

DEVELOPING A WINDOWS GUI APPLICATION
FOR A WATER QUALITY MODEL
WITH VISUAL BASIC 6

By

ZHENGPING YAN

Bachelor of Science

Tianjin University

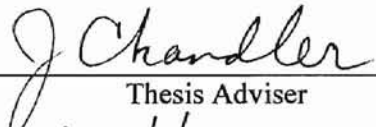
Tianjin, China

1987

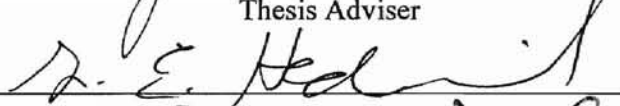
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
In partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2000

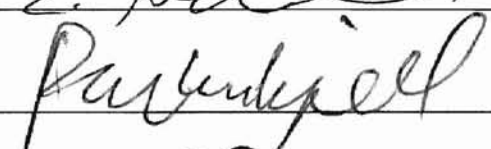
DEVELOPING A WINDOWS GUI APPLICATION
FOR A WATER QUALITY MODEL
WITH VISUAL BASIC 6

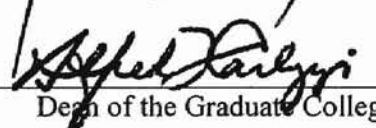
Thesis Approved:



Thesis Adviser







Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my major advisor, Dr. J. P. Chandler for his supervision, guidance, and support. My sincere appreciation extends to my other committee members Dr. G. E. Hedrick and Dr. N. Park, whose guidance and assistance are also invaluable.

I would also like to thank my son, Spencer, and my parents for their love, support and encouragement.

Finally, I would like to thank the Department of Computer Science for providing me financial support during my two years of study.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	1
Statement of the Problem	1
Objects and General Approach	4
II. LITERATURE REVIEW	6
Graphical User Interface	6
Visual Basic	10
III. DESKTOP TMDL APPLICATION	13
Design of User Interfaces	13
Embedded Model	15
Splashing Window	16
Modeler, Facility and Stream information and Water Quality Standards	17
Rate Constants	20
Wasteload Allocations, Minimum D.O. and Margin of Safety	22
Summarized Report and D.O. Plots	23
IV. HELP FOR THE DESKTOP TMDL MODEL	26
V. VALIDATION OF THE DESKTOP TMDL MODEL	29
VI. SUMMARY, RESULTS AND POSSIBLE FUTURE WORK	30
REFERENCES	32
APPENDIXS	34
APPENDIX A--SOURCE CODE FOR THE DESKTOP TMDL MODEL	34
APPENDIX B--REPORT AND PLOT FROM THE DESKTOP TMDL MODEL AND FROM THE ORIGINAL EXCEL MODEL	88

LIST OF FIGURES

Figure	Page
1. Part of Input Data Sheet of TMDL Model	2
2. Input screen of a DOS-based program	8
3. Input screen of a Visual Basic program	8
4. Embedded Excel Water Quality Model	16
5. Splashing Window for Desktop TMDL Model	17
6. Modeler, Facility and Stream Information and Water Quality Standards	18
7. Rate Constants	21
8. Wasteload allocation, minimum D.O. and Margin of Safety	22
9. Dissolved Oxygen (D.O.) Plots	24
10. Summarized Report of the TMDL Model	25
11. An Example of Standard HTML Language	27

CHAPTER I

INTRODUCTION

1.1 Statement of the Problem

Graphical user interfaces, or GUIs, have revolutionized the microcomputer industry [1]. The introducing of GUIs to microcomputers has changed not only the look of microcomputer operating system, but also the feel that applications developed for it have [2]. And at the same time, GUIs make using a computer a lot easier. Unlike the DOS prompt, users are presented with a desktop filled with little pictures called icons. With the help of the visual guide of these icons, users could use their computer knowledge together with their "common sense" to find out what the program will do. Nowadays, GUIs have become so popular that almost all personal computers, or PCs, have been equipped with a window user interface.

Staff in the Water Quality Division of the Oklahoma Department of Environmental Quality (ODEQ) have developed a Microsoft Excel-based desktop model for evaluation of water quality in the receiving stream of a wastewater treatment facility. This model is frequently used to determine the total maximum daily load (TMDL) of toxic pollutants that a wastewater treatment facility could discharge into a stream while ensuring that the water quality in the stream meets Oklahoma Water Quality Standards. The desktop model has been recognized and accepted by the Environmental Protection Agency (EPA)

as a tool for evaluating the total maximum daily load of wastewater facilities in Oklahoma. Therefore, it is not only used by state agencies for the purpose of regulation, but also used by some consulting companies, for instance INCOG in Tulsa.

