);
            }
            else {
                xx = Math.max((float)(Math.log(array[i])/Math.log(10)),(float)0.0);
                cx = Math.min(Math.max((int)(xx * (float)maxWidth / 3.0), 1),
                    maxWidth);
                g.fillRect(0,cy-yPos,cx,10);
            }
        }
        g.setColor(Color.black);
        g.drawLine(0,1,size().width-1,1);
        g.drawLine(0,size().height-2,size().width-1,size().height-2);
        if(alignment == 1)
            g.drawLine(0,0,0,size().height-1);
        else
            g.drawLine(size().width-1,0,size().width-1,size().height-1);
    }
}

class newCanvas extends Canvas {
    Font textFont;
    FontMetrics fm;
    String[] array = new String[100];
    int count;
    Dimension mySize;
    int maxLength;
    int height;
    int yPos;

    newCanvas(String[] sArray, int top) {
        for(int i=0;i<top;i++)
            array[i] = sArray[i];
        count = top;
        yPos = 0;
    }

    void redraw(int yPos) {
        this.yPos = fm.getHeight()*yPos;
        if(this.yPos < 0 )

```

```

        return;
    repaint();
}

void reset() {
    yPos = 0;
    repaint();
}

void init() {
    textFont = new java.awt.Font("Arial", Font.BOLD, 12);
    fm = getFontMetrics(textFont);
    for(int i=0;i<count;i++)
        maxLength=Math.max(maxLength,fm.stringWidth(array[i]));
    height = textFont.getSize();
    mySize = new Dimension(200, size().height);
}

public void addNotify() {
    super.addNotify();
    init();
}

public Dimension preferredSize() {
    return new Dimension(200,size().height);
}

public Dimension minimumSize() {
    return new Dimension(200,size().height);
}

public void paint(Graphics g) {
    int cx, cy;
    int length;

    g.drawRect(1,1,size().width-2,size().height-2);
    for(int i=0;i<count;i++) {
        length = fm.stringWidth(array[i]);
        cx = (maxLength - length)/2;
        cy = fm.getAscent() + fm.getHeight()*i;
        g.drawString(array[i], 5, cy-yPos);
    }
}
}

public class peetGraph extends Applet{
    String[] chemName;
    int[] normalYield;
    int[] highYield;
    int[] lowYield;
    int[] hazard;
    int maximum=0, maxHazard=0;
    newCanvas soilCanvas;
    graphCanvas barCanvas, hzBars;
    keyCanvas key;
}

```

```

headCanvas leftHead, rightHead;
Panel footer;
Panel graphs;
Scrollbar slider;
CheckboxGroup myChoice;
Checkbox box1, box2, box3;
String currentChoice;
int totalSoils=0;

public void init(){
    String inputLine;
    String tmpBuffer;
    Dimension dim;
    try {
        totalSoils=Integer.valueOf(getParameter("TOTALSOILS")).intValue();
        chemName = new String[totalSoils];
        lowYield = new int[totalSoils];
        normalYield = new int[totalSoils];
        highYield = new int[totalSoils];
        hazard = new int[totalSoils];
        chemName = new String[totalSoils];
        for(int i=0;i<totalSoils;i++) {
            chemName[i] = getParameter("SOIL"+i);
            if(chemName[i].equals("Pre-Emergence") ||
               chemName[i].equals("Spot") ||
               chemName[i].equals("Post-Emergence") ||
               chemName[i].equals("Pre-Plant Incorporated") ||
               chemName[i].startsWith("+")) {
                lowYield[i]=-1;
                normalYield[i]=-1;
                highYield[i]=-1;
                hazard[i]=-1;
            }
            else {
                lowYield[i] =
                    Integer.valueOf(getParameter("LOW"+i)).intValue();
                normalYield[i] =
                    Integer.valueOf(getParameter("TYPICAL"+i)).intValue();
                highYield[i] =
                    Integer.valueOf(getParameter("HIGH"+i)).intValue();
                tmpBuffer=getParameter("LOSS"+i);
                if(tmpBuffer.equals("<1"))
                    hazard[i] = 0;
                else
                    hazard[i]=
                        Integer.valueOf(getParameter("LOSS"+i)).intValue();
            }
            maximum = Math.max(maximum, highYield[i]);
            maxHazard=Math.max(maxHazard, hazard[i]);
        }
        maximum = (maximum/50 + 1)*50;
        maxHazard= (maxHazard/50 + 1)*50;

        // Set background color
        setBackground(Color.white);
    }
}

```

```

// Create new panel to place graphs and soilnames.
graphs = new Panel();
graphs.setLayout(new BorderLayout());

// Create canvas to place soilnames
soilCanvas = new newCanvas(chemName, totalSoils);
graphs.add("Center",soilCanvas);
dim=new Dimension(soilCanvas.preferredSize());

// Create canvas to draw bar graphs for cost.
barCanvas = new graphCanvas(normalYield, totalSoils, 200, maximum, 1,
                             Color.blue);
graphs.add("West",barCanvas);

