A WWW INTERFACED DECISION SUPPORT SYSTEM

THAT INTEGRATES RDBMS AND GIS

WITH WWW SERVER

By

TAO ZHU

Bachelor of Science

Peking University

Beijing, P. R. China

1991

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, 1997

A WWW INTERFACED DECISION SUPPORT SYSTEM

THAT INTEGRATES RDBMS AND GIS

WITH WWW SERVER

Thesis Approved:

H. Lu

Thesis Advisor

D. C. Wyatt

M. E. Hodgson

Thomas C. Collins

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my major advisor, Dr. Huizhu Lu, for her intelligent supervision, constructive guidance, inspiration, encouragement, and friendship.

Many thanks go to Dr. David Nofziger for providing me the opportunity to work on the project, and the knowledgeable instructions throughout the whole process.

Sincere thanks also go to Dr. George Hedrick for providing precious advice and instructions during my thesis work .

Special thanks go to the GIS specialist Mr. Mark Gregory at the Agronomy Department of Oklahoma State University, for patiently guiding me in learning and using GRASS.

Finally, I would like to express my special appreciation to my wife, Yan Gu, for her love, encouragement, and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

CMLS (Chemical Movement in Layered Soils) is a simulation tool used in managing agriculture chemicals. Many scientists and government agencies such as Oklahoma Department of Agriculture interface CMLS with a Geographical Information System (GIS) package to produce maps from the CMLS output. The maps generated can be used to evaluate the risk of ground water contamination for specific soil-pesticide-water management systems [19]

Fengxia Ma [19] created a shell program, go & draw, to provide a user interface on UNIX system, and run CMLS to generate results, which can be used to generate maps from GRASS (Geographic Resource Analysis Support System) GIS package. But running CMLS is very time consuming (e.g. doing the simulation for the entire Oklahoma state on the mainframe in Oklahoma State University takes more than 100 CPU hours), and space consuming (e.g. it needs approximately 100MB disk space for each county, some counties even need 200MB) It is not feasible to run CMLS as an on-line information system.

An alternative method is to run CMLS simulation in advance, and save the simulation results in files in a certain format. J. S. Chen et al [23] saved the simulation results in binary files, and created a user friendly DOS-based decision support system

called MAPS. MAPS can accept user's inputs, generate the results of interest, and draw maps on the PC screen. MAPS provides image drawing functionality itself by using the C language, so it is very fast to render a map on the screen. The drawbacks of MAPS are:

(1) It is a single user DOS-based system;

(2) Users are involved in the installation on their PCs (with those large data files which take a lot of disk space);

(3) If there are any improvements or modifications to the system, users must update their system on their machines;

(4) The program provides very limited capabilities for data searching and data operations for both simulation data and GIS data.

(5) It's a large complex program to maintain.

The World Wide Web (WWW) technology allows users from anywhere at anytime to access the information managed by the WWW server. 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 relational data storage, query, and operations. GRASS is a powerful GIS package which has a full range of functions for GIS data storage, calculations, and operations. The project MAPS for WWW, which was proposed for the thesis, utilizes the WWW Common Gateway Interface (CGI), calculates the results by accessing ORACLE

RDBMS; then uses GRASS to generate maps. It can overcome the problems described above.

The goal of the thesis is to design and implement a WWW version Client/Server Decision Support System called MAPS for WWW, in which a GIS database system (GRASS) and a relational database management system (ORACLE) work with WWW server (NCSA httpd 1.3) to provide decision support information to WWW users. This project can also serve as a framework to many other database related WWW applications.

Chapter II reviews the literature of the related components used in the system. Chapter III describes the application background of the decision support system, including GIS data, database schema and simulation. Details of design and implementation of the system are given in Chapter IV. Finally, the conclusions and future work recommendations are given in Chapter V

Several appendices are included for reference. Appendix A includes the major CGI program of the system. Appendix B is the source code of a Pro*C program that accesses ORACLE RDBMS to do the calculation for Travel Time according to user's inputs. The source code of a stored procedure used in the system to get the soil name from a given soil index is given in Appendix C. Appendix D is the C source code for generating legend GIF picture for Travel Time simulation. A sample output screen of Travel Time simulation is included in Appendix E.

CHAPTER II

LITERATURE REVIEW

As described in Chapter I, MAPS for WWW is a WWW version Client/Server decision support system, which integrates ORACLE RDBMS and GRASS GIS package to work with the WWW server. All these systems run on the UNIX operating system. Bourne Shell is the UNIX shell language used for CGI programming in the system. This chapter reviews these software and systems used in the project, and also gives a brief review of the simulation model.

2.1 Simulation Model

The simulation model used in this project was published by Dr. D. L. Nofziger, et al [23]. It can be used to assist farmers and researchers in managing pesticides to minimize the risk of ground water quality degradation. Through the simulation model, a user can calculate Travel Time, Amount Leached and Groundwater Hazard at different probability levels, and POE (Probability Of Exceeding health advisory level). As described in Chapter I, the simulation data used for the model was generated by running CMLS in advance, and saved in files. More details about the simulation model, such as database schema and computational formulas, will be described in chapter III.

2.2 World Wide Web

The World Wide Web (WWW) was first developed at a particle physics lab called CERN (acronym from French: Conseil Europeen pour la Recherche Nucleaire) in Geneva, Switzerland. Tim Berners-lee led the development work, beginning in 1989. From then on, a lot of people around the Internet community started to develop their own browsers, supporting additional platforms and developing new features. Currently, WWW servers and WWW browsers support all computer architectures and operating systems [17]. From a user's perspective, the WWW is a collection of documents, or pages, which contains text, images, and hypertext links to other pages. WWW merges the techniques of information retrieval and hypertext to create a powerful global information system. By simply pointing and clicking, the user has instant access to a big collection of information, distributed around the globe. From an information provider's point of view, the WWW is an easy and efficient way of distributing information to a very large audience [6].

Within the WWW, ASCII text documents are marked up with a tagging language called HyperText Markup Language (HTML) HTML is a simple data format used to create hypertext documents that are portable from one platform to another. A WWW browser is actually a browser of HTML documents. WWW browsers and WWW servers communicate using HyperText Transport Protocol (HTTP). HTTP is an application level protocol between a WWW client and a WWW server [17]. When a WWW browser requests an HTML document from a WWW server, a connection is opened. Then the WWW server will grab the specified document and send it back to the WWW browser

After the document is transported, the connection is closed. Each request for a document from a WWW browser to a WWW server is a new connection [17].

A WWW server not only can handle HTML documents, but also can access other services through an interface called Common Gateway Interface (CGI). When a user inputs data from a WWW browser and clicks on a button to send the request, the user inputs are sent to the WWW server. The CGI program can read user's inputs through CGI interface from the WWW server, then access other information systems, and return back the results in HTML form to the WWW server. The output from the CGI program then will be returned by the WWW server to the WWW browser through the Internet. So, the user will see the results on the WWW browser. Figure 2-1 shows the information flow through the WWW and CGI.

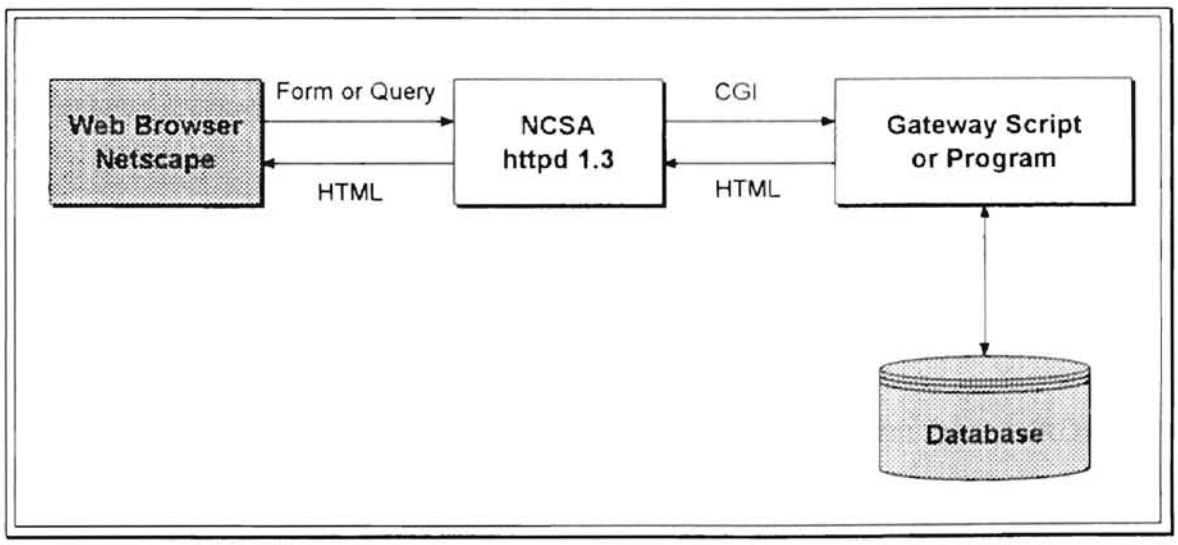


Figure 2-1 Information flow through Common Gateway Interface (Source from [17])

HTML supports a certain level of user interface programming. It provides many user interface controls like: radio box, check box, select list, text input and button. We used these controls to design our screens for users to input data. The following are some examples of these controls :

```
<!-- The following is a text input box -->
<INPUT TYPE="text" NAME="USERID" VALUE="HELLO">

<!-- The following is a Radio Box -->
<INPUT TYPE="radio" NAME="CHOICE" VALUE="Yes" CHECKED> Radio Box

<!-- The following is a Check Box -->
<INPUT TYPE="checkbox" NAME="SIZE" VALUE="Big"> Check Box

<!-- The following is a selection list -->
<SELECT NAME="SELECTION">
    <OPTION SELECTED>Option1
    <OPTION>Option2
</SELECT>

<!-- The following is a button used to submit request -->
<INPUT TYPE="SUBMIT" VALUE="Run">
```

2.3 ORACLE

ORACLE is a very powerful Relational Database Management System (RDBMS). It fully supports ANSI SQL, and has its own extended SQL -- PL/SQL (Procedural Language extension of SQL) [4]. It provides many ways for users to access the server, such as SQL*Plus, SQL*Report, SQL*Forms etc It also provides libraries and preprocessors to allow users to embed SQL into a host language, such as C, PASCAL,

ADA, FORTRAN etc. This makes it convenient for developers to write some standalone programs to access the RDBMS. In this project, we used Pro*C (embed SQL into C language) to write our CGI programs to access the ORACLE server. The following are some of the ORACLE components used in this project :

- SQL*Plus: a utility program that provides developers and end users the ability to interact directly with the database. It has a command-line interpreter where users can directly submit SQL commands, SQL*Plus commands and PL/SQL blocks.

- SQL*Loader: a tool for loading external data files into one or more ORACLE tables. It can load data in a variety of data formats.

- Pro*C: ORACLE supports embedded SQL statements and PL/SQL blocks in a host language, like C, PASCAL, ADA, FORTRAN etc. Pro*C is a preprocessor for embedded C programs.

- PL/SQL and Stored Procedure SQL is a non-procedural language PL/SQL is ORACLE's Procedural Language extension to SQL. Procedures or functions written by PL/SQL can be stored in the database. Stored procedures can enhance the performance of the database server.

2.4 GRASS

GRASS is a public domain, image processing, geographic information system (GIS), written in the C language and running under the UNIX operating system. It was originally designed and developed by researchers and engineers in the U. S. Army Construction Engineering Research Laboratory (CERL). GRASS stands for "Geographic Resources Analysis Support System" [10]. It is used extensively at government offices, universities, and commercial organizations. The following are some of the commands we use in the project:

- `g.region -- program to manage the boundary definitions for the geographic region.`
- `g.gisenv -- setup or output the current GRASS environment settings`
- `g.remove -- removes database element files from current mapset.`
- `p.select ppm -- select PPM as hardcopy output format`
- `p.map -- command language interface to color hardcopy and graphics monitor output.`
- `r.reclass -- create a new map layer whose category values are based upon the user's reclassification of categories in an existing raster map layer.`
- `r.colors -- creates/modifies the color table associated with a raster map layer.`

2.5 UNIX and Shell programming

UNIX is a well known multi-user time sharing operating system, which was first developed by Ken Thompson in 1969 at AT&T Bell Laboratories in Murray Hill, New Jersey. It is widely used in universities, government organizations and industry [26]. A shell is not only a command interface to UNIX operating system, but also a very powerful programming language. It supports many high-level language constructs, such as variables, flow control structures, parameter passing, subroutine calls and interrupt handling. There are three major UNIX shells: they are Bourne shell, C shell, and Korn shell [26]. The Bourne Shell was used in this project to write CGI programs.

CHAPTER III

A DECISION SUPPORT SYSTEM

This chapter describes the decision support system, including GIS data used, database schema, and formulas for calculations.

3.1 GIS Data

The soils maps and land use maps of the counties in Oklahoma were collected by the GIS experts from the Agronomy Department of Oklahoma State University. These maps are stored in GRASS GIS database system on a Sun workstation. There are 5 different kinds of maps in our GRASS database for each county in Oklahoma. They are listed in the following table :

(NOTE $COUNTYNAME represents a name of a county, e.g. CADDO)

File Name	Description
$COUNTYNAME.soils	Soils map of the county $COUNTYNAME
$COUNTYNAME.landuse	Land use map of the county $COUNTYNAME
$COUNTYNAME.crop	Cultivated area map of the county $COUNTYNAME
$COUNTYNAME.irrigated	Irrigated area map of the county $COUNTYNAME
$COUNTYNAME.boundary	Vector boundary map of the county $COUNTYNAME

Table 3-1. GIS data in the system

3.2 Simulation Data

As discussed in Chapter I, CMLS was run in advance. The results have been stored in binary format, and also in Paradox format. In this project, these data were moved to ORACLE RDBMS (about 100MB). Based on these simulation data, and the formulas that we will introduce in the next section, users can view the Travel Time, Amount Leached and Groundwater Hazard at different probability levels, and POE (Probability of Exceeding health advisory level).