	A	B	C	D	E	F	G	H	I	J
8	I	DISCHARGE INFORMATION								
9		LOCATION:	SEX\NV\SEX - S18 - T11N - R10W				BASIN:	520500		
10		COUNTY:	OKFULGEE							
12		DESIGN FLOW:	0.060 MGD							
13	II	RECEIVING STREAM: Battle Creek to Flat Rock Creek to North Canadian River								
14								4.30	MILES	
15		INTERMITTENT STREAM		7Q2=	0.00	MGD				
16		NUMBER OF SEGMENTS			40					
17		STREAM SLOPE (S)=			5.100	FT/MILE				
18		SIDE SLOPE (P)=			0.100	FT/FT				
19		MANNING'S N=			0.151					
20		VELOCITY COEFFICIENT Cv=			1.620					
21		DEPTH COEFFICIENT Ch=			0.191					
23		1.0 CFS	VELOCITY =		2.74	M/IDAY		0.17	FPS	
24			DEPTH =		0.12	METERS		0.40	FT	
25		0.0 CFS	VELOCITY =		1.48	M/IDAY		0.090	FPS	
26			DEPTH =		0.05	METERS		0.16	FT	
28	III	WATER QUALITY CRITERIA OF RECEIVING STREAM								
29		WARM WATER AQUATIC COMMUNITY								
30		AVERAGE D.O. REQUIREMENT				SUMMER	5.00	MG/L		
31						SPRING	6.00	MG/L		
32						WINTER	6.00	MG/L		
34	IV	UPSTREAM CONDITIONS								
35		ANY POINT/SIPS POLLUTION SOURCES?								
36		D.O. SATURATION			85.00	%				
37		UPSTREAM CBOD5			2.00	MG/L				
38		UPSTREAMNH3-N			0.15	MG/L				
41	V	RATE CONSTANTS at 20° C								
42				FLOW	1.0	CFS		0.0	CFS	
43		CBOD DECAY RATE (K1)								
44		(0.2*(H/8)^-0.434)		K1	0.40	/DAY		0.40	/DAY	
45		REAERATION RATES (K2)								
46		TSIVOGU	K2=CVS; C= 1.8		1.53	/DAY		0.83	/DAY	
47		TURNNEY-HARRIS								
48										
49			K2=1.33*S^0.32h^0.64		7.51	/DAY		7.51	/DAY	

Figure 1. Part of Input Data Sheet of TMDL Model

Since the model is developed on Microsoft Excel spreadsheets (See Figure 1 for example) and Excel is a very popular software application, people are finding the desktop model easy to use according to the ODEQ staff. Unfortunately, this good feature is accompanied by a major drawback: the model was often changed accidentally without realizing it by new users or even by some "old-hands" too.

Although Figure 1 shows only part of the input data sheet of the model, it is enough to demonstrate why mistakes are so easily made. 1) Not all numbers you can see are input data. Some numbers, such as cells F20 and F21, are calculated based on actual input data. Most numbers in the "Data" sheet are referenced in the other sheets of the model. Therefore, if an input datum is accidentally placed in a cell that is calculated using a formula, the entire model could be wrong. 2) To keep the report of the model neat and clear, a lot of cells in the "Data" sheet, for example cells B22, B24 and F43, are hidden in the background. Any unwanted changes in these hidden cells will also lead to false results. 3) Other sheets of the model may be changed accidentally too. 4) Certain options are made by changing a "hidden" cell (in the sense that the user cannot see the content of the cell). For example, the Water Quality Standards for a stream would be one of the following three options: Habitat Limited Aquatic Community, Warm Water Aquatic Community and Cool Water Aquatic Community. The option is made by changing the value of cell E30 to 1, 2 or 3, respectively, while the user can see nothing in cell E30 until he/she actually clicks on the cell. The same situation applies to choosing a reaeration formula and calculating the margin of safety. 5) There is no manual for the model (maybe it was considered too easy to need a manual). Thus, the potential users must be taught how to use the model in person.

Some of the above problems could be eliminated by locking the cells which should not be changed, then protecting the sheets of the model. However, the Margin of Safety feature of the model will not work when the sheets are protected. In reality, starting from the same model, different users ended up with slightly different models after using the original model for a few months. Therefore there is a need to convert the current TMDL model into a Windows application with good communications between users and model. Using Visual Basic to create such interfaces is probably the best solution.

1.2 Objects and General Approach

The object this thesis is to use Visual Basic (version 6) to develop Windows user interfaces for the TMDL model. This new TMDL model will have all of the functions the old model has with a completely new layout and new features, such as instructions and/or help, guiding users step-by-step through the whole model. This is to say that the new model will be easier to use and more user-friendly; and at the same time it will eliminate any unwanted changes that might happen during the use of the model.

All input parameters will be classified and grouped. Interfaces will be created based on the classifications of input parameters, and the same thing will be done for tabular and graphic outputs. There will be interfaces for inputting parameters and interfaces for displaying the modeling results. The model will be performed in such a manner that input parameters need to be frequently adjusted to make the simulation results in the model outputs meet certain water quality standards. Thus, communications among the interfaces will have to be established so that users could easily move from one interface to another to modify the input parameters, or to view the tabular or graphic results.