// Create canvas to draw bar graphs for hazard
hzBars = new graphCanvas(hazard, totalSoils, 200, maxHazard, 2, Color.red);
graphs.add("East",hzBars);

// Create panel for placing choice
footer = new Panel();
footer.setLayout(new FlowLayout());

// add choice to the panel
myChoice = new CheckboxGroup();
box1 = new Checkbox("Low Yield", myChoice, false);
box2 = new Checkbox("Normal Yield", myChoice, true);
box3 = new Checkbox("High Yield", myChoice, false);
currentChoice = "Normal Yield";

key = new keyCanvas("Normal Yield");
footer.add(new Label("Enter Yield Type: "));
footer.add(box1);
footer.add(box2);
footer.add(box3);

// Create panel for placing the header.
Panel header = new Panel();
header.setLayout(new BorderLayout());
leftHead = new headCanvas(maximum,1);
header.add("West",leftHead);
rightHead = new headCanvas(maxHazard,2);
header.add("East",rightHead);

// Create a scrollbar for the applet.
slider = new Scrollbar(Scrollbar.VERTICAL, 0,0,0,totalSoils-13);

setLayout(new BorderLayout()); // Set layout for the applet

add("Center",graphs); // adds graph panel to the applet
add("East",slider); // adds scrollbars to the applet
add("North", header); // adds header to the applet
add("South", footer); // adds footer to the applet

}catch(NumberFormatException ne) {
    System.err.println("Number Format exception: " + ne);

```

```

    }
}

public boolean handleEvent(Event e) {
    if(e.target instanceof Scrollbar) {
        int yPos = slider.getValue();
        soilCanvas.redraw(yPos);
        barCanvas.redraw(yPos);
        hzBars.redraw(yPos);
    }
    else if(e.target instanceof Checkbox) {
        if(myChoice.getCurrent().getLabel().equals(currentChoice))
            return true;
        if(myChoice.getCurrent().getLabel().equals("Normal Yield")) {
            barCanvas.reInit(normalYield, totalSoils, 200, maximum, 1,
                Color.blue);
            leftHead.reset(maximum, 1);
        }
        else if(myChoice.getCurrent().getLabel().equals("High Yield")) {
            barCanvas.reInit(highYield, totalSoils, 200, maximum, 1, Color.blue);
            leftHead.reset(maximum, 1);
        }
        else if(myChoice.getCurrent().getLabel().equals("Low Yield")) {
            barCanvas.reInit(lowYield, totalSoils, 200, maximum, 1, Color.blue);
            leftHead.reset(maximum, 1);
        }
        currentChoice = myChoice.getCurrent().getLabel();
        soilCanvas.reset();
        hzBars.reset();
        slider.setValue(0);
        return true;
    }
    return false;
}
}
}

```


Appendix IV

Sever Side Graphics Generation Code

```
#include <stdio.h>
#include <stdlib.h>

#include "gd.h"

/* Define Constants */

#define SEG_H    5
#define SEG_W    12
#define GRAPH_HEIGHT 5*3*16
#define GRAPH_WIDTH 12*3*8

/* Global variables for color */
int white;
int blue;
int red;
int cyan;
int yellow;
int green;
int black;

struct {
    int range;
    char level1[5];
    char level2[5];
    char level3[5];
} levels[] = {
    { 150, " 50", "100", "150" },
    { 300, "100", "200", "300" },
    { 450, "150", "300", "450" },
    { 600, "200", "400", "600" },
    { 900, "300", "600", "900" },
    { 1200, "400", "800", "1200" }
};

/* Draw rectangle */
void DrawRectangle(gdImagePtr im, int x1, int y1, int x2, int y2, int color ) {
    gdImageLine( im, x1, y1, x2, y1, color );
    gdImageLine( im, x2, y1, x2, y2, color );
    gdImageLine( im, x2, y2, x1, y2, color );
    gdImageLine( im, x1, y2, x1, y1, color );
}

/* Plot a rank mark at the specified place */
void PlotAtXY(gdImagePtr im, int x1, int y1, char ch) {
    char str[2];
    char blankbuf[3];
```

```

x1 = (x1 - 1) * 8 + 32 + 1;
y1 = GRAPH_HEIGHT - y1 * 16 + 10;

str[1] = '\0';
str[0] = ch;

sprintf( blankbuf, "%c", 127 );
gdImageString( im, x1, y1, blankbuf, white );

gdImageString( im, x1, y1, str, black );
}

/* Draw the dimension labels */
void DrawDims(gdImagePtr im,int max_y,int y) {
    int i, line;

    /* find the level */
    for ( i=0; i<6; i++)
        if ( levels[i].range == max_y )
            break;
    if ( i == 6 ) /* can not find, then use the last one */
        i = 5;

    /* draw Y axel dimension */
    if ( i != 5 )
        gdImageString( im, 4, 0+10, levels[i].level3, black );
    else
        gdImageString( im, 0, 0+10, levels[i].level3, black );
    gdImageString( im, 4, SEG_H*16+10, levels[i].level2, black );
    gdImageString( im, 4, 2*SEG_H*16+10, levels[i].level1, black );
    gdImageString( im, 4, (3*SEG_H-1)*16+10, " 0", black );