The following is the schema of the simulation database:

```
/*-------------------------------------------------------------*/
/* Table Name : CHEMOD                                         */
/* Description: This table contains the related parameter of */
/*              a certain active ingredient for a pesticide. */
/*-------------------------------------------------------------*/
CREATE TABLE CHEMOD
(
    CommonName     VARCHAR2(31)   PRIMARY KEY,
    Koc            NUMBER         NOT NULL,
    HalfLife       NUMBER         NOT NULL,
    HALEQ          NUMBER         NOT NULL
);


/*-------------------------------------------------------------*/
/* Table Name : COUNTY                                         */
/* Description: This table maps county index to its name    */
/*-------------------------------------------------------------*/
CREATE TABLE COUNTY
(
    CountyIndex    NUMBER(6)      PRIMARY KEY,
    CountyName     VARCHAR2(12)   NOT NULL
);


/*-------------------------------------------------------------*/
/* Table Name : TRAVEL                                         */
/* Description: This table contains the main simulation data*/
/*              generated from CMLS simulation tool.         */
/*-------------------------------------------------------------*/
CREATE TABLE TRAVEL
(
    CountyIndex     NUMBER(6),
    SoilIndex       NUMBER(6),
    CropName        VARCHAR2(8),
    ApplDate        VARCHAR2(9),
    IrrigationType  VARCHAR2(8),
    Probability     NUMBER(6),
```

```
    Koc0            NUMBER(6),
    Koc1            NUMBER(6),
    Koc2            NUMBER(6),
    Koc4            NUMBER(6),
    Koc7            NUMBER(6),
    Koc10           NUMBER(6),
    Koc20           NUMBER(6),
    Koc40           NUMBER(6),
    Koc70           NUMBER(6),
    Koc100          NUMBER(6),
    Koc200          NUMBER(6),
    Koc400          NUMBER(6),
    Koc700          NUMBER(6),
    Koc1000         NUMBER(6),
    Koc2000         NUMBER(6),
    Koc4000         NUMBER(6),
    Koc7000         NUMBER(6),
    Koc10000        NUMBER(6),

    PRIMARY KEY (   CountyIndex,
                    SoilIndex,
                    CropName,
                    ApplDate,
                    IrrigationType,
                    Probability )
);


/*-------------------------------------------------------------*/
/* Table Name : GRASSID                                        */
/* Description: This table is a bridge between the soil in     */
/*              the RDBMS to the soil in GRASS GIS package     */
/*-------------------------------------------------------------*/
CREATE TABLE GRASSID
(
    Muid            VARCHAR2(7),
    CountyIndex     NUMBER(6),
    SoilIndex       NUMBER(6),
    GrassIndex      NUMBER(6)
);


/*-------------------------------------------------------------*/
/* Table Name : SOILNAME                                       */
/* Description: This table maps soil index to its name         */
/*-------------------------------------------------------------*/
CREATE TABLE SOILNAME
(
    CountyIndex     NUMBER(6),
    SoilIndex       NUMBER(6),
    Muid            VARCHAR2(7),
    SoilName        VARCHAR2(46),
    Muname          VARCHAR2(90),

    PRIMARY KEY (   CountyIndex,
                    SoilIndex )
);


/*-------------------------------------------------------------*/
```

```
/* Table Name : UNIT                                                    */
/* Description: Units and their conversion                              */
/*----------------------------------------------------------------------*/
CREATE TABLE UNIT
(
    UnitType        VARCHAR2(1),
    OldUnit         VARCHAR2(7),
    StdUnit         VARCHAR2(7),
    Conversion      NUMBER,

    PRIMARY KEY (   UnitType,
                    OldUnit )
);


/*----------------------------------------------------------------------*/
/* Table Name : TRADCOMM                                                */
/* Description: Pesticides information and their ingredients*/
/*----------------------------------------------------------------------*/
CREATE TABLE TRADCOMM
(
    TradeName       VAPCHAR2(46),
    CommonName      VARCHAR2(31),
    ProdUnit        VARCHAR2(7),
    AIlb_ProdUnit   NUMBER,

    PRIMARY KEY (   TradeName,
                    CommonName )
);


/*----------------------------------------------------------------------*/
/* View Name : CHEMICAL                                                 */
/* Description : Joint view of TRADCOMM and CHEMOD                      */
/*----------------------------------------------------------------------*/
CREATE VIEW CHEMICAL AS
SELECT TradeName, a.CommonName, ProdUnit, AIlb_ProdUnit,
       Koc, HalfLife, HALEQ
  FROM TRADCOMM a, CHEMOD b
 WHERE a.CommonName = b.CommonName;


/*----------------------------------------------------------------------*/
/* View Name : TRAVELID                                                 */
/* Description : Joint view of TRAVEL and GRASSID                       */
/*----------------------------------------------------------------------*/
CREATE VIEW TRAVELID AS
SELECT a.*, GrassID
  FROM Travel a, GrassID b
 WHERE a.CountyIndex = b.CountyIndex
   AND a.SoilIndex = b.SoilIndex;
```

3.3 Computational Formulas

Computational formulas are provided as the following to calculate Travel Time, Amount Leached, Groundwater Hazard, and POE :

Travel Time:

Calculate Travel Time t for the chemical of interest by interpolating between the travel times of the interest neighbors using the following equation.

$$t = (K_{oc} - K_{oc1}) \frac{t_2 - t_1}{K_{oc2} - K_{oc1}} + t_1$$

the K_{oc} is the partition coefficient for the active ingredient of interest, K_{oc1} and K_{oc2} are the nearest neighbors of K_{oc} in the TRAVEL table where $K_{oc1} <= K_{oc} < K_{oc2}$, and t_1 and t_2 are the corresponding travel times in the TRAVEL table.

Amount Leached :

Calculate Amount Leached by using the following equation :

$$\text{AmtLeached} = \text{ApplAmt} * 0.5^{T}$$

where ***ApplAmt*** is the amount of pesticide applied (in Kg/ha). And ***T*** represents the number of half-lifes which occur before the chemical reaches the critical depth. The equation for ***T*** is:

$$T = \frac{t}{t_{1/2}}$$

where $t_{1/2}$ is the half life of the chemical. The amount applied can be obtained using the following equation

$$ApplAmt = AI * ProdApplRate * 1.12 * \frac{c_2}{c_1}$$

where ***AI*** (or AIlb_ProdUnit in TRADCOMM table) is the amount of active ingredient present in each product unit (ProdUnit); c_1 is the number of standard units per product unit; ***ProductApplRate*** is the amount of product applied per acre (user input); c_2 is the number of standard units per ApplUnit, and ***1.12*** is the number converting lb/ac to Kg/ha. The constant c_1 can be obtained from UNIT table with ProdUnit as OldUnit; c_2 is the conversion factor in the database with ApplUnit as OldUnit.

Groundwater Hazard .

Calculate the Groundwater Hazard associated with a pesticide using the following equation

$$Hazard = \sum_{i=1}^{n} \frac{C_i}{HALEQ_i} = \frac{100}{DepthMixing * Porosity} \sum_{i=1}^{n} \frac{AmtLeached_i}{HALEQ_i}$$

where n is the number of active ingredients in the pesticide. Depth Mixing and Porosity are from the user inputs. HALEQ is from the CHEMICAL table associated with active ingredient of the pesticide of interest.

POE :

POE stands for Probability of Exceeding HAL (Health Advisory Level). After calculating the ground water hazard for all the given probabilities in the TRAVEL table, we will decide at which probability level the value of ground water hazard exceeds 1. The given probabilities in the TRAVEL table are: 2%, 5%, 10%, 25%, 50%, 75%, 90%, 95%, 98%. For an example, if the ground water hazard exceeds 1 at the 5% probability level, but not at the 2% level, then POE is in the range of (2% - 5%).

CHAPTER IV

DESIGN AND IMPLEMENTATION

This chapter describes the design and implementation of MAPS for WWW. In the design part, it covers the system running environment, client/server model, user interface design, and CGI programs. The implementation part describes the details of methods used to develop the system.

4.1 DESIGN

4.1.1 System Running Environment

In the project MAPS for WWW, data storage, calculations, and operations are handled at the server side. The facility requirements for a user's system are quite limited This makes the system more acceptable to users. The following table shows the system running environment at the server side, and the system requirements at the client side.

Server Side Hardware	Sun Workstation (128MB RAM)
Server Side Software	SUN OS 4.1 3 NCSA HTTPD 1 3 ORACLE 7.2 GRASS 4.1
Client Side requirements	Any platform Internet Connection HTML 3.0 Compatible Browser

Table 4-1 System Requirements

4.1.2 Client/Server Model

In MAPS for WWW, three different systems are integrated -- WWW server, ORACLE, and GRASS. Figure 4-1 shows the client/server working model and the relationships among these three systems.

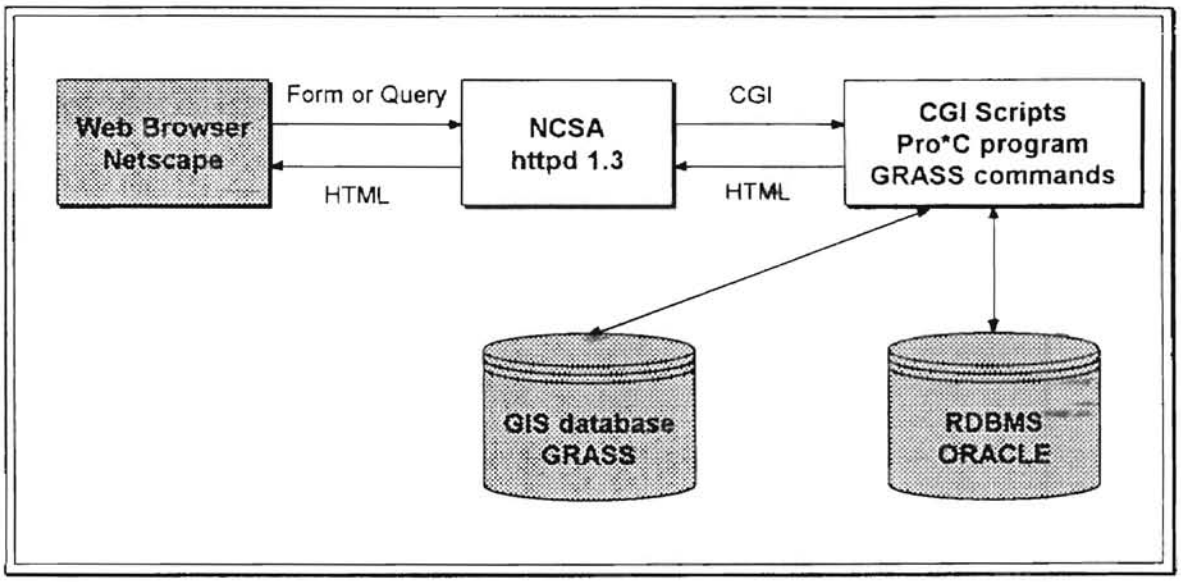


Figure 4-1 Client/Server Model of MAPS for WWW

4.1.3 User Interface Design

MAPS for WWW is screen oriented Users define inputs by means of a WWW browser, and click on a button to go to the next screen (including the final output screen). Figure 4-2 shows the hierarchy of the screens in the system.

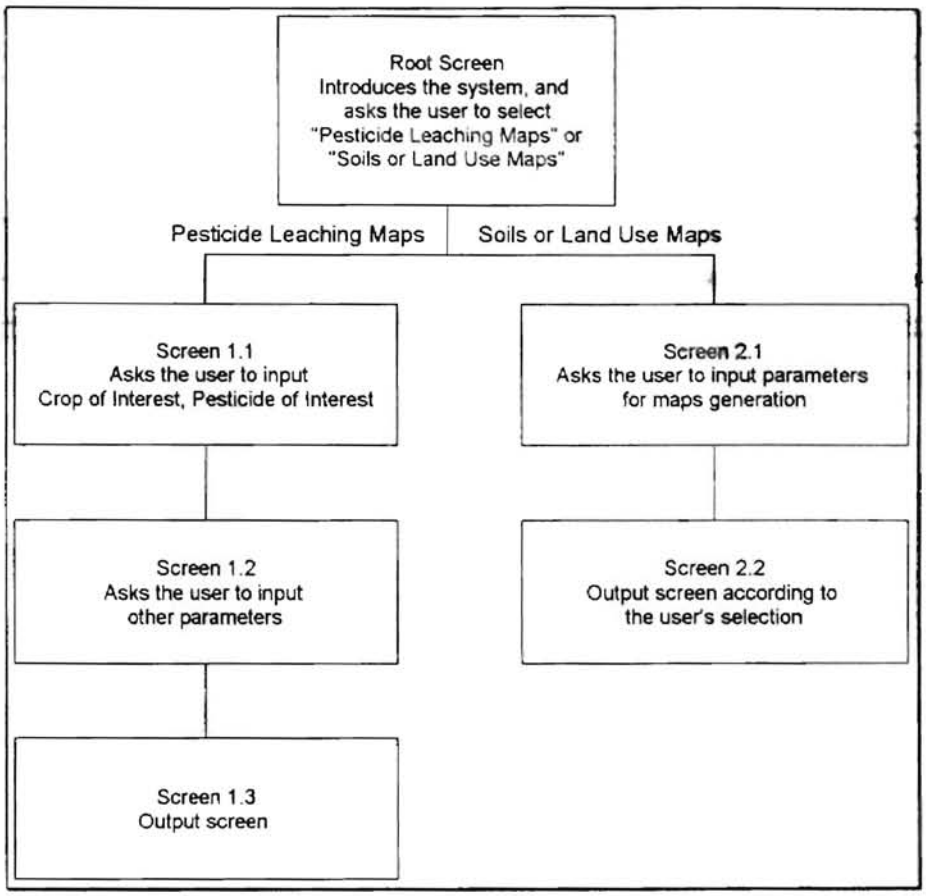


Figure 4-2 Outline of the screens

Root Screen :

This is the first screen of MAPS for WWW. It shows some welcome words, a brief description of the project, names of the developers, and the usage of the system. There are two buttons after the description part: "Soil or Land Use Maps", and "Pesticide Leaching Maps" If the user selects the "Soil or Land Use Maps" button, it leads them to Screen 2.1, which enables the user to generate the Soils or Land Use maps of interest. If the user selects the "Pesticide Leaching Maps" button, it leads them to Screen 1.1, which will enable the user to define the soil-chemical system of interest. Then the system

will do the calculation according to the inputs and the computational formulas, and generate the maps and tables.

Screen 1.1 :

Figure 3.1 shows the user interface of Screen 1.1.