The following design components were considered in developing this water quality modeling application:

- The command menu such as “File”, “View”, “Window”, “Help”, etc.
- Icons for the frequently used commands, so that users would have alternative ways to manipulate the model.
- Bring to users’ attention (popping up a message box) if any input data are missing when a user runs the model.
- Provide users a typical range for input parameters if possible. This would be a great help for new users.
- Provide a Help document within a Help menu in the application.

CHAPTER II

LITERATURE REVIEW

Since the ultimate goal of the thesis is to develop a GUI application for the TMDL water quality model by using Visual Basic, in this chapter, GUI and Visual Basic will be briefly reviewed.

2.1 Graphical User Interface

GUIs have existed for almost 20 years. According to [3] [4], the first GUI interface was designed, tested, and extensively prototyped by a group of engineer at the Xerox Palo Alto Research Center in the late 1970's. However, it has been about 10 years since "the computer world at large became aware of the importance of GUIs" [5]. The Apple Macintosh is an example of an early GUI based operating system. GUIs are now the "dominant model for the user interface of the microcomputer operating systems" [6].

The definition of GUIs may be different in some aspects for users or corporations, but essentially the whole concept used is similar for all concerned. In [4], the first GUI interface is described as a display featuring windows, icons, drag-and-drop manipulation of objects, and formatted text. In [7], a GUI is defined as one that "runs in computer graphics mode". Alan Morse and George Reynolds [8] argue that a true GUI would include a complete representation of the system being controlled, and would support the ability to interact with the system through the representation. According to Microsoft, a

true GUI is one that offers a WYSIWYG (What You See Is What You Get) screen for output, that looks good and is easy to work with [9].

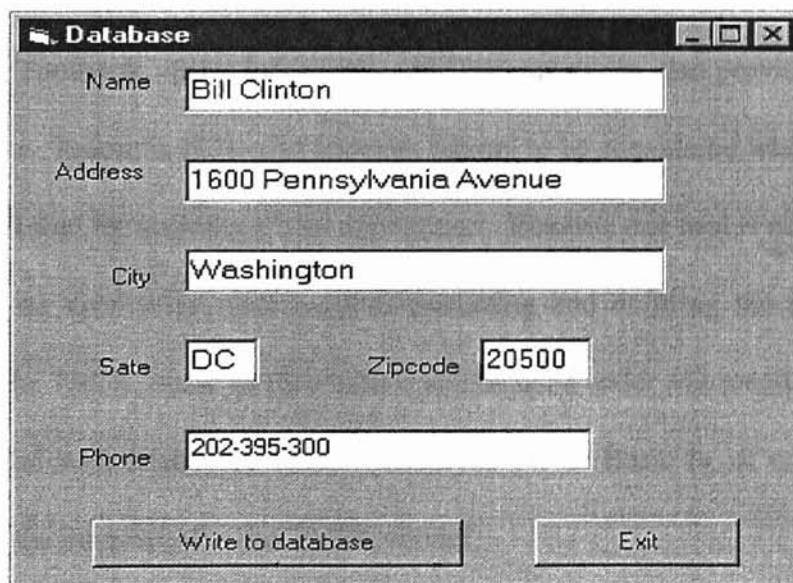
Ease of use is probably one of the most important factors which contributes to user's acceptance of GUIs. For a command-based or text-based interface, users will have to learn all commands needed to use a command-based interface. This might be difficult for users, especially if that user is a novice. Therefore, Tenopir [10] suggested that command interfaces are thought to be too complex for infrequent users. GUIs present users with windows, icons, menus etc., which make it easy to learn and to work with. Recognition and association of "real-world" objects are always easier than recalling commands. The field of human-computer interaction (HCI) has shown that GUIs have design components that fundamentally match the cognitive abilities, expectations, and limitations that a user brings to a system [4]. In a command-based interface, a date has to be presented as month/day/year. It can be presented as a calendar in GUIs, letting the user point and click on a desired date [11]. There is no doubt that the GUI presentation of a date, which offers a visual object, is easy to understand and looks much better because it is closer to the user's real-life experiences.