    /* draw X axis dimension */
    gdImageString( im, 28, 3*SEG_H*16+10+2, "1%", black );
    gdImageString( im, 28+(SEG_W-1)*8, 3*SEG_H*16+10+2, "10%", black );
    gdImageString( im, 28+(SEG_W*2-1)*8, 3*SEG_H*16+10+2, "100%", black );
    gdImageString( im, 28+(SEG_W*3-2)*8, 3*SEG_H*16+10+2, "1000%", black );

    /* draw 'Groundwater Hazard' under the X-axis */
    gdImageString( im, 28+(SEG_W-3)*8, 3*SEG_H*16+10+2+16+4,
        "Groundwater Hazard", black );

    /* draw the NO TREATMENT line */
    line = y * SEG_H*3 / atoi(levels[i].level3);
    if ( ( y*SEG_H*3 % atoi(levels[i].level3) ) != 0 )
        line = line + 1;
    if ( line == 0 )
        line = SEG_H*3*16 + 10;
    else
        line = (SEG_H*3 - line)*16 + 10 + 8;
    gdImageLine( im, 28+4+2, line, 28+SEG_W*3*8, line, red );
}

int main(int argc,char* argv[]) {

```

```

FILE *out;
gdImagePtr im;
char blankbuf[21];
int ch, x, y;
char buf[81];
FILE *fp;
char rankfile[31], giffile[31];

im = gdImageCreate( GRAPH_WIDTH+32+32, GRAPH_HEIGHT+32+20 );

white = gdImageColorAllocate(im, 255, 255, 255);
red = gdImageColorAllocate(im, 255, 0, 0);
blue = gdImageColorAllocate(im, 0, 0, 255);
cyan = gdImageColorAllocate( im, 255, 0, 255 );
green = gdImageColorAllocate( im, 0, 255, 0 );
yellow = gdImageColorAllocate( im, 255, 255, 0 );
black = gdImageColorAllocate( im, 0, 0, 0 );

DrawRectangle( im, 32, 10, GRAPH_WIDTH+31, GRAPH_HEIGHT+10, blue );

gdImageLine(im,32+GRAPH_WIDTH/3,10,32+GRAPH_WIDTH/3,10+GRAPH_HEIGHT,blue );
gdImageLine(im,32+GRAPH_WIDTH*2/3,10,32+GRAPH_WIDTH*2/3,10+GRAPH_HEIGHT,blue);
gdImageFillToBorder( im, 80, 100, blue, green );
gdImageFillToBorder( im, 130, 100, blue, cyan );
gdImageFillToBorder( im, 230, 100, blue, yellow );

strcpy( rankfile, "/usr/DISK2/tmpdir/rankFile.txt.");
strcat(rankfile,argv[1]);

fp = fopen( rankfile, "r+" );
if ( fp == NULL ) {
    printf("Unable to open rank file\n");
    exit(1);
}

/* get the dimension info from file and draw dimensions */
fgets( buf, 80, fp );
sscanf( buf, "%d %d", &x, &y );
DrawDims( im, x, y );

while( fgets( buf, 80, fp ) != NULL ) {
    if ( buf[0] == '#' )
        break;
    sscanf( buf, "%d %d %d", &ch, &x, &y );
    ch = ch - 1 + 65;
    PlotAtXY( im, x, y, ch );
}

sprintf( giffile, "%s/rank.gif.",getenv("GIF_PATH"));
strcat(giffile,argv[1]);

out = fopen( giffile, "w+" );
if(!out) {
    printf("Unable to construct Rank GIF %s\n", giffile);
    exit(-1);
}

```

```
    }  
    gdImageGif( im, out );  
    fclose( out );  
    gdImageDestroy( im );  
    return 0;  
}  
  
void replLF(char *mybuffer) {  
    int i;  
  
    for(i=0;i<81;i++)  
        if(mybuffer[i]=='\n')  
            mybuffer[i]='\0';  
}
```

VITA ²

Aniruddha Gujrathi

Candidate for the Degree of

Master of Science

Thesis: DATA REPRESENTATION FORMATS ON WORLD WIDE WEB USING
DECISION SUPPORT SYSTEMS

Major Field: Computer Science

Biographical:

Personal Data: Born in Pune, India on May 31, 1973, the son of Ashok and Prabha Gujrathi.

Education: Graduated from N. Wadia College of Arts & Science, Pune in May 1990; received Bachelor of Engineering in Computer Science and Engineering from D.Y. College of Engineering, University of Pune, Pune in May 1994. Completed requirements for the Master of Science degree with major in Computer Science at Oklahoma State University in May 1999.

Experience: Worked for Tata Engineering and Locomotive Company (TELCO), Pune, India as systems engineer, July 1994 to July 1996; employed by Oklahoma State University, Department of Agronomy as a graduate research assistant, Dec 1996 to Feb. 1998