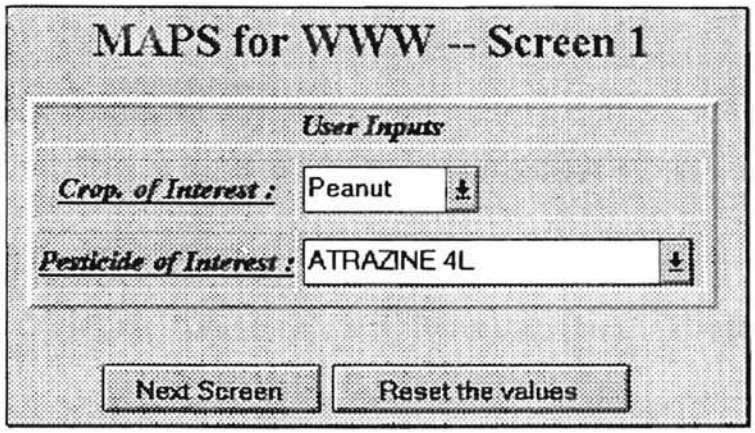


Figure 4-3 Layout of Screen 1.1

The following are the descriptions of the user inputs from Screen 1.1 :

Title	Variable Name	Help Description
Crop of Interest	MAPS_CROP	Select a crop from the list provided. This selection will determine the counties available for viewing, application dates for the pesticide, and the type and period of irrigation used.
Pesticide of Interest	MAPS_TRADENAME	Select the tradename of the pesticide from the list provided. The system will find active ingredients in that pesticide along with their properties. The properties can be examined and modified in the next screen if you have more appropriate values. Note: To be meaningful, the pesticide you select should be labeled for the crop you selected. However, you may select any pesticide from the list.

Table 4-2 User inputs from Screen 1.1

There are 2 buttons: "Next Screen", and "Reset the values". The file containing the form described above also contains on-line help for users. If the user clicks on the "Reset the values" button, the inputs will be restored to their original settings. If the user clicks on the "Next Screen" button, the system will lead the user to the Screen 1.2 for more inputs.

Screen 1.2 :

Screen 1.2 is used to define the remaining parameters for the system of interest. Figure 4-4 shows the user interface of Screen 1.2.

There are 2 buttons: "Get result", and "Reset the values". The file containing the form described above also contains on-line help for users. If the user clicks on the "Reset the values" button, the inputs will be turned back to their original settings. If the user clicks on the "Get result" button, the system will lead the user to the Screen 1.3, which is the output screen for the simulation results.

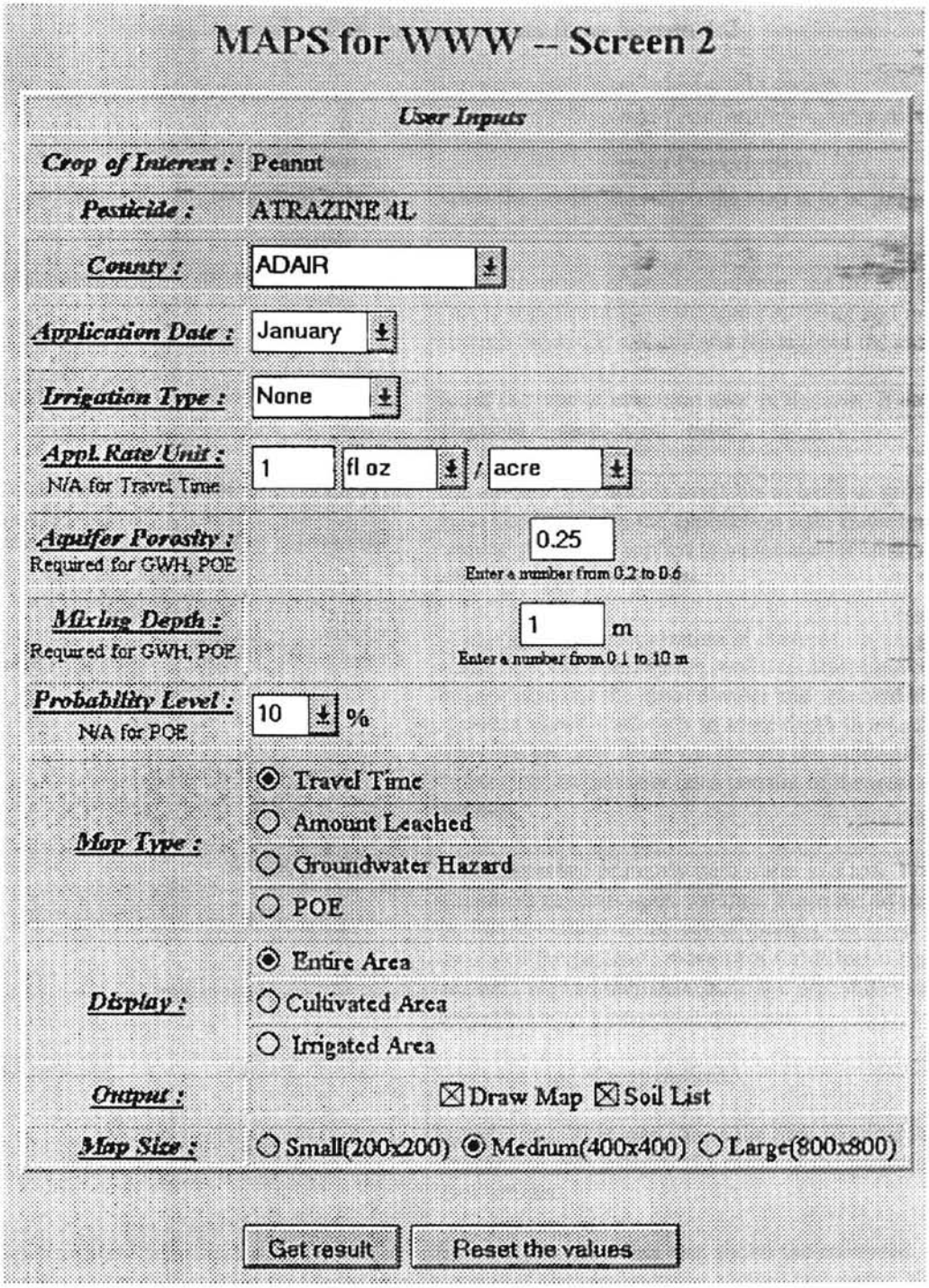


Figure 4-4 Layout of Screen 1.2

The following is a table of all the user inputs from Screen 1.2 :

Title	Variable Name	Help Description
County	MAPS_COUNTY	Select the county for which the map is to be drawn from the list of available counties.
Application Date	MAPS_APPLDATE	Select the month of application from the list provided. This is significant since the water entering and moving through the soil is not uniform throughout the year.
Irrigation Type	MAPS_IRRTYPE	Select the type of irrigation used in the area. If no irrigation is used, select "None".
Application Rate Application Unit Unit of Area	MAPS_PRODRATE MAPS_APPLUNIT MAPS_AREAUNIT	The amount of pesticide available to leach to ground water depends upon the application rate. Enter the amount of product applied to a unit area of land and select the appropriate units.
Mixing Depth Aquifer Porosity	MAPS_DEPTHMIX MAPS_POROSITY	To estimate the concentration of chemical in the ground water, we used a simulation model to determine the amount entering the ground water and assumed that the chemical mixed uniformly to some depth in an aquifer of known porosity. Here you specify the mixing depth (in meters) and porosity (as a percent of the total soil volume).
Probability Level	MAPS_PROB	Pesticides move primarily with water in a soil. Water movement depends upon weather. Since we do not know the future weather at the site of interest, we have simulated the pesticide movement for different years of weather. We can then determine the probability associated with a particular question.
Map Type	MAPS_TYPE	Select the type of map desired. Travel Time: This map shows the time required for the chemical to pass a depth of 1 meter at the probability level selected. Amount Leached: This map shows the estimated amount of chemical passing a depth of 1 meter. Note: All of the chemical passing this depth is assumed to enter the ground water at the probability level selected. Groundwater Hazard: This map shows the calculated ground water hazard associated with this pesticide at the probability level selected. This is the ratio of the estimated pesticide concentration and the HAL or MCL value for that chemical. Values in excess of 100% indicate the concentration of pesticide in the ground

		water exceeds the HAL or MCL value for that chemical. Probability of Exceeding the HAL or MCL: This map shows the probability of the estimated pesticide concentration in ground water exceeding the HAL or MCL for that material. That is, if the product is used many times with weather characteristic of this site, this map shows what percent of the applications will result in ground water concentrations in excess of the HAL or MCL.
Display	DISPLAY_FOR	Select the area of interest. Entire Area: The selected map will be displayed for all of the areas in the county containing soils in classes 1 - 4. Cultivated Area: The selected map will be displayed for only areas of the county with land uses of cultivated land or irrigated land. Irrigated Area: The selected map will be displayed only for areas of the county with land use category of irrigated land.
Output	MAPS_OUTMAP MAPS_OUTLIST	The software is capable of producing maps and tables. Select the output desired.
Map Size	GIF_SIZE	Select the map size desired. This will likely depend upon the resolution of the graphics system you are using. There are 3 options available : 200x200, 400x400, 800x800. Note: Smaller maps are created somewhat faster than large maps.

Table 4-3 User inputs from Screen 1.2

Screen 1.3 :

This is the simulation output screen. The user inputs from the previous screens determine the output contents of this screen. If the user selects "Maps" output, the image will be generated and shown along with a legend picture to show the meaning of different colors in the map. If the user selects "List" output, the calculation results for Travel Time, or Amount Leached, or Groundwater Hazard, or POE will be shown. A sample output screen for Travel Time is given in Appendix E.

Screen 2.1 :

Figure 4-5 shows the user interface for Screen 2.1.

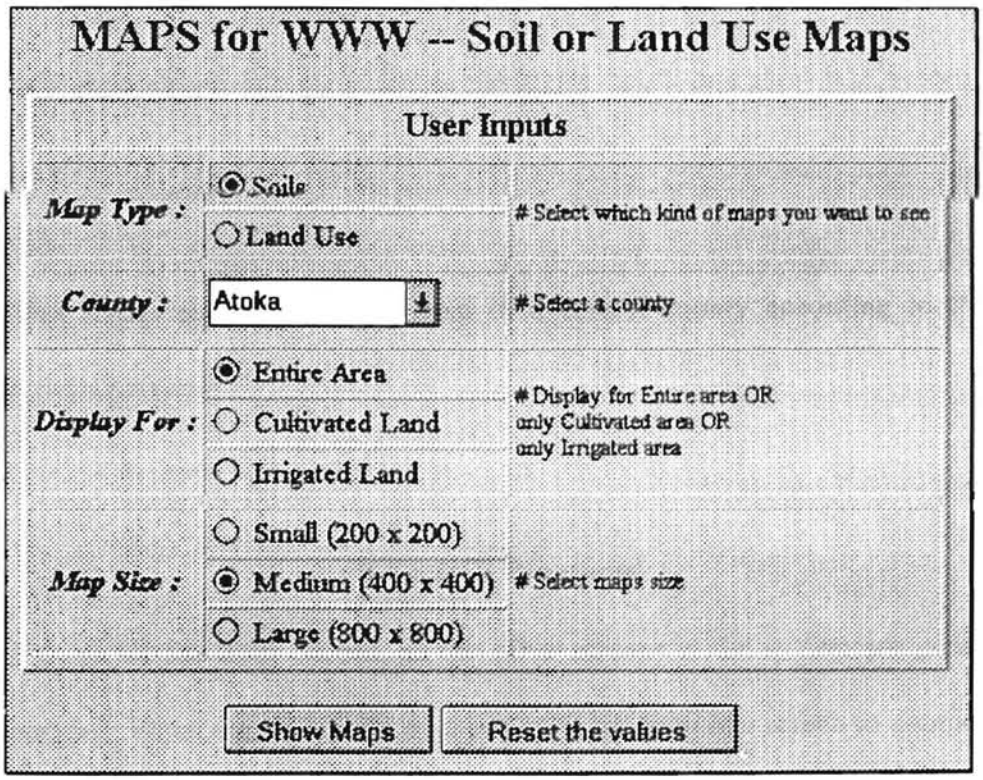


Figure 4-5 Layout of Screen 2.1

The following is a table of all the user inputs from Screen 2.1 :

Title	Variable Name	Help Description
Map Type	MAP_TYPE	Select which kind of maps you want to see.
County	COUNTY	Select a county
Display For	DISPLAY_FOR	Display for the Entire area OR only cultivated area OR only Irrigated area
Map Size	GIF_SIZE	Select map size (200x200, 400x400, 800x800)

Table **4-4** User inputs for Screen 2.1

There are also 2 buttons in Screen 2.1. "Show Maps", and "Reset the values". If the user clicks on the "Reset the values" button, the values of the user inputs will be restored to their original settings. If the user clicks on the "Show Maps" button, the system will generate the map according to the user's inputs, and show it in Screen 2.2 .

Screen 2.2 :

This screen shows the image map of a certain county according to the user's selection from Screen 2.1.

4.1.3 CGI Programs

Except for the Root Screen, all the other screens in the system are generated by CGI programs. When the user clicks on a button to go from one screen to another, a CGI program will be called by the WWW server to handle it. The following is a list of CGI programs used in the system .

From Screen	To Screen	CGI Program Called
Root Screen	Screen 1.1	maps_11.cgi
Screen 1.1	Screen 1.2	maps_12.cgi
Screen 1.2	Screen 1.3	mapswww.cgi
Root Screen	Screen 2.1	maps_21.cgi
Screen 2.1	Screen 2.2	mapsonly.cgi

Table 4-5 CGI Programs

Among these CGI programs, the most complicated and comprehensive one is mapswww.cgi. It is the CGI program that is used to generate the final simulation output screen -- Screen 1.3. It covers all the implementation techniques used in the system.

When a user gives all the inputs needed from Screen 1.2, and clicks on the "Get result" button, the request is sent to the WWW server. The WWW server will call the CGI program -- mapswww.cgi Then, the CGI program will (1) get the user's inputs; (2) check if there is anything wrong in the inputs; (3) run a corresponding Pro*C program to access ORACLE database and calculate the Travel Time or Amount Leached or Groundwater Hazard or POE; (4) run GRASS commands to access the corresponding map stored in GRASS database and reclass the map according to the results from the previous calculation; (5) convert the map from PPM (Portable PixMap) format to GIF format and generate a legend GIF file for it; (6) return the maps and results back in HTML format to WWW server through CGI. The WWW server then returns the results to the user. The control flow chart of mapswwww.cgi is given below as Figure 4-6.