Along with the revolution in how programs look was a revolution in how they feel. In [2], the author gives an excellent example illustrating how the relationship between user and program is different for a DOS-based program (command-based interface) and a Visual Basic program (a GUI application). In the DOS-based program, the six pieces of data have to be input one at a time, with no opportunity to go back and alter previously entered information (Figure 2). The program is in control in this case. Figure 3 shows how an equivalent Visual Basic program gets the same information. The box may be

filled in any order. The user can either type in new information or edit existing information. When the user is satisfied that all the information is correct, he or she just clicks on the "Write to Database" button to update the database. The user is in control in this VB program [2]. In other words, the Visual Basic program is more user-friendly.

```
Enter Name (Enter EOD to terminate): Bill Clinton
Enter Address: 1600 Pennsylvania Avenue
Enter City: Washington
Enter State: DC
Enter Zipcode: 20500
Enter Phone Number: 202-395-3000
```

Figure 2. Input screen of a DOS-based program



The screenshot shows a window titled "Database" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains a form with the following elements:

- Name:
- Address:
- City:
- State: Zipcode:
- Phone:
- Buttons: "Write to database" and "Exit"

Figure 3. Input screen of a Visual Basic program

In sum, the benefits of the GUIs are obvious. The GUIs make it easy for users to learn and to use programs. A well-designed GUI application would offer users more flexibility and is more user-friendly. But the potential benefits of GUIs are much more than what we have mentioned here. The GUI, originally introduced primarily to provide a visual metaphor for an operating system, has already changed our culture in surprising ways.

When it comes to build a new application, one has to decide which GUI development tool to use if he or she does not want to build it manually. A variety of development tools are currently available. In [12], the authors present considerations in how to choose an appropriate tool when designing a GUI application. Julien [13] lists five GUI development products for Windows which represent a cross-section of the market. These products include: 1) Microsoft Visual Basic, 2) Within Technologies' Realizer, 3) Borland International Inc.'s Turbo Pascal for Windows, 4) Borland's ObjectVision, and 5) Asymetrix's Toolbook. Brief information and comments are also provided for each of the five products. Julien, in [13], also presents factors to be considered when choosing an appropriate GUI tool for building a new application. Because one tool is not suited for all jobs, choosing an appropriate tool requires analyzing and defining the problem to be solved. Since the TMDL water quality model will be used under Microsoft Windows and maybe associated with Microsoft Excel, Microsoft Visual Basic is, of course, the first choice to build the interfaces for the TMDL model.

Head, in [4], presents the design components that a well-designed GUI application should have from a cognitive standpoint. These components include:

- Providing multiple methods for completing the same task - this allows a user to choose one that's easiest for the individual.
- Unfurling commands and menu options as needed - this limits a user's memory overload.
- Using icons that rely on users' recognition of "real-world" objects.
- Creating visual, auditory, or tactile feedback that quickly sends information back to users.
- Enhancing screen visuals with color, font, shape, arrangement, and contrast - these all can help a user focus attention on tasks at hand.

These will be used as a guide in designing the interfaces for the Microsoft Excel-based TMDL water quality model.

2.2 Visual Basic

Microsoft Visual Basic is a popular GUI tool for developing Windows applications. The latest version, Visual Basic 6, has to work under the Windows 95, 98 or NT operating systems which account for about 95% of operating systems in personal computers. As its name suggests, Visual Basic is a visual programming language. But "Basic" in Visual Basic don't mean simple [14]. Instead, Visual Basic is based on a modern structured version of BASIC. Therefore, it is easy to build large applications by using modular and object-oriented techniques. In 1993, Robert A. DelRossi and Carla Mathews [15] compared four GUI development tools based on performance, documentation, technical support and so on. Visual Basic (version 2 at that time) was considered as the most savvy development tool. Furthermore, according to [1], Visual

Basic is even said to be “the perfect programming environment for the 1990s” in the *New York Times*. A more recent paper [16] shows that 53 percent of U.S. professional developers use Visual Basic. Visual Basic is not only the leading development environment but will continue to gain market share and popularity.

Visual Basic is fundamentally different from the conventional programming languages, which run from the top down. The core of a Visual Basic program is a set of independent groups of instructions that are activated by the events they have been told to recognize. Instead of designing a program to do what the programmer thinks should happen, users could, to some extent, decide what a Visual Basic program will do.

Visual Basic 6 offers many useful features for developing a good Windows application. According to [2], two features that make Visual Basic different from almost any other programming tool are:

- We literally draw the user interface, much like using a paint program.
- After drawing the interface, the command buttons, text boxes, and other objects that are in a blank window automatically recognize user actions such as mouse movements and button clicks.

A Visual Basic interface consists of various objects, such as command buttons, text boxes, option box and so on. These objects are independent from each other. The programmer will have write code to glue them together to form a complete application. Each object has its properties and methods. It also has its associated events. Visual Basic controls these objects' behaviors through manipulating objects' properties and

methods. Event procedures present the handle to users to let them decide when they want an event to happen and what event will happen.

Visual Basic allows programmers to build standalone applications, ActiveX Client or ActiveX server applications. Once programming of an application is finished, Visual Basic's Setup Wizard can help programmers create a professional looking setup program for the distribution of the application.

Visual Basic 6 also provides sophisticated error handling for the all-too-common task of preventing users from bombing an application [1]. So far, error handling techniques are basically limited within procedure level [17]. One of the most common uses of error handling is to deal with opening a file, saving to a file, etc. If a user specifies a file to open and the file does not exist, or if a user specifies a path to save a file and the path is not correct, a run-time error will be raised. An error handler is the best place to write code to deal with this type of situations.