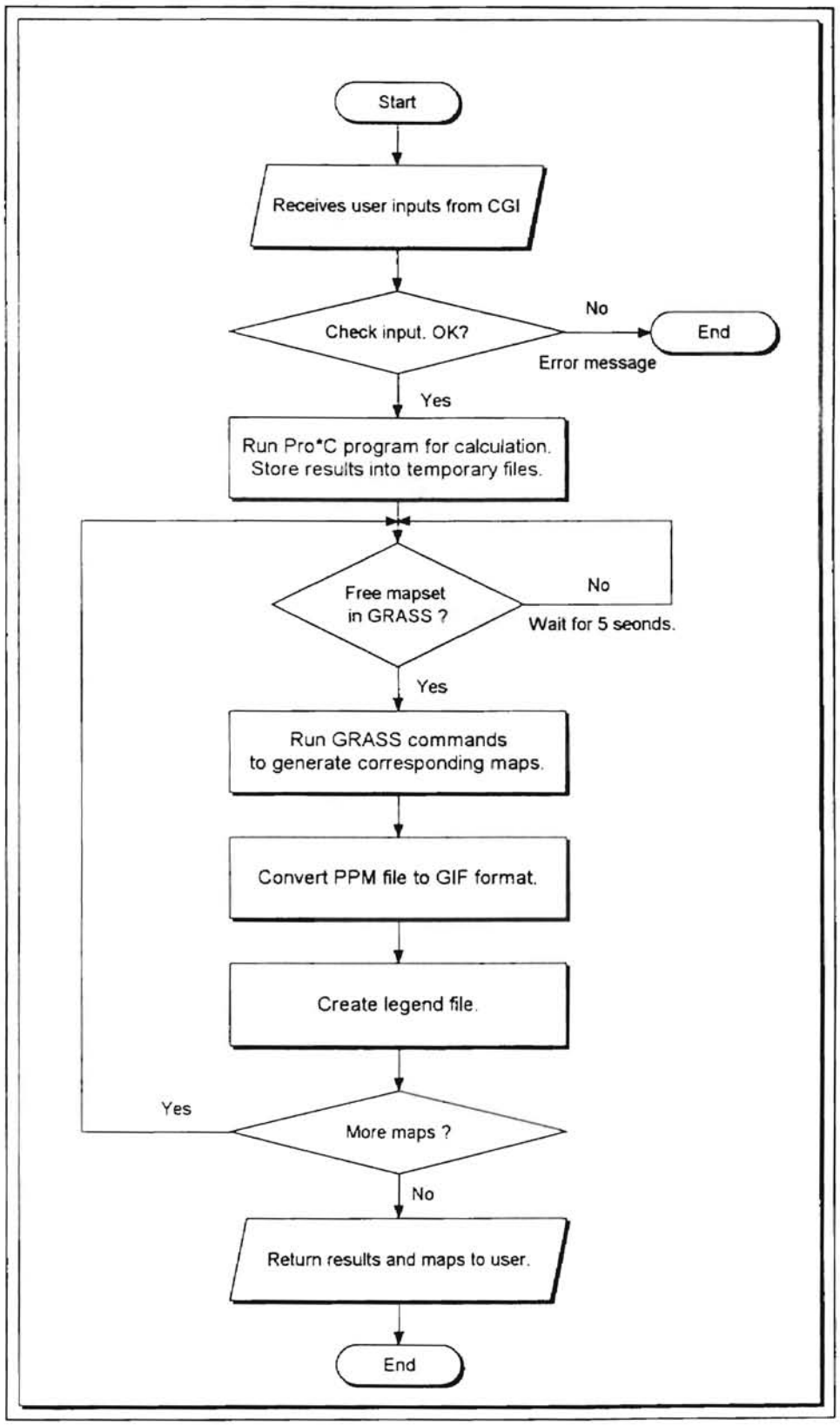


Figure 4-6 Control Flow for mapswww.cgi

4.1.4 Pro*C programs

ORACLE allows programmers to embed SQL statements into C language. In this project, all the database query and calculations are done by Pro*C programs. The following is a list of the Pro*C programs developed in the project :

Pro*C Program	CGI Program that runs it	Description
mapsim.pc	maps_11.cgi	Generate pesticide list for Screen 1.1
calc.pc	maps_12.cgi	Generate certain inputs (County, Application Date, Irrigation Type, Product Rate and Unit, Area Unit) for Screen 1.2, according to user's inputs in Screen 1.1
travel.pc	mapswww.cgi	Calculate Travel Time according to user's inputs, and generates results into temporary files.
amtleach.pc	mapswww.cgi	Calculate Amount Leached according to user's inputs, and generates results into temporary files.
gwhazard.pc	mapswww.cgi	Calculate Groundwater Hazard according to user's inputs, and generates results into temporary files.
poe.pc	mapswww.cgi	Calculate POE according to user's inputs, and generates results into temporary files.

Table 4-6 Pro*C Programs

4.2 IMPLEMENTATION

This project integrates a WWW server, an ORACLE RDBMS, and the GRASS GIS package on the UNIX operating system. In the implementation of MAPS for WWW, many technical problems must be solved. The following are the implementation details of the methods used to solve these problems. They are the keys to implement MAPS for WWW.

4.2.1 Get user inputs from CGI

There are two ways for a WWW server to transfer user data to CGI programs. They are the GET method and the POST method. The GET method uses an environment variable QUERY_STRING to hold all the user inputs separated by an '&' mark. The POST method sends the user inputs to STDIN separated by an '&' mark, so a CGI program can read from STDIN to get the data. The environment variable CONTENT_LENGTH holds the length of the user input data. The GET method is good for small amounts of input, while the POST method is preferred for large amounts of data. Some people even suggest using only the POST method [17]. We used the POST method in the system.

A C program was developed to receive a user's data from the CGI interface and save them into a temporary file. The C program reads from STDIN, separates data by '&' mark; and writes into a temporary file ($TMP_PATH/posted.$$) in the following format

e.g.

```
COUNTY = beaver
MAP_TYPE = landuse
```

Then in the Bourne shell CGI program, the COUNTY value was read by the following piece of code:

```
COUNTY=`grep "COUNTY =" $TMP_PATH/posted.$$ | sed 's/COUNTY = //'`
```

4.2.2 Interface with ORACLE

Two environment variables are needed to enable a Pro*C program to access an ORACLE database. They are ORACLE_HOME and ORACLE_SID. ORACLE_HOME specifies the home directory of the ORACLE package. ORACLE_SID specifies the ORACLE System ID (or instance ID). In order to run a Pro*C program from a Bourne Shell script, we need to add the following lines before calling the Pro*C program .

```
ORACLE_HOME=/home/oracle
ORACLE_SID=sid1
export ORACLE_HOME
export ORACLE_SID

# Run the Pro*C program
# ......
```

4.2.3 ORACLE database management

Database management is always a big issue in a database system. The following are descriptions of the methods used in this project to manage the ORACLE database for the simulation data.

- Create a separate tablespace for the project.

```
CREATE TABLESPACE MAPS
DATAFILE '/home/oracle/orahome/dbs/mapsid1.dbf' SIZE 150M ONLINE;
```

- Special user accounts for management and user

MAPMGR account is responsible for the database management of the project, such as maintenance of TABLEs, VIEWs, PROCEDUREs, and loading data.

```
CREATE USER MAPMGR IDENTIFIED BY xxxxxxxx
DEFAULT TABLESPACE "MAPS" TEMPORARY TABLESPACE "MAPS"
PROFILE "DEFAULT";

GRANT CONNECT, RESOURCE TO MAPMGR WITH ADMIN OPTION;

/* give MAPMGR the power to create and drop user */
GRANT CREATE USER TO MAPMGR;
GRANT DROP USER TO MAPMGR;
```

MAPUSR account is needed for Pro*C programs to login to ORACLE database and query the tables for calculation.

```
/* Create the MAPUSR by MAPMGR */
CREATE USER MAPUSR IDENTIFIED BY xxxxxxxx
DEFAULT TABLESPACE "MAPS" TEMPORARY TABLESPACE "MAPS"
PROFILE "DEFAULT";

/* Allow MAPUSR to connect to the Oracle server */
GRANT CONNECT TO MAPUSR:

/* Only allow mapusr to select on tables */
GRANT SELECT ON CHEMOD TO MAPUSR;
GRANT SELECT ON COUNTY TO MAPUSR;
GRANT SELECT ON CROPS  TO MAPUSR;
GRANT SELECT ON TRAVEL TO MAPUSR;
GRANT SELECT ON GRASSID TO MAPUSR;
GRANT SELECT ON SOILNAME TO MAPUSR;
GRANT SELECT ON UNIT TO MAPUSR;
GRANT SELECT ON TRADCOMM TO MAPUSR;
GRANT SELECT ON CHEMICAL TO MPAUSR;
GRANT SELECT ON TRAVELID TO MAPUSR;

/*Allow MAPUSR to execute some stored procedures & functions */
GRANT EXECUTE ON GETCOUNTYINDEX TO MAPUSR;
GRANT EXECUTE ON GETCONVERSION TO MAPUSR;
GRANT EXECUTE ON GETCOUNTYNAME TO MAPUSR;
GRANT EXECUTE ON GETSTDUNIT TO MAPUSR;
GRANT EXECUTE ON PGETMUID TO MAPUSR;
```

- Stored procedures and functions to improve the performance

  Several stored procedures and functions are implemented and stored in the ORACLE RDBMS Because stored procedures and functions are already compiled, they do not need to be compiled again when a user calls them from a Pro*C program. This can make the database query and operation faster than submitting PL/SQL block directly from the Pro*C program. Another advantage of stored procedures or functions is that the Pro*C source programs do not need to be modified or recompiled if we make some modifications to the procedures or functions, because the Pro*C programs just send requests for running the procedures or functions and do not care about what they really do.

The following table shows the stored procedures and functions developed in this project :

Procedure/Function Name	Description
GetCountyIndex - Function	Get CountyIndex from COUNTY table, according to CountyName
GetConversion - Function	Get conversion from UNIT table, according to the unit given.
GetStdUnit - Function	Get the stdunit from UNIT table, according to the unit given
pGetMuid - Procedure	Given CountyIndex, SoilIndex, returns the Muid, and Muname.

Table 4-7 Stored procedures and functions

4.2.4 Interface with GRASS

To run GRASS commands from a Bourne Shell script, several environment variables must be set up. The following is the piece of code that creates these environment variables for GRASS:

```
GISBASE=/home/grass4.1
export GISBASE

GISDBASE=/usr/DISK3/grass4.1.data
export GISDBASE

LOCATION_NAME=albers
export LOCATION_NAME
GISRC=$gisrc
export GISRC

GIS_LOCK=$$
export GIS_LOCK

LOCATION=$location
export LOCATION

MAPSET=$mapset
export MAPSET

PAINTER=ppm
export PAINTER

MAPLP=$maplp
export MAPLP

PATH=/home/grass4.1/bin:/home/grass4.1/scripts:/home/grass4.1/gard
en/bin:/home/grass4.1/tools:/home/grass4.1/etc:$PATH
export PATH

PROJ_LIB=/home/grass4.1/etc
export PROJ_LIB

# set up env
g.gisenv GISDBASE=$GISDBASE
g.gisenv LOCATION_NAME=$LOCATION_NAME
g.gisenv LOCATION=$LOCATION
g.gisenv MAPSET=$MAPSET
g.gisenv PAINTER=ppm
g.gisenv MAPLP=$MAPLP

eval `g.gisenv`

#-----------------------------
# Followed by GRASS command
#-----------------------------
# ...
```

4.2.5 Handling multi-users in GRASS

This is a very important issue, because there can be multiple WWW users running the system at the same time. Although GRASS is a multi-user GIS database system, each user must occupy a mapset completely and other users can not use it at the same time. This problem was solved by setting up several mapsets in GRASS solely for the WWW users. In the implementation of MAPS for WWW, there are 5 mapsets (wwwusr1, wwwusr2, wwwusr3, wwwusr4, wwwusr5) setup for WWW users A dummy file named UNLOCK is also created in each mapset directory to show it is a free mapset, otherwise the file name should be LOCK to show it is being used by someone. The following Bourne Shell code is used to determine if there are any free mapsets available.

```
#-------------------------------
# check if we have free mapset
#-------------------------------
locnumber=1
location=$GISDBASE/$LOCATION_NAME/wwwusr
while test $locnumber -lt 10
do
        if mv -f $location$locnumber/UNLOCK $location$locnumber/LOCK
        then
            touch $location$locnumber/LOCK
            mapset=wwwusr$locnumber
            location=$GISDBASE/$LOCATION_NAME/$mapset
            gisrc=$TMP_PATH/.grassrc$locnumber
            maplp=$TMP_PATH/wwwusr$locnumber.ppm
            break
        else
            locnumber=`expr $locnumber + 1`

            if [ $locnumber = 6 ]
            then
                sleep 5
                locnumber=1
            fi
        fi
done
```

4.2.6 Map operations in GRASS.

- Control the map size in GRASS.

  The simplest way to control map size is to set up 2 environment variables which define the width and the height.

  e.g.

```
WIDTH=400
HEIGHT=400
export WIDTH
export HEIGHT
```

  Then, when GRASS generates a map, the map size will be 400 x 400. And the smaller the map size is, the faster GRASS can create it. E.g. in a test to generate the soils map for CADDO county, it takes 6 seconds for 200 x 200 map size, 10 seconds for 400 x 400 map size, and 25 seconds for 800 x 800 map size.

- Reclass and change map colors in GRASS

  In MAPS for WWW, the maps stored in GRASS are reclassed according to the calculated results generated by the Pro*C programs. The Pro*C programs create a reclass rule file. The reclass rule file contains the pairs of soil ID and category value.

  e.g.

```
1 = 5
3 = 1
4 = 2
...
```

  The CGI program uses the rule file and r.reclass command to reclass the map in GRASS. It then uses the r.colors command to define the color of each map category

e.g.

```
#-- ... Pro*C program generates the $rulefile ... ---
#-- The rulefile is a list of (soil --> category) map

#-- reclass the soils map --
r.reclass input=$MAPS_BASE out=reclass1 < $rulesfile

#-- change the color of the reclassed map --
cat $PROG_PATH/rules.color | r.colors map=reclass1 color=rules >
    /dev/null 2>&1
```

4.2.7 Convert PPM file to GIF format

In the project, we used GRASS to generate maps in PPM (Portable PixMap) format. Then we used a program called "ppmtogif " to convert the PPM file to GIF format. The "ppmtogif " can be downloaded from the World Wide Web

e.g.

```
PPMFILE=maps.ppm
GIFFILE=maps.gif

ppmtogif < $PPMFILE > $GIFFILE
```

4.2.8 Create legend picture file

Legend pictures are designed to show users the meaning of colors associated with the maps. Several C programs were developed to create the legend pictures (in GIF format). They use a simple GIF library to draw a picture in memory and then output it to a file. The C source code for generating a legend GIF file for Travel Time maps is given in Appendix D

4.2.9 Control the temporary files

MAPS for WWW generates many temporary files during its execution. To prevent these temporary files from using too much disk space, we should delete them when they are no longer needed. A program was developed to delete the temporary files more than 60 minutes old.

4.2.10 Security issues

The security issues of the INTERNET or WWW are beyond the scope of this thesis. But WWW application developers should be very careful when they write the CGI programs. There are several rules for writing UNIX Shell CGI programs:

1. Always put CGI programs in the "cgi-bin" directory which is only accessible by WWW developers and system administrators.
2. Be careful when using the wildcard characters (e.g. *, ?, $$, $* etc.) in UNIX Shell CGI programs
3. Always check the user's inputs. Because the users or hackers may input things unexpected, which may either make your program die or leak important information of your system.
4. Do not use the user's input as a command in a CGI program. Otherwise, the outside users are given the privilege to operate on your machine, which can be dangerous to your system.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

MAPS for WWW makes it possible for users to access pesticide transport data and get useful decision support information from anywhere at anytime through the Internet. The only requirement for the user is an HTML 3.0 compatible WWW browser running on any platform with an Internet connection. Users are free of the headaches of worrying about anything like system installation, management, and enhancement. ORACLE RDBMS on the server provides powerful data storage, data query, and calculation capabilities to manage data. GRASS provides full featured GIS data storage and operations that are efficient and convenient to use. MAPS for WWW utilizes the power of both ORACLE and GRASS to handle data storage and operations, and integrates them with a WWW server to run on the UNIX operating system.

To implement the system, the knowledge of UNIX, Shell, C, ORACLE, GRASS, SQL, WWW, HTML was needed as well as the ability to integrate these heterogeneous systems into one efficiently working system. This system extends the application of World Wide Web The work also provides valuable experience to other WWW developers MAPS for WWW can be reached at http://www.agr.okstate.edu/MAPS.

Many programs have been developed for MAPS for WWW. The following table shows some statistical information of these programs. Some of the HTML codes are output directly from the CGI programs, and are not included in the following table.

Code	Lines
C and Pro*C	4581
Bourne Shell	1144
SQL and PL/SQL	326
Import Control Files	105
HTML Files	416
Make File	331
Total	6903

Table 5-1 Statistics of the programs

5.2 Future work

1. More features and functionality can be added to the system in the future. For an example, the capability for users to input Koc, Half-life, and HALEQ values in Screen 1.2 has been discussed and is listed for future development.

2. The information provided by MAPS for WWW is for public use. WWW users are not restricted from accessing the system in the future. However, some Internet hackers like to destroy other's systems. Although the security issue is not the topic of the thesis, it is good for the system administrator to setup a firewall to protect their local area network.

3. It will be interesting to explore the possibility and feasibility of using Java to implement MAPS for WWW. This will require the users to have a Java enabled WWW browser. Also the developers must have a Java development environment.

BIBLIOGRAPHY

1. Abbey, Michael. Oracle, a beginner's guide, Osborne McGraw-Hill, 1995

2. Arthur, Lowell Jay. UNIX Shell Programming, John Wiley & Sons, Inc., 1994

3. Ault, Michael R. Oracle 7.0 administration & management, John Wiley & Sons, Inc. 1994

4. Bobrowski, S. M. Mastering Oracle7 & Client/Server Computing, SYBEX Inc. 1994

5. Cornell, Gary Core Java, SunSoft Press, 1996

6. December, John. The World Wide Web unleashed, SAMS Publishing, 1994

7. Eckel, George, Building a UNIX Internet Server, NewRiders Publishing, 1995

8. Feuerstein, Steven. Oracle PL/SQL Programming, O'Reilly & Associates, 1995

9. GRASS 4.0 Programmer's Manual, U. S. Army Engineers Construction Engineering Research Laboratory (Unpub.), August, 1992

10. GRASS 4.0 User's Manual, U. S. Army Engineers Construction Engineering Research Laboratory (Unpub.), July, 1991

11. Greene, Joseph B. Oracle DBA Survival Guide, SAMS Publishing, 1995

12. Harlow, England. HTML 3 : electronic publishing on the World Wide Web, New York : Addison-Wesley, 1996

13. Hatfield, Thomas H. Environmental health database on the World Widw Web, Journal of Environmental Health (ISSN:0022-0892) v 57 p 30-2 June '95

14. Kernighan B W and Mashey J. R. The UNIX Programming Environment, IEEE Computer, April 1981, pp. 12-24

15. Koch, G., Muller, R. and Loney, K. Oracle: The complete Reference, 3/e, McGraw-Hill, Inc. 1995

16. Kwan, Thomas T., McGrath, Robert E. and Reed, Daniel A. NCSA's World Wide Web server: design and performance, Computer (ISSN:0018-9162) v 28 p 68-74 November '95

17. Liu, C. , Peek, J and Jones, R. Managing INTERNET Information Services, O'Reily & Associates, Inc. 1994

18. Loney, Kevin. Oracle DBA handbook, Osborne McGraw-Hill, 1994

19. Ma, Fengxia. Using the UNIX shell to integrate a management model with GIS, MS Thesis, Oklahoma State University, Computer Science Department, Stillwater, Oklahoma, 1993

20. McClanahan, David. Oracle Developer's Guide, 1996

21. Nofziger, D L. and Hornsby, A. G. A microcomputer-based management tool for chemical movement in soil, Applied Agric, 1:50-56, 1986

22. Nofziger, D. L. and Hornsby, A. G. Chemical Movement in Layered Soils: User's Manual, Circular 780, Florida Cooperation Extension Service, Institute of Food and Gainesville. FL. 1992, pp 44.

23 Nofziger, D. L., Chen, J. S. and Haan, C. T Evaluating the Chemical Movement in Layered Soil Model as a Tool for Assessing Risk of Pesticide Leaching to Groundwater, Journal of Environmental Science And Health, p1133-p1155, 1994

24. Pennell, K. D. , Hornsby, A. G., Jessup, R. E., and Rao, P. S. C. Evaluation of five simulation models for predicting aldicard and bromide behavior under field conditions, Water Resources Res., 26:2679-2693, 1990

25 Sturner, Gunter Oracle 7 . a user's and developer's guide, including version 7.1, International Thomson Publishing, 1994

26. Sobell, M. G. UNIX SYSTEM V : A Practical Guide, Third Edition, The Benjamin/Cummings PublishingCompany, Inc. 1995

27. Vetter, Ronald J., Spell, Chris, and Ward, Charles. Mosaic and the World-Wide Web, Computer (ISSN:0018-9162) v 27 p49-57 October '94