In addition, ActiveX is another powerful feature which allows users to build their own controls. Microsoft invites all parties to write powerful, reusable controls for Visual Basic. This makes Visual Basic even more powerful. But programmers had to use other programming languages to write the controls until the release of Visual Basic 5.

Since Visual Basic is a widely used tool, its references are numerous. Among them, I found books by Howard Hawhee [17][18] and by Peter G. Aitken [20] are excellent references. For up-to-date new features of Visual Basic, one could refer to Microsoft's Visual Basic web site [19]. The Visual Basic library on this site allows one to find functions which cannot be found in most reference books.

CHAPTER III

DESKTOP TMDL APPLICATION

3.1 Design of User Interfaces

The goal of this application is to provide users an easy to follow user interfaces and to prevent the model from being changed accidentally by its users. Based upon this goal, it was decided to embed the Excel-based water quality model in the application. As a result, users will still be able to view the model, but not be able to make any changes to the model since the model is part of executable file of the application.

Based on the suggestions of DEQ staffs, input data to the model were divided into following categories:

- Information about modeler
- Facility information
- Stream characteristics
- Water quality standards applying to the stream
- Rate constants
- Wasteload allocation for the facility (proposed permit limits)
- Dissolved Oxygen (D.O.) output and safety factor
- Plots of D.O. profile in the stream

These categories sort the input information out in a well organized manner, which gives users a clear path to follow when inputting information to the model. Therefore, the

same categories were carried over to the application user interfaces.

This desktop TMDL model has six Visual Basic (VB) standard forms. All six forms are contained in a parent form, MDIForm1 (Multiple-Document Interface Form). The names and contents of the six child forms are:

1. frmModel – Hosting the embedded model
2. frmSplash – A cover-up form when the application was opened at the first time.
3. frmTmdl1 – modeler and facility information, stream characteristics and Water Quality Standards
4. frmTmdl2 – Rate constants
5. frmTmdl3 – Wasteload allocations, minimum D.O. & safety factor
6. frmTmdl4 – Summarized report & D.O. Plots

The affix 'frm' of each name is a type identifier to the variable name, which indicates the variable is a form object. Similar naming practice can be seen throughout the code of this application. For example, an integer 'A' would be named as 'intA'; a string variable 'Text' would be named as 'strText'; and so on. This practice is known as Hungarian Notation, so named for the nationality of its inventor, Charles Simonyi [17].

In addition to the above forms or interfaces, three Visual Basic standard modules were created to hold functions or procedures that might be used repeatedly in different interfaces. The use of standard modules was to avoid writing the same lengthy code for interfaces with the same action. One could simply make a function call or a procedure call to the function or procedure in standard module to perform the action. As a result, the code for the application can be reduced dramatically. Besides, standard module is also a good place to declare public variables that are needed throughout the application.

In the following sections, the creation and contents of each interface will be discussed in detail.

3.2 Embedded Model

There are two ways to connect the Excel based water quality model to its user interfaces. One is to create a link between user interface and the model, and the other is to embed the model in the interface.

When the Excel model is linked to this application, the Excel model will be stored in a separate file in Excel format. The application will be able to read data from or write data to the model. But at the same time, other applications can also read data from or write data to the model. Therefore, it is still under the risk that the model may be changed unintentionally.

When the Excel model was embedded into this application at the design time, it will become part of the application's exe file after compiling. Only programmer of the application can make changes to the embedded model at the design time. After compiling of the TMDL application, no one will be able to change the original model. This leaves users no chance to make any changes to the model. The original model will be completely safe.

The Excel model was embedded in this interface through an Object Linking and Embedding (OLE) container (Figure 4). This interface is running in the background and a user normally does not see it. However, if a user is really interested to see how the model looks like, he/she could view the model from View menu. This is the only way to view the model. After viewing the model, he/she can click on the another interface in the application or make a choice from menu to go back.

The screenshot shows a window titled 'TMDL Desktop Model' with a menu bar containing 'View' and 'Window'. Inside the window is an Excel spreadsheet with the following content:

JUSTIFICATION FOR SELECTION OF INPUT PARAMETER					
	CITY:	City of Stillwater			
	PERFORMED BY:	John Doe			
	DATE:	4/28/99			
I	DISCHARGE INFORMATION				
	LOCATION:	SE¼/NW¼/SE¼ - S18 - T11N - R10W	BASIN:	520500	
	COUNTY:	PAYNE			
	DESIGN FLOW:	0.250 MGD			
II	RECEIVING STREAM: Battle Creek to Flat Rock Creek to North Canadian River				
					4.30
	INTERMITTENT STREAM	7Q2=	0.00	MGD	
	NUMBER OF SEGMENTS		40		
	STREAM SLOPE (S) =		8.000	FT/MILE	
	SIDE SLOPE (P) =		0.100	FT/FT	
	MANNING'S N =		0.110		
	VELOCITY COEFFICIENT Cv =		2.055		
	DEPTH COEFFICIENT Ch =		0.169		
	1.0 CFS	VELOCITY =	4.36	M/DAY	0.27
		DEPTH =	0.11	METERS	0.36
	0.0 CFS	VELOCITY =	3.17	M/DAY	0.193
		DEPTH =	0.07	METERS	0.22

Figure 4. Embedded Excel Water Quality Model

3.3 Splashing Window

When a user try to open an application, it may take a few seconds to load the application into the memory. Therefore, there may be a gap between a request to open an application and the application actually opened. Many programmers are using a splashing window to fill this gap [17]. Indeed, most Microsoft applications, if not all, have such a window. Word, Excel, PowerPoint and Access etc. all have a splashing

window. Visual Basic has one too. Since an Excel model was embedded in this TMDL model application, it was expected to take a while to load everything into memory. Therefore, a splashing window is also provided for this application (Figure 5).

A splashing window often contains products name, version, Company name and distribution right etc. Visual Basic 6 provides a template to create a splashing window. The splashing window for this application is modification of the template.

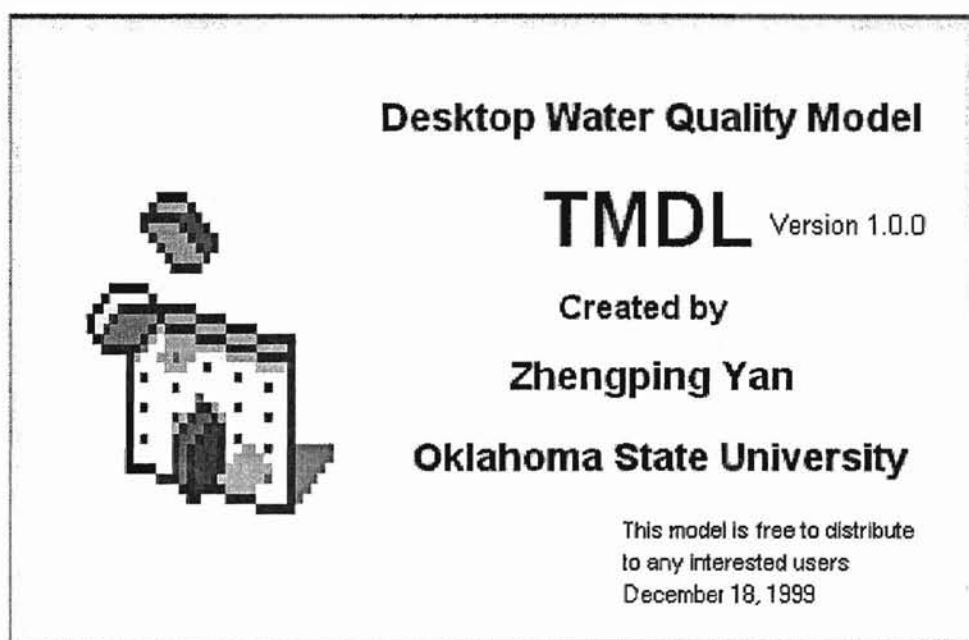


Figure 5. Splashing Window for Desktop TMDL Model

3.4 Modeler, Facility and Stream information and Water Quality Standards

In the previous section, input information to the water quality model is divided into eight categories. Of them, modeler and facility information, stream characteristics, water quality standards for the stream are less likely to be changed again once input because

they are very much fixed for a given facility. Therefore, within a manageable space, they are placed in one interface so that user does not have to move to this page very often once data on this page were input.

The screenshot shows a software window titled "TMDL Desktop Model - [TMDL#1 - 1]". The window has a menu bar with "File", "View", "Run", "Window", and "Help". Below the menu bar are navigation buttons: "First Page", a left arrow, a right arrow, and "Last Page". On the right side of the window are "Run" and "View Plots" buttons. The main area is divided into three sections:

- Modeler:** Includes "Model Performed By:" (John Doe), "Date:" (04/28/99), and "Company/Agency Name:" (Water Quality Division, Oklahoma Department Of Environmental Quality).
- Facility & Discharge:** Includes "Facility Name:" (City of Clinton), "Stream Segment #:" (410200), "Legal Description:" (SE/SW/SE S18, T11N, R10W), "Design Flow (MGD):" (0.06), "County:" (Custer), and "State:" (Oklahoma).
- Receiving Stream:** Includes "Name of Receiving Stream:" (Unnamed Tributary to Washta River), "Low Critical Flow, 7Q2 (MGD):" (0.0), "Stream Length (miles):" (5.0), "Stream Slope (ft/mile):" (5.1), "Stream Side Slope (ft/ft):" (0.1), "Manning's Number:" (.10), and a "Classification of Stream" section with three radio button options: "Habitat Limited Aquatic Community", "Warm Water Aquatic Community" (which is selected), and "Cool Water Aquatic Community".