APPENDICES

APPENDIX A

A BOURNE SHELL CGI PROGRAM

```
#!/bin/sh
#---------------------------------
# Author : Tao Zhu
# Email  : ztao@a.cs.okstate.edu
# Project: MAPS For WWW
#---------------------------------
trap '/bin/mv -f $LOCATION/LOCK $LOCATION/UNLOCK' 0 1 2 3 5 9 15

#================================
# Setup several important env var
#================================
PROG_PATH=/usr/local/etc/httpd_1.3/cgi-bin/MAPS
TMP_PATH=/usr/DISK5/htdocs/MAPS/tmp
DOC_PATH=/usr/DISK5/htdocs/MAPS
HTTP_PATH=/usr/local/etc/httpd_1.3/cgi-bin/MAPS
GISBASE=/usr/DISK3/grass4.1.data
PBM_PATH=$PROG_PATH
GIF_PATH=/usr/DISK5/htdocs/MAPS/gif
export PROG_PATH
export HTTP_PATH
export TMP_PATH
export GISBASE
export PBM_PATH
export GIF_PATH
export DOC_PATH

#-- HTML header --
echo 'Content-type: text/html'
echo
echo '<HTML>'
echo '<HEAD>'
echo '<TITLE>MAPS for WWW -- Output</TITLE>'
echo '</HEAD>'
echo
echo '<BODY BGCOLOR="#C0C0C0">'

#================================
# create parameter file
#================================
$PROG_PATH/postquery-MAPS $* > $TMP_PATH/posted.$$
parmfile=$TMP_PATH/posted.$$
#cat $parmfile

#================================
# Get all the user data
#================================
MAPS_TYPE=`grep "MAPS_TYPE =" $parmfile | sed 's/MAPS_TYPE = //'`
MAPS_COUNTY=`grep "MAPS_COUNTY =" $parmfile | sed 's/MAPS_COUNTY = //'`
MAPS_CROP=`grep "MAPS_CROP =" $parmfile | sed 's/MAPS_CROP = //'`
MAPS_APPLDATE=`grep "MAPS_APPLDATE =" $parmfile | sed 's/MAPS_APPLDATE = //'`
MAPS_IRRTYPE=`grep "MAPS_IRRTYPE =" $parmfile | sed 's/MAPS_IRRTYPE = //'`
MAPS_PROB=`grep "MAPS_PROB =" $parmfile | sed 's/MAPS_PROB = //'`
MAPS_TRADENAME=`grep "MAPS_TRADENAME =" $parmfile | sed 's/MAPS_TRADENAME = //'`
MAPS_PRODRATE=`grep "MAPS_PRODRATE =" $parmfile | sed 's/MAPS_PRODRATE = //'`
MAPS_APPLUNIT=`grep "MAPS_APPLUNIT =" $parmfile | sed 's/MAPS_APPLUNIT = //'`
MAPS_DEPTHMIX=`grep "MAPS_DEPTHMIX =" $parmfile | sed 's/MAPS_DEPTHMIX = //'`
MAPS_POROSITY=`grep "MAPS_POROSITY =" $parmfile | sed 's/MAPS_POROSITY = //'`
MAPS_OUTMAP=`grep "OUTPUT_MAP =" $parmfile | sed 's/OUTPUT_MAP = //'`
MAPS_OUTLIST=`grep "OUTPUT_LIST =" $parmfile | sed 's/OUTPUT_LIST = //'`
MAPS_AREAUNIT=`grep "MAPS_AREAUNIT =" $parmfile | sed 's/MAPS_AREAUNIT = //'`
MAPS_PID=$$
#---------------
export MAPS_TYPE
export MAPS_COUNTY
export MAPS_CROP
export MAPS_APPLDATE
export MAPS_IRRTYPE
export MAPS_PROB
export MAPS_PID
export MAPS_TRADENAME
export MAPS_PRODRATE
export MAPS_APPLUNIT
export MAPS_DEPTHMIX
export MAPS_POROSITY
export MAPS_AREAUNIT
```

```
#echo MAPS_COUNTY = $MAPS_COUNTY
#echo MAPS_CROP = $MAPS_CROP
#echo MAPS_APPLDATE = $MAPS_APPLDATE
#echo MAPS_IRRTYPE = $MAPS_IRRTYPE
#echo MAPS_PROB = $MAPS_PROB
#echo MAPS_TRADENAME = $MAPS_TRADENAME
#echo MAPS_PRODRATE = $MAPS_PRODRATE
#echo MAPS_APPLUNIT = $MAPS_APPLUNIT

#------------------------ Check user input ---------------------------
#-- For Travel Time, we don't need to check anything                  --
#-- For AmtLeached , we need to check ProdRate                        --
#-- For GWH and POE, we also need to check DepthMix, Porosity Aquifer --
#---------------------------------------------------------------------
echo '<CENTER><H2>Maps For WWW -- Output</H2></CENTER><P>'
case "$MAPS_TYPE" in
      T) echo '<CENTER>'
         echo '<H3> Travel Time (Probability = '$MAPS_PROB'%)</H3>'
         echo '</CENTER>'
         ;;
      A) echo '<CENTER>'
         echo '<H3>Amount Leached (Probability = '$MAPS_PROB'%)</H3>'
         echo '</CENTER>'
         $PROG_PATH/check
         ;;
      G) echo '<CENTER>'
         echo '<H3>Groundwater Hazard (Probability = '$MAPS_PROB'%)</H3>'
         echo '</CENTER>'
         $PROG_PATH/check_GWH
         ;;
      *) echo '<CENTER>'
         echo '<H3>Probability of Exceeding HAL or MCL</H3>'
         echo '</CENTER>'
         $PROG_PATH/check_POE
         ;;
esac
#-- check the return value --
#echo '<PRE>'
if [ $? != 0 ]
then
    echo '</PRE>'
    echo '</BODY>'
    echo '</HTML>'
    exit -1
fi

#-- Check if MAP and LIST are both unchecked --
#echo MAP -- $MAPS_OUTMAP
#echo LIST -- $MAPS_OUTLIST
if [ "$MAPS_OUTMAP" != "M" ]
then
    if [ "$MAPS_OUTLIST" != "L" ]
    then
        echo '<PRE><CENTER>'
        echo 'You must select at least 1 of the Output field.'
        echo '</CENTER></PRE>'
        echo '</BODY></HTML>'
        exit -1
    fi
fi

#-------------------------------------------------------------
#-- Following 2 environment variables are needed for Oracle --
#-------------------------------------------------------------
ORACLE_HOME=/home/oracle
ORACLE_SID=sid1
export ORACLE_HOME
export ORACLE_SID

#-----------------------------------------------------
#-- Run different Pro*C program for different map type --
#-----------------------------------------------------
case "$MAPS_TYPE" in
    T) $PROG_PATH/travel
```

```
        mapsnum=$?
        TYPE=0
        ;;
    A) $PROG_PATH/amtleach
        mapsnum=$?
        TYPE=0
        ;;
    G) $PROG_PATH/gwhazard
        mapsnum=$?
        TYPE=1
        ;;
    *) $PROG_PATH/poe
        mapsnum=$?
        TYPE=1
        ;;
esac

#-- Check if there is anything wrong during the calculation --
if [ $mapsnum = 255 ]
then
    echo 'Something wrong!'
    echo '</PRE>'
    echo '</BODY> </HTML>'
    exit -1
fi

#-- For Groundwater Hazard and POE there should be only 1 map --
if [ $TYPE = 1 ]
then
    mapsnum=1
fi

echo '</PRE>'

#-------------------------------------------------------
#-- Setup the environment variables needed for GRASS --
#-------------------------------------------------------
if [ "$MAPS_OUTMAP" = "M" ]
then

GISBASE=/home/grass4.1
export GISBASE

GISDBASE=/usr/DISK3/grass4.1.data
export GISDBASE

LOCATION_NAME=albers
export LOCATION_NAME

#---------------------------------
#-- check if we have free mapset --
#---------------------------------
locnumber=1
location=$GISDBASE/$LOCATION_NAME/wwwusr
while test $locnumber -lt 10
do
    if mv -f $location$locnumber/UNLOCK $location$locnumber/LOCK
    then
        touch $location$locnumber/LOCK
        mapset=wwwusr$locnumber
        location=$GISDBASE/$LOCATION_NAME/$mapset
        gisrc=$TMP_PATH/.grassrc$locnumber
        maplp=$TMP_PATH/wwwusr$locnumber.ppm
        break
    else
        locnumber= expr $locnumber + 1

        if [ $locnumber = 6 ]
        then
            sleep 5
            locnumber=1
        fi
    fi
done
```

```
GISRC=$gisrc
export GISRC

GIS_LOCK=$$
export GIS_LOCK

LOCATION=$location
export LOCATION

MAPSET=$mapset
export MAPSET

PAINTER=ppm
export PAINTER

MAPLP=$maplp
export MAPLP

PATH=/home/grass4.1/bin:/home/grass4.1/scripts:/home/grass4.1/garden/bin:/home/grass4.1/to
ols:/home/grass4.1/etc:$PATH
export PATH

PROJ_LIB=/home/grass4.1/etc
export PROJ_LIB

g.gisenv GISDBASE=$GISDBASE
g.gisenv LOCATION_NAME=$LOCATION_NAME
g.gisenv LOCATION=$LOCATION
g.gisenv MAPSET=$MAPSET
g.gisenv PAINTER=ppm
g.gisenv MAPLP=$MAPLP

eval `g.gisenv`

#-------------------
#-- Select region --
#-------------------
county=`grep "$MAPS_COUNTY =" $PROG_PATH/county.lst`
COUNTY=`$PROG_PATH/getcounty $county`
#echo COUNTY = $COUNTY
MAPS_BASE=$COUNTY.soils
VECT=$COUNTY.boundary
#echo $MAPS_BASE
g.region rast=$MAPS_BASE

#-------------------------
#-- set MASK to overlay --
#-------------------------
AREA="Entire Area"
OVERLAY=`grep "DISPLAY_FOR =" $parmfile | sed 's/DISPLAY_FOR = //'`
#echo OVERLAY = $OVERLAY
if [ "$OVERLAY" != "E" ]
then
    if [ "$OVERLAY" = "C" ]
    then
        maskfile=$COUNTY.crop
        AREA="Cultivated Area"
    else
        maskfile=$COUNTY.irrigated
        AREA="Irrigated Area"
    fi

#-- copy it to MASK --
    g.copy rast=$maskfile,MASK > /dev/null 2>&1
fi

#-- For MAPS_OUTMAP == M --
fi
#-------------------------


#==============================
#-- Loop to show all the maps --
#==============================
loopnum=0
```

```
while [ $loopnum -lt $mapsnum ]
do
    loopnum=`expr $loopnum + 1`
    #echo loopnum = $loopnum

if [ "$MAPS_OUTMAP" = "M" ]
then
    rulesfile=$TMP_PATH/rules.$MAPS_PID.$loopnum
    maxminfile=$TMP_PATH/maxmin.$MAPS_PID.$loopnum

#-- reclass the soils map --
r.reclass input=$MAPS_BASE out=reclass1 < $rulesfile

#-- change the color of the reclassed map --
cat $PROG_PATH/rules.color | r.colors map=reclass1 color=rules > /dev/null 2>&1

#-------------------------------------------------
#-- Generate ppm file, and convert it into gif file  --
#-------------------------------------------------
GIF=maps.$$.$loopnum.gif
GRASTER=reclass1
GVECTOR=$VECT
GSITES=none
VCOLOR=none
SCOLOR=none
GIF_SIZE=`grep "GIF_SIZE =" $TMP_PATH/posted.$$ | sed 's/GIF_SIZE = //'`
$PROG_PATH/rast2gif $GIF $GRASTER $GVECTOR $GSITES $VCOLOR $SCOLOR $GIF_SIZE

#-- remove the reclass map from mapset --
g.remove rast=reclass1 > /dev/null 2>&1

#---------------------
#-- Remove the MASK --
#---------------------
if [ "$OVERLAY" != "E" ]
then
    g.remove rast=MASK > /dev/null  2>&1
fi

#-----------------------------
# Generate legend gif file  --
#-----------------------------
TTGIF=TT-$MAPS_PID.$loopnum.gif
ttgif=$GIF_PATH/$TTGIF
case "$MAPS_TYPE" in
T)
    maxval=`grep "MAX =" $maxminfile | sed 's/MAX = //'`
    minval=`grep "MIN =" $maxminfile | sed 's/MIN = //'`
    $PROG_PATH/tt-gif $ttgif $minval $maxval
    TTGIF=gif/$TTGIF
    ;;
A)
    maxval=`grep "MAX =" $maxminfile | sed 's/MAX = //'`
    minval=`grep "MIN =" $maxminfile | sed 's/MIN = //'`
    $PROG_PATH/a1-gif $ttgif $minval $maxval
    TTGIF=gif/$TTGIF
    ;;
G)  TTGIF=gwh.gif
    ;;
*)  TTGIF=poe.gif
    ;;
esac

#-------------------
#-- Show the MAP  --
#-------------------
echo '<CENTER>'
echo '<TABLE BORDER=3>'

echo '<!-- table header -->'
echo '<tr valign=middle>'
echo '    <th align=middle> <B>' $MAPS_COUNTY -- $AREA' </B> </th>'
echo '</tr>'

headerfile=$TMP_PATH/head.$MAPS_PID.$loopnum
```

```
echo '<!-- User input info -->'
echo '<tr valign=middle>'
echo '    <td align=left><pre>'
cat $headerfile
echo '</pre></td>'

echo '<tr><td align=middle>'
echo '        <IMG SRC="http://clay.agr.okstate.edu/MAPS/gif/'$GIF'"> <BR>'
echo '    </td>'
echo '</tr>'

echo '<tr><td align=middle>'
echo '        <IMG SRC="http://clay.agr.okstate.edu/MAPS/'$TTGIF'"> <BR>'
echo '    </td>'
echo '</TABLE>'
echo '</CENTER>'

echo '<BR>'

#-- -- -- -- -- --
fi
#-- -- -- -- -- --

#-------------------------------------
# For GWH and POE, show detail info
#-------------------------------------
detailfile=$TMP_PATH/detail.$MAPS_PID
if [ "$MAPS_TYPE" = "G" ]
then
    echo '<CENTER><PRE>'
    cat $detailfile
    echo '</PRE></CENTER>'
fi
if [ "$MAPS_TYPE" = "P" ]
then
    echo '<CENTER><PRE>'
    cat $detailfile
    echo '</PRE></CENTER>'
fi


#---------------------------
#-- Show List Information --
#---------------------------
if [ "$MAPS_OUTLIST" = "L" ]
then
echo '<CENTER>'
echo '<TABLE BORDER=0 WIDTH=50%>'
echo '<tr valign=middle>'
echo '    <td align=left>'
prob= expr 100 - $MAPS_PROB
case "$MAPS_TYPE" in
T)
ingfile=$TMP_PATH/ingredient.$MAPS_PID.$loopnum
MAPS_ING= grep "INGREDIENT =" $ingfile | sed 's/INGREDIENT = //'

echo 'Travel time for <U><B>' $MAPS_ING '</B></U> to reach a depth of 1 meter'
echo 'for different soils in <U><B>'$MAPS_COUNTY'</B></U> county. '
echo 'Computing the travel time for applications in many different years indicates'
echo 'that the times shown below were exceeded for <U><B>'$MAPS_PROB'%</B></U> of the
applications;'
echo 'travel times for <U><B>'$prob'%</B></U> of the applications were less than the
values'
echo ' shown.'
echo '<BR><BR>'
;;
A)
ingfile=$TMP_PATH/ingredient.$MAPS_PID.$loopnum
MAPS_ING= grep "INGREDIENT =" $ingfile | sed 's/INGREDIENT = //'

echo 'Amount of <U><B>' $MAPS_ING '</B></U> leached beyond of 1 meter '
echo 'for different soils in <U><B>' $MAPS_COUNTY '</B></U> county.'
echo 'Computing the amount leached for applications in many different years indicates'
echo 'that the amounts shown below were exceeded for <U><B>'$MAPS_PROB'%</B></U> of the '
echo 'applications; amounts for <U><B>'$prob'%</B></U> of the applications were less than'
```

```
echo 'the values shown.'
echo '<BR><BR>'
;;
G)
echo 'Groundwater hazard for <U><B>' $MAPS_TRADENAME '</B></U> for different soils'
echo 'in <U><B>' $MAPS_COUNTY '</B></U> county. </B>'
echo 'Computing the groundwater hazard for many different applications '
echo 'indicates that the hazards shown below were exceeded for <U><B>'$MAPS_PROB'%</B></U>
of'
echo 'the applications; groundwater hazards for <U><B>'$prob'%</B></U> of the applications
'
echo 'were less than the values shown.'
echo '<BR><BR>'
;;
*)
echo 'Probability of <U><B>' $MAPS_TRADENAME '</B></U> exceeding the health '
echo 'advisory level or maximum contaminant level for different soils '
echo 'in <U><B>' $MAPS_COUNTY '</B></U> county. '
echo 'Values shown below represent the fraction of the applications '
echo 'which result in pesticide concentrations greater than the health '
echo 'advisory level or the maximum contaminant level of the pesticide.'
echo '<BR><BR>'
;;
esac
echo '</td>'

#-- show the soil list --
listfile=$TMP_PATH/list.$MAPS_PID.$loopnum
cat $listfile

echo '</TABLE>'
fi

#-- seperate between multiple maps --
echo '<HR>'

done
#-------- end of loop --------

#------------------------
# Free the mapset in use
#------------------------
#/bin/mv -f $LOCATION/LOCK $LOCATION/UNLOCK

#-------------
echo '</BODY>'
echo '</HTML>'

#=========
#== End ==
#=========
```