Figure 6. Modeler, Facility and Stream Information and Water Quality Standards

Water quality standards for a stream can be selected by click the corresponding option box. All other information on this page needs to be input through labeled textboxes (Figure 6). Information was grouped according to its category. Thus it is easy for a user to follow. Since every piece of information on the user interface was labeled

properly, user would immediately know what data was needed for every entry. Therefore, new users could learn how to use this model by themselves.

After input data in one textbox, user can press Tab key to go to the next textbox or click on the next textbox. These features come with Visual Basic, which means a programmer does not need to write any code to make those happen. In practice, after inputting one piece of data, a lot of computer users would press the "Enter" or "Return" key. Based on this understanding, some codes were written to recognize "Enter" key so that pressing "Enter" key would also take user to next data entry.

When a user moves to a data entry, all text in the entry will be automatically select so that he/she can immediately start enter data if he/she wants to wipe out the old data. Another good feature of entering data is when a entry is being edited the color of it's text will turn red. The color makes it easy for a user to spot where he/she was when he/she came back from doing something else. When user leave the entry, the color of its text will changed back to black again. So, there is only one red color entry at a time.

On the top of this interface are six icons that allow users to navigate from one interface to another. Right arrow icon takes users to the next page and left arrow icon takes users to the previous page; first page icon brings users to the first input interface and last page icon leads brings users to the last input interface. Since this is the first input data page, it does not make sense to go its previous page or run model. Therefore, these icons were grayed out. To be consistent with other input pages, these icons were still kept on the page so that all input pages would have the same set of icons.

A menu was designed for each user interface. The menu was made as close as possible to common menus [22], such as menu of Word, Excel or WordPerfect etc. As a

result, users who are already familiar with the common applications would be able to apply their knowledge on menus to this Desktop TMDL Model application directly. It is like they don't have to learn how to drive again when they get a new car.

All of these efforts are to make this Desktop TMDL Model as user friendly as possible. The features described above also apply to other user interfaces in this application. Thus, they will not be repeated in the following sections.

The code for this interface can be found in Appendix A. Some functions or subroutines called in this interface's code can be found in the section of standard modules of Appendix A.

3.5 Rate Constants

This interface holds all water quality rate constants to the desktop model. These rate constants were grouped in the same way as in the original Excel spreadsheet model so that the users who ever used the original spreadsheet model will find these parameters arranged in a familiar and logical way.

As in the spreadsheet model, the reaeration rates are calculated by four different formulas. Only one formula will be selected for the water quality calculations later in the model. Therefore, a group of option boxes were used to serve this purpose. The values of reaeration rates are listed besides each formula. Once a user selects a formula, the value corresponding to the selected formula will become bold. If a user changes his/her selection to another formula, the values of the previous selected formula will be set back to normal and the values of currently selected will become bold. This is easier for a user to see which formula has been selected.

Since reaeration rates are determined by stream and flow characteristics, any changes on stream or flow parameters will result in a change in reaeration rates. Therefore, if a user makes any changes on the stream or flow parameters, it will automatically deselect the option made on reaeration rates. And the user will need to reselect a formula based on the updated values for each formula.

The screenshot shows the 'Rate Constants of the Desktop Model' window. It contains two main sections: 'Rate Constants at 20° C' and 'Reaeration Rates, K2'.

Rate Constants at 20° C

CBDD Decay Rate (/day)	0.4		
	SUMMER	SPRING	WINTER
Sediment Oxygen Demand, SOD (g/l ² /d)	0.06	0.075	0.1
CBDD Settling Rate, KS (/day)	0.03	0.03	0.03
NBDD Decay Rate, KN (/day)	0.3	0.3	0.3

Reaeration Rates, K2

	0.0 cfs	1.0 cfs
<input type="radio"/> Tsvoglu Formula (/day)	1.13	2.09
<input type="radio"/> Turney-Harris Formula (/day)	9.78	9.78
<input type="radio"/> Texas Formula (/day)	13.38	6.93
<input type="radio"/> Owens et al. Formula (/day)	209.41	57.22

Hint: Select One of Formulas to View Aeration Rates in Above Text Boxes.

Figure 7. Rate Constants

3.6 Wasteload Allocations, Minimum D.O. and Margin of Safety

Once a user inputs all of the information in the above two user interfaces, he/she can use this interface to determine the proper wasteload allocations for a wastewater discharge. This interface allows users to change the wasteload for a discharge and to run the model to see instream D.O. and Margin of Safety (Figure 8). If the instream water meets water quality standards, the Margin of Safety will be calculated. Otherwise, a message box will pop up telling the user what needs to be done in order to meet water quality standards, and then calculations will be halted.