APPENDIX B

A PRO*C PROGRAM

```
/*************************************************************/
/* File Name : travel.pc                                     */
/* Author    : Tao Zhu                                       */
/* Created   : 07-05-1996                                    */
/* Description:                                              */
/*   Calculate the travel time for Maps for WWW.             */
/*************************************************************/

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

/*-- Define constants for VARCHAR lengths. --*/
#define     UNAME_LEN       20
#define     PWD_LEN         40
#define     ARRAY_LENGTH    10
#define     MAX_REC         100

/*-- Declare variables.  No declare section is needed if MODE=ORACLE. --*/
char *username = "MAPUSR";
char *password = "xxxxxx";

/*-- Define a host structure for the output values of a SELECT statement --*/
struct
{
    VARCHAR    s_CommonName[31];
    VARCHAR    s_ProdUnit[7];
    double     s_AIlb_ProdUnit;
    double     s_Koc;
    double     s_HalfLife;
    double     s_HALEQ;
} *g_TradeCommRec, gTradeCommRec;

struct
{
    short     s_CountyIndex;
    short     s_SoilIndex;
    VARCHAR   s_CropName[8];
    VARCHAR   s_ApplDate[9];
    VARCHAR   s_IrrigationType[8];
    short     s_Probability;
    short     s_Koc0;
    short     s_Koc1;
    short     s_Koc2;
    short     s_Koc4;
    short     s_Koc7;
    short     s_Koc10;
    short     s_Koc20;
    short     s_Koc40;
    short     s_Koc70;
    short     s_Koc100;
    short     s_Koc200;
    short     s_Koc400;
    short     s_Koc700;
    short     s_Koc1000;
    short     s_Koc2000;
    short     s_Koc4000;
    short     s_Koc7000;
    short     s_Koc10000;
    short     s_GrassIndex;
} *g_TravelRec, gTravelRec;

/* Include the SQL Communications Area.
   You can use #include or EXEC SQL INCLUDE. */
/* #include <sqlca.h> */
EXEC SQL INCLUDE sqlca.h;

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

/*-------------------------------------*/
/* global variables for the test data */
/*-------------------------------------*/
VARCHAR    var_TradeName[47];
```

```
EXEC SQL BEGIN DECLARE SECTION;
short     var_CountyIndex;
short     var_SoilIndex;
VARCHAR   var_Muid[8];
VARCHAR   var_SoilName[91];
EXEC SQL END DECLARE SECTION;

VARCHAR   var_CountyName[12];
VARCHAR   var_CropName[8];
VARCHAR   var_ApplDate[9];
VARCHAR   var_IrrigationType[9];
short     var_Probability;

short     Koc, Koc1, Koc2, t1, t2;
double    tt, mintt, maxtt, dif;
int       i, j;
int       ingno = 0;
int       pid;

struct
{
    short  s_GrassIndex;
    double s_TravelTime;
} gGrassIndex[ MAX_REC ];


/**********************/
/* main() starts here */
/**********************/
main( argc, argv )
int argc;
char *argv[];
{
    char temp_char[50];
    char temp_string[50];
    char rulesfile_name[100];
    char listfile_name[100];
    char maxminfile_name[100];
    char header_name[100];
    char tmp_path[50];
    int  segment=0;
    int  s1,s2,s3,s4;
    double delta;

    FILE *fp_rules;
    FILE *fp_list;
    FILE *fp_maxmin;
    FILE *fp_header;

    g_TradeCommRec = &gTradeCommRec;

    /*-------------------------------------------*/
    /* Register sql_error() as the error handler. */
    /*-------------------------------------------*/
    EXEC SQL WHENEVER SQLERROR DO sql_error("ORACLE error--\n");


    /*-------------------------------------------------------------*/
    /* Connect to ORACLE.  Program will call sql_error()           */
    /* if an error occurs when connecting to the default database.*/
    /*-------------------------------------------------------------*/
    EXEC SQL CONNECT :username IDENTIFIED BY :password;


    /*-----------------------------*/
    /* setup parameters from env   */
    /*-----------------------------*/
    strcpy((char *)var_CountyName.arr, getenv("MAPS_COUNTY"));
    var_CountyName.len = strlen( (char *)var_CountyName.arr );

    strcpy((char *)var_CropName.arr, getenv("MAPS_CROP"));
    var_CropName.len = strlen((char *)var_CropName.arr);

    strcpy((char *)var_ApplDate.arr, getenv("MAPS_APPLDATE"));
    var_ApplDate.len = strlen((char *)var_ApplDate.arr);
```

```
strcpy((char *)var_IrrigationType.arr, getenv("MAPS_IRRTYPE"));
var_IrrigationType.len = strlen((char *)var_IrrigationType.arr);
var_IrrigationType.arr[8] = '\0';

strcpy((char *)var_TradeName.arr, getenv("MAPS_TRADENAME"));
var_TradeName.len = strlen((char *)var_TradeName.arr);

var_Probability =  atoi( getenv("MAPS_PROB") );

pid = atoi( getenv("MAPS_PID") );
strcpy( tmp_path, getenv("TMP_PATH") );

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


/*-------------------------------*/
/* Declare cursor for cur_Travel */
/*-------------------------------*/
EXEC SQL DECLARE cur_Travel CURSOR FOR
         SELECT *
           FROM MAPMGR.TravelID
          WHERE CountyIndex = :var_CountyIndex
            AND CropName = :var_CropName
            AND ApplDate = :var_ApplDate
            AND IrrigationType = :var_IrrigationType
            And Probability = :var_Probability;

/*-------------------------------*/
/* Declare cursor cur_TradeComm */
/*-------------------------------*/
EXEC SQL DECLARE cur_TradeComm CURSOR FOR
         SELECT CommonName, ProdUnit, AIlb_ProdUnit, Koc, HalfLife, HALEQ
           FROM MAPMGR.CHEMICAL
          WHERE TradeName = :var_TradeName
            AND Koc IS NOT NULL
            AND Koc <= 10000
            AND HalfLife IS NOT NULL
            AND HALEQ > 0
            AND AILB_ProdUnit IS NOT NULL
            AND ProdUnit IS NOT NULL;

/*---------------------------------*/
/*---------------------------------*/
/* START THE QUERY AND CALCULATION */
/*---------------------------------*/
/*---------------------------------*/
EXEC SQL OPEN cur_TradeComm;
EXEC SQL WHENEVER NOT FOUND DO break;


for(;;)
{
    EXEC SQL FETCH cur_TradeComm INTO :g_TradeCommRec;

    gTradeCommRec.s_CommonName.arr[gTradeCommRec.s_CommonName.len] = '\0';
    gTradeCommRec.s_ProdUnit.arr[gTradeCommRec.s_ProdUnit.len] = '\0';

    ingno++;

    EXEC SQL OPEN cur_Travel;
    EXEC SQL WHENEVER NOT FOUND DO break;

    /*-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -*/
    /*-- Print out the header description part for the map --*/
    /*-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -*/
    sprintf(header_name, "%s/head.%d.%d", tmp_path, pid, ingno );
    fp_header = fopen( header_name, "w+" );
```

```
fprintf(fp_header,
    "<B> COUNTY    : %-12s   IRRIGATION TYPE : %8s </B>\n",
    var_CountyName.arr,
    var_IrrigationType.arr);
fprintf(fp_header,
    "<B> APPL.DATE : %-14s            CROP : %8s </B>\n",
    var_ApplDate.arr,
    var_CropName.arr);
fprintf(fp_header,
    "<B> PESTICIDE : %-40s  </B>\n",
    var_TradeName.arr);
fprintf(fp_header,
    "<B> ACTIVE INGREDIENT: %-33s  </B>\n",
    gTradeCommRec.s_CommonName.arr);

sprintf(temp_char, "%.1f ml/gOC", gTradeCommRec.s_Koc);
fprintf(fp_header,
    "<B> Koc : %-16s                                    </B>\n",
    temp_char);
fclose( fp_header );

/*----------------------------------------*/
/* Output the active ingredients to a file */
/*----------------------------------------*/
sprintf(listfile_name, "%s/ingredient.%d.%d", tmp_path, pid, ingno );
fp_list = fopen( listfile_name, "w+" );
fprintf(fp_list, "INGREDIENT = %s\n", gTradeCommRec.s_CommonName.arr);
fclose( fp_list);

/*-------------------------------*/
/*-- print out the output list --*/
/*-------------------------------*/
i = 0;
sprintf(listfile_name, "%s/list.%d.%d", tmp_path, pid, ingno );
fp_list = fopen( listfile_name, "w+" );

/*-- print out the header --*/
fprintf(fp_list, "<tr><td align=middle>\n");
fprintf(fp_list, "<PRE>\n");

fprintf(fp_list,
    "<B>=====================================================</B>\n");
fprintf(fp_list,
    "<B>COUNTY    : %-12s   IRRIGATION TYPE : %8s</B>\n",
    var_CountyName.arr,
    var_IrrigationType.arr);
fprintf(fp_list,
    "<B>APPL.DATE : %-14s            CROP : %8s</B>\n",
    var_ApplDate.arr,
    var_CropName.arr);
fprintf(fp_list,
    "<B>PESTICIDE : %-40s </B>\n",
    var_TradeName.arr);
fprintf(fp_list,
    "<B>ACTIVE INGREDIENT: %-33s </B>\n",
    gTradeCommRec.s_CommonName.arr);
fprintf(fp_list,
    "<B>Koc : %-16s                                    </B>\n",
    temp_char);
fprintf(fp_list,
    "<B>-----------------------------------------------------</B>\n");
fprintf(fp_list,
    "<B>                                              Travel Time</B>\n");
fprintf(fp_list,
    "<B>  Muid      Soil Name                                (days)</B>\n");
fprintf(fp_list,
    "-----------------------------------------------------\n");

/*-- Loop for the current ingrediant --*/
for(;;)
{
    EXEC SQL FETCH cur_Travel INTO :gTravelRec;

    gTravelRec.s_CropName.arr[gTravelRec.s_CropName.len] = '\0';
    gTravelRec.s_ApplDate.arr[gTravelRec.s_ApplDate.len] = '\0';
```

```
gTravelRec.s_IrrigationType.arr[gTravelRec.s_IrrigationType.len] = '\0';

Koc = gTradeCommRec.s_Koc;

if ( Koc <= 1 )
{
    Koc1 = 0;    Koc2 = 1;
    t1 = gTravelRec.s_Koc0;
    t2 = gTravelRec.s_Koc1;
}

if ( Koc > 1 && Koc <= 2 )
{
    Koc1 = 1;    Koc2 = 2;
    t1 = gTravelRec.s_Koc1;
    t2 = gTravelRec.s_Koc2;
}

if ( Koc > 2 && Koc <= 4 )
{
    Koc1 = 2;    Koc2 = 4;
    t1 = gTravelRec.s_Koc2;
    t2 = gTravelRec.s_Koc4;
}

if ( Koc > 4 && Koc <= 7 )
{
    Koc1 = 4;    Koc2 = 7;
    t1 = gTravelRec.s_Koc4;
    t2 = gTravelRec.s_Koc7;
}

if ( Koc > 7 && Koc <= 10 )
{
    Koc1 = 7;    Koc2 = 10;
    t1 = gTravelRec.s_Koc7;
    t2 = gTravelRec.s_Koc10;
}

if ( Koc > 10 && Koc <= 20 )
{
    Koc1 = 10;    Koc2 = 20;
    t1 = gTravelRec.s_Koc10;
    t2 = gTravelRec.s_Koc20;
}

if ( Koc > 20 && Koc <= 40 )
{
    Koc1 = 20;    Koc2 = 40;
    t1 = gTravelRec.s_Koc20;
    t2 = gTravelRec.s_Koc40;
}

if ( Koc > 40 && Koc <= 70 )
{
    Koc1 = 40;    Koc2 = 70;
    t1 = gTravelRec.s_Koc40;
    t2 = gTravelRec.s_Koc70;
}

if ( Koc > 70 && Koc <= 100 )
{
    Koc1 = 70;    Koc2 = 100;
    t1 = gTravelRec.s_Koc70;
    t2 = gTravelRec.s_Koc100;
}

if ( Koc > 100 && Koc <= 200 )
{
    Koc1 = 100;    Koc2 = 200;
    t1 = gTravelRec.s_Koc100;
    t2 = gTravelRec.s_Koc200;
}

if ( Koc > 200 && Koc <= 400 )
```

```
{
    Koc1 = 200;     Koc2 = 400;
    t1 = gTravelRec.s_Koc200;
    t2 = gTravelRec.s_Koc400;
}

if ( Koc > 400 && Koc <= 700 )
{
    Koc1 = 400;     Koc2 = 700;
    t1 = gTravelRec.s_Koc400;
    t2 = gTravelRec.s_Koc700;
}

if ( Koc > 700 && Koc <= 1000 )
{
    Koc1 = 700;     Koc2 = 1000;
    t1 = gTravelRec.s_Koc700;
    t2 = gTravelRec.s_Koc1000;
}

if ( Koc > 1000 && Koc <= 2000 )
{
    Koc1 = 1000;     Koc2 = 2000;
    t1 = gTravelRec.s_Koc1000;
    t2 = gTravelRec.s_Koc2000;
}

if ( Koc > 2000 && Koc <= 4000 )
{
    Koc1 = 2000;     Koc2 = 4000;
    t1 = gTravelRec.s_Koc2000;
    t2 = gTravelRec.s_Koc4000;
}

if ( Koc > 4000 && Koc <= 7000 )
{
    Koc1 = 4000;     Koc2 = 7000;
    t1 = gTravelRec.s_Koc4000;
    t2 = gTravelRec.s_Koc7000;
}

if ( Koc > 7000 && Koc <= 10000 )
{
    Koc1 = 7000;     Koc2 = 10000;
    t1 = gTravelRec.s_Koc7000;
    t2 = gTravelRec.s_Koc10000;
}

/*---------------------------*/
/* calculate the travel time */
/*---------------------------*/
tt = (double)(Koc - Koc1) * (t2 - t1) / (Koc2 - Koc1) + t1;

/*---------------------------------*/
/* save the value into grass array */
/*---------------------------------*/
gGrassIndex[i].s_GrassIndex = gTravelRec.s_GrassIndex;
gGrassIndex[i].s_TravelTime = tt;

/*-----------------------*/
/* decide max/min value */
/*-----------------------*/
if ( i == 0 )
    maxtt = mintt = tt;

if ( tt > maxtt )
    maxtt = tt;

if ( tt < mintt )
    mintt = tt;

/*-------------*/
/* increment i */
/*-------------*/
i++;
```

```
    /* Get the Muid and Name for the soil */
    var_SoilIndex = gTravelRec.s_SoilIndex;
    EXEC SQL EXECUTE
        BEGIN
            MAPMGR.PGETMUID( :var_CountyIndex, :var_SoilIndex,
                             :var_Muid, :var_SoilName );
        END;
    END-EXEC;

    var_Muid.arr[ var_Muid.len ] = '\0';
    var_Muid.arr[7] = '\0';
    var_SoilName.arr[ var_SoilName.len ] = '\0';
    var_SoilName.arr[35] = '\0';

    /* print out for the current record */
    fprintf(fp_list,
            "%-7s  %-35s %7d\n",
            var_Muid.arr,
            var_SoilName.arr,
            (int)tt);

    if ( ++segment == 5 )
    {
        /* print an empty line */
        fprintf(fp_list,"\n");
        segment = 0;
    }
}

/*-- print out the last line --*/
fprintf(fp_list,
   "<B>============================================================</B>\n");
fprintf(fp_list, "</PRE>\n");
fprintf(fp_list, "</td>\n");

fclose( fp_list );

EXEC SQL CLOSE cur_Travel;


/*----------------------------*/
/* Write min/max information */
/*----------------------------*/
sprintf( maxminfile_name, "%s/maxmin.%d.%d", tmp_path, pid, ingno );
fp_maxmin = fopen( maxminfile_name, "w+" );

fprintf( fp_maxmin, "MAX = %d\n", (int)maxtt );
fprintf( fp_maxmin, "MIN = %d\n", (int)mintt );

fclose( fp_maxmin );


/*-----------------------------------*/
/*-----------------------------------*/
/* reclass soil by their travel time */
/*-----------------------------------*/
/*-----------------------------------*/
j = 0;
sprintf(rulesfile_name, "%s/rules.%d.%d", tmp_path, pid, ingno);
fp_rules = fopen( rulesfile_name, "w+" );

delta = (maxtt - mintt) / 5.0;
s1 = (int) (mintt+delta+0.5);
s2 = (int) (mintt+2*delta+0.5);
s3 = (int) (mintt+3*delta+0.5);
s4 = (int) (mintt+4*delta+0.5);
while( j < i )
{
     dif = gGrassIndex[j].s_TravelTime;
     if ( dif <= s1 )
     {
          fprintf(fp_rules, "%d = %d\n", gGrassIndex[j].s_GrassIndex, 1);
          j++;
```

```
                continue;
            };

            if ( dif <= s2 )
            {
                fprintf(fp_rules, "%d = %d\n", gGrassIndex[j].s_GrassIndex, 2);
                j++;
                continue;
            };

            if ( dif <= s3 )
            {
                fprintf(fp_rules, "%d = %d\n", gGrassIndex[j].s_GrassIndex, 3);
                j++;
                continue;
            };

            if ( dif <= s4 )
            {
                fprintf(fp_rules, "%d = %d\n", gGrassIndex[j].s_GrassIndex, 4);
                j++;
                continue;
            };

            /* else, it is in the 5th section */
            fprintf(fp_rules, "%d = %d\n", gGrassIndex[j].s_GrassIndex, 5);
            j++;
            continue;
        }

        /* category 99 is the border */
        fprintf(fp_rules, "99 = 99\n");

        fclose( fp_rules );

    }

    EXEC SQL CLOSE cur_TradeComm;


    /*--------------------------*/
    /* Disconnect from ORACLE. */
    /*--------------------------*/
    EXEC SQL COMMIT WORK RELEASE;


    /*--------------------------*/
    /* Return No of ingredients */
    /*--------------------------*/
    exit( ingno );
}


/*--------------------------------*/
/* sql_error()                    */
/* Display oracle error messages */
/*--------------------------------*/
void
sql_error(msg)
char *msg;
{
    char err_msg[128];
    int buf_len, msg_len;

    EXEC SQL WHENEVER SQLERROR CONTINUE;

    printf("\n%s\n", msg);
    buf_len = sizeof (err_msg);
    sqlglm(err_msg, &buf_len, &msg_len);
    printf("%.*s\n", msg_len, err_msg);

    EXEC SQL ROLLBACK RELEASE;
    exit(-1);
}
```

APPENDIX C

A STORED PROCEDURE

```
/*-------------------------------------------------*/
/* Procedure Name : pGetMuid                       */
/* Parameters :                                    */
/*     pCountyIndex IN INTEGER,                    */
/*     pSoilIndex   IN INTEGER,                    */
/*     pMuid       OUT SOILNAME.MUID%TYPE,         */
/*     pSoilName   OUT SOILNAME.MUNAME%TYPE        */
/* Description :                                   */
/*     Get MUID and MUNAME for a soil according to */
/*     CountyIndex, and SoilIndex given.           */
/*-------------------------------------------------*/
CREATE OR REPLACE
PROCEDURE pGetMuid ( pCountyIndex IN INTEGER,
                     pSoilIndex IN INTEGER,
                     pMuid     OUT SOILNAME.MUID%TYPE,
                     pSoilName OUT SOILNAME.MUNAME%TYPE)
IS
--
BEGIN
    SELECT Muid, Muname
      INTO pMuid , pSoilName
      FROM SoilName
     WHERE CountyIndex = pCountyIndex
       AND SoilIndex = pSoilIndex;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
      pMuid := 'N/A';
      pSoilName := 'N/A';
END ;
```

APPENDIX D

A C PRORGAM FOR GIF GENERATION

```
/*************************************************************/
/* Filename : tt-gif.c                                       */
/* Author   : Tao Zhu                                        */
/* Created  : Aug-07-1996                                    */
/* Compile  : cc tt-gif.c gd.o -o tt-gif                     */
/* description:                                              */
/*            Draw graphic legend for MAPS Web version       */
/* --------------------------------------------------------- */
/* Update History :                                          */
/*************************************************************/
#include <stdio.h>
#include <stdlib.h>

#include "gd.h"

/*---------------------*/
/* constant definition */
/*---------------------*/
#define LM              5     /* Left Margin */
#define TM              5     /* Top Margin */
#define SM              5     /* Space Margin */
#define GIF_HEIGHT      TM*2+16*2+5+16+5
#define GIF_WIDTH       LM*2+8*8+5*6*8+4*8

/*----------------------------*/
/* global variables for color */
/*----------------------------*/
int white;
int blue;
int red;
int cyan;
int yellow;
int green;
int black;
int brown;
int purple;
int magenta;

int colors[5];

int minval, maxval;

/*----------------*/
/* Draw rectangle */
/*----------------*/
void DrawRectangle( im, x1, y1, x2, y2, color )
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 );
}

/*-----------------------*/
/* Draw the bar          */
/*-----------------------*/
void DrawBar( im )
gdImagePtr im;
{
    int Y, X;

    Y = TM + 16 + 8 + SM;

    X = LM + 8*8;
    DrawRectangle( im, X, Y, X+6*8, Y+10, colors[0] );
    gdImageFillToBorder( im, X+2, Y+2, colors[0], colors[0] );

    X = X + 6*8;
    DrawRectangle( im, X, Y, X+6*8, Y+10, colors[1] );
```

```
    gdImageFillToBorder( im, X+2, Y+2, colors[1], colors[1] );

    X = X + 6*8;
    DrawRectangle( im, X, Y, X+6*8, Y+10, colors[2] );
    gdImageFillToBorder( im, X+2, Y+2, colors[2], colors[2] );

    X = X + 6*8;
    DrawRectangle( im, X, Y, X+6*8, Y+10, colors[3] );
    gdImageFillToBorder( im, X+2, Y+2, colors[3], colors[3] );

    X = X + 6*8;
    DrawRectangle( im, X, Y, X+6*8, Y+10, colors[4] );
    gdImageFillToBorder( im, X+2, Y+2, colors[4], colors[4] );

    /* Draw the title */
    gdImageString( im, LM, TM, "Travel", black );
    gdImageString( im, LM, TM+16+SM, " Time", black );
    gdImageString( im, LM, TM+2*(16+SM), "(days)", black);

}

/*-----------------------*/
/* Draw the range texts  */
/*-----------------------*/
void DrawRange( im )
gdImagePtr im;
{
    char t1[10], t2[10];
    int X, Y, len;
    float delta;

    len = 5;  /* one char is 8 pixel wide */

    Y = TM + 3 + 8;
    X = LM + 7*8;

    delta = (maxval - minval) / 5.0;

    /* Range -- 1 */
    sprintf( t1, "%-5d", minval );
    gdImageString( im, X, Y, t1, black );

    /* Range -- 2 */
    X = X + 6*8;
    sprintf( t1, "%-5d", (int)(minval+delta+0.5) );
    gdImageString( im, X, Y, t1, black );

    /* Range -- 3 */

    X = X + 6*8;
    sprintf( t1, "%-5d", (int)(minval+2*delta+0.5) );
    gdImageString( im, X, Y, t1, black );

    /* Range -- 4 */
    X = X + 6*8;
    sprintf( t1, "%-5d", (int)(minval+3*delta+0.5) );
    gdImageString( im, X, Y, t1, black );

    /* Range -- 5 */
    X = X + 6*8;
    sprintf( t1, "%-5d", (int)(minval+4*delta+0.5) );
    gdImageString( im, X, Y, t1, black );

    /* Range -- 6 */
    X = X + 6*8;
    sprintf( t1, "%-5d", maxval );
    gdImageString( im, X, Y, t1, black );

}

/*-----------------------*/
/* main() starts here    */
/*-----------------------*/
int main( argc, argv )
int argc;
```

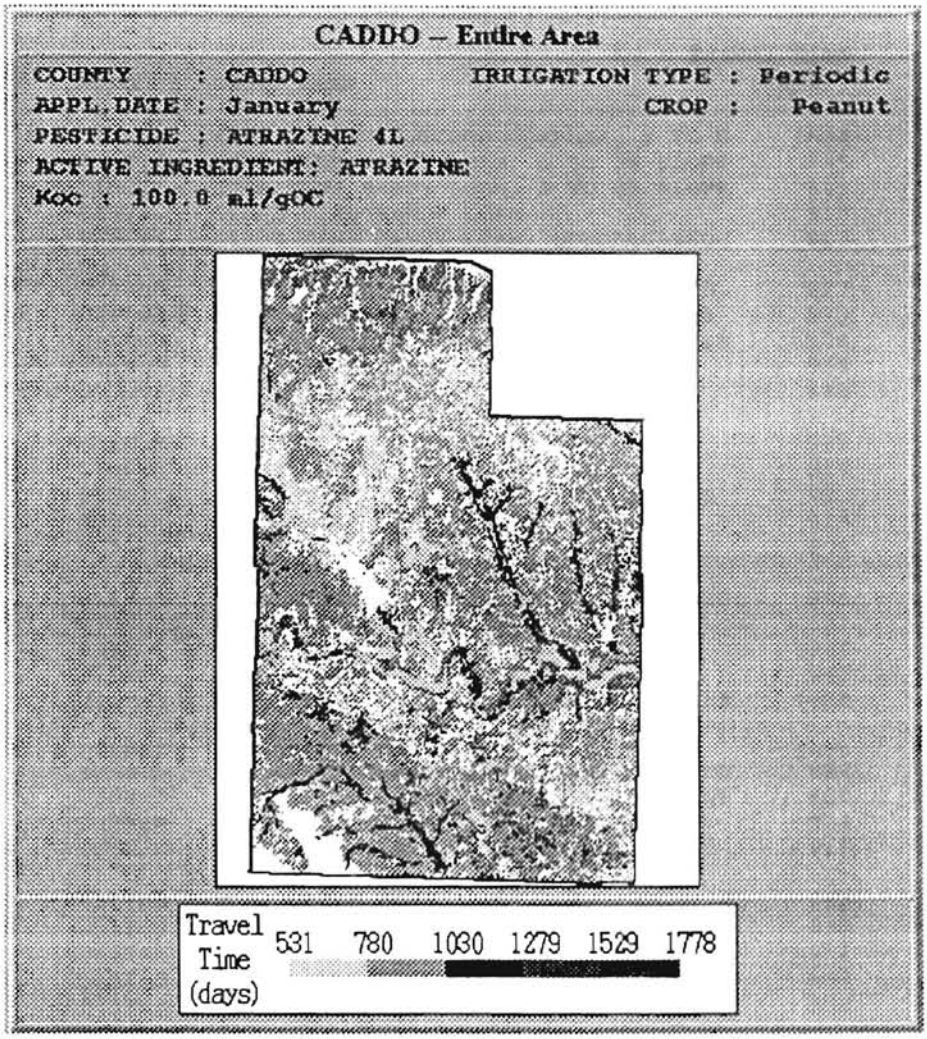
```
char *argv[];
{
    FILE *out;
    gdImagePtr im;
    int ch, x, y, xx;
    char buf[81];
    FILE *fp;
    int lines, Y;
    char gif_file[101];

    if ( argc != 4 )
    {
        printf("Needs 3 arguments!\n");
        exit(-1);
    }

    /*-------------------------------------*/
    /* argv[1] holds gif file name         */
    /* argv[2] holds the min value         */
    /* argv[3] holds the max value         */
    /*-------------------------------------*/
    strcpy( gif_file, argv[1] );
    minval = atoi( argv[2] );
    maxval = atoi( argv[3] );

    if ( minval < 0 )
        minval = 0;

    if ( maxval <= minval )
    {
        printf("Max value <= min value !\n");
        exit(-1);
    }

    /* create the image in memory */
    im = gdImageCreate( GIF_WIDTH, GIF_HEIGHT );

    /* define the colors */
    white = gdImageColorAllocate(im, 255, 255, 255);
    red = gdImageColorAllocate(im, 255, 0, 0);
    blue = gdImageColorAllocate(im, 0, 0, 255);
    cyan = gdImageColorAllocate( im, 0, 255, 255 );
    green = gdImageColorAllocate( im, 0, 255, 0 );
    yellow = gdImageColorAllocate( im, 255, 255, 0 );
    black = gdImageColorAllocate( im, 0, 0, 0 );
    brown = gdImageColorAllocate( im, 128, 64, 0 );
    magenta = gdImageColorAllocate( im, 255, 0, 255 );
    purple = gdImageColorAllocate( im, 0, 0, 128 );

    /* define the color table */
    colors[0] = yellow;
    colors[1] = cyan;
    colors[2] = blue;
    colors[3] = magenta;
    colors[4] = brown;

    /* draw the frame */
    DrawRectangle( im, 0, 0, GIF_WIDTH-1, GIF_HEIGHT-1, blue );

    /* draw the color bars */
    DrawBar( im );

    /* draw range texts */
    DrawRange( im );

    /* dump the image to gif file */
    out = fopen( gif_file, "w+" );
    gdImageGif( im, out );
    fclose( out );

    /* clean up memory */
    gdImageDestroy( im );

    return 0;
}
```

APPENDIX E

AN OUTPUT SCREEN

Maps for WWW -- Output

Travel Time (Probability = 10%)



Travel time for **ATRAZINE** to reach a depth of 1 meter for different soils in **CADDO** county Computing the travel time for applications in many different years indicates that the times shown below were exceeded for **10%** of the applications; travel times for **90%** of the applications were less than the values shown.

COUNTY : CADDO **IRRIGATION TYPE : Periodic**
APPL.DATE : January **CROP : Peanut**
PESTICIDE : ATRAZINE 4L
ACTIVE INGREDIENT: ATRAZINE
Koc : 100.0 ml/gOC

Muid	Soil Name	Travel Time (days)
015AgD	ACME-GYPSUM OUTCROP COMPLEX, 2 TO 8	1634
015CoB	COBB FINE SANDY LOAM, 1 TO 3% SLOPE	545
015CoC	COBB FINE SANDY LOAM, 3 TO 5% SLOPE	545
015Cs	CYRIL FINE SANDY LOAM	910
015Cy	CYRIL FINE SANDY LOAM, NONCALCAREOU	917
015DnD	DARNELL-NOBLE ASSOCIATION, ROLLING	827
015DoB	DOUGHERTY LOAMY FINE SAND, 1 TO 3%	573
015DuD	DOUGHERTY AND EUFAULA LOAMY FINE SA	576
015FoA	FOARD SILT LOAM, 0 TO 1% SLOPES	948
015Gm	GRACEMONT SOILS	545
015GrB	GRANT LOAM, 1 TO 3% SLOPES	925
015GrC	GRANT LOAM, 3 TO 5% SLOPES	924
015GrC2	GRANT LOAM, 3 TO 6% SLOPES, ERODED	910
015GrD	GRANT LOAM, 5 TO 8% SLOPES	924
015GwC	GRANT-WING COMPLEX, 1 TO 5% SLOPES	904
015HoA	HOLLISTER SILT LOAM, 0 TO 1% SLOPES	945
015KoC2	KONAWA LOAMY FINE SAND, 1 TO 5% SLO	531
015Mc	MCLAIN SILTY CLAY LOAM	966
015Me	MILLER SILTY CLAY LOAM	947
015MoD	MINCO VERY FINE SANDY LOAM, 3 TO 8%	1628
015MsC	MINCO SILT LOAM, 3 TO 5% SLOPES	1644
015NoB	NOBLE FINE SANDY LOAM, 1 TO 3% SLOP	894
015NoD	NOBLE FINE SANDY LOAM, 3 TO 8% SLOP	894
015NrB	NORGE SILT LOAM, 1 TO 3% SLOPES	924
015NrC	NORGE SILT LOAM, 3 TO 5% SLOPES	919
015PcA	POND CREEK FINE SANDY LOAM, 0 TO 1%	938
015PcB	POND CREEK FINE SANDY LOAM, 1 TO 3%	938
015PkA	POND CREEK SILT LOAM, 0 TO 1% SLOPE	952
015PkB	POND CREEK SILT LOAM, 1 TO 3% SLOPE	953
015PkB2	POND CREEK SILT LOAM, 1 TO 3% SLOPE	938
015Po	PORT SILT LOAM	1261
015Pu	PULASKI SOILS	538
015ReA	REINACH SILT LOAM, UPLAND, 0 TO 1%	1627
015ReB	REINACH SILT LOAM, UPLAND, 1 TO 3%	1626
015RhA	REINACH SILT LOAM, 0 TO 1% SLOPES	1778
015ShB	SHELLABARGER FINE SANDY LOAM, 1 TO	682
015ShC	SHELLABARGER FINE SANDY LOAM, 3 TO	673
015TlB	TILLMAN SILTY CLAY LOAM, 1 TO 3% SL	939
015TlC	TILLMAN SILTY CLAY LOAM, 3 TO 5% SL	939
015TlC2	TILLMAN SILTY CLAY LOAM, 2 TO 5% SL	919
015WuC	WOODWARD-QUINLAN COMPLEX, 3 TO 5% S	1236
015Ya	YAHOLA SOILS	540

VITA

TAO ZHU

Candidate for the Degree of

Master of Science

Thesis : A WWW INTERFACED DECISION SUPPORT SYSTEM THAT INTEGRATES RDBMS AND GIS WITH WWW SERVER

Major Field : Computer Science

Biographical :

Personal Data : Born in Liyang, Jiangsu, People's Republic of China, January 31, 1969, the son of Limin Zhu and Xiuying Zhao.

Education : Graduated from Tianjin No. 3 middle school, Tianjin, P. R. China, in August 1987; received Bachelor of Science Degree in Computer Science from Peking University, Beijing, P. R. China in July, 1991; completed requirements for the Master of Science degree at Oklahoma State University in December, 1996.

Professional Experience : Programmer, Department of Agronomy, Oklahoma State University, August, 1995 to November, 1996; Software Engineer, VLSystems Inc., Irvine, CA, February, 1995 to July, 1995 ; Programmer, Project Director, Fourth Shift Asia, Tianjin, P. R. China, May, 1993 to February, 1995; Software Engineer, Tianjin Navigation Instruments Institute, Tianjin, P. R. China, September, 1991 to May, 1993.