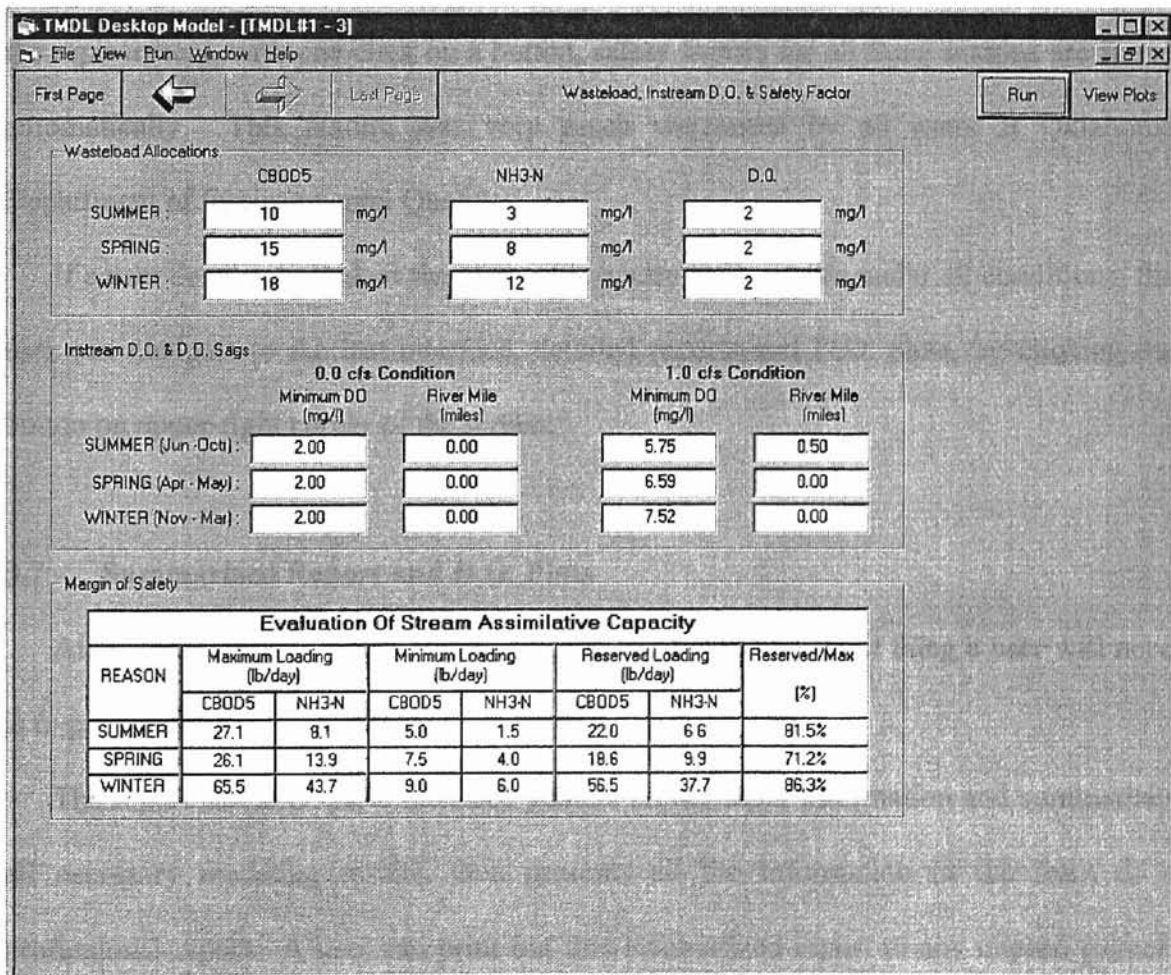


Figure 8. Wasteload allocation, minimum D.O. and Margin of Safety

The way the margin of safety is calculated is completely different from that in the original spreadsheet model. The margin of safety was computed using Excel's build-in function "Solver". The inconvenience comes from the fact that the user has to respond to the message box by click on "Yes" or "No" button for each season (spring, summer and winter). In this application, based on the monotone relationship between the wasteload and the margin of safety, a new procedure was written to find the margin of safety. Actually, once the new procedure was written, it replaced the recorded macro (using Solver function to find the margin of safety) in the spreadsheet model immediately. After the replacement, with one click on a button, safety factors for all three seasons are found automatically. This feature was very much welcomed by all users at Oklahoma Department of Environmental Quality.

If a user decides to look at the plots of instream D.O. profile under all conditions, the user can navigate to the last interface, detailed reports and D.O. plots, by clicking the button on upper-right corner of the screen.

3.7 Summarized Report and D.O. Plots

After a successfully model simulation, the last but not the least thing a user will need is to print out the modeling results.

The report and D.O. plots interface gathers all the input information and summarizes all necessary modeling results, then presents all the information in the form of a professional report. A user can print out this summarized report to any chosen printer. The printouts consist of two parts: five pages of description/tabular information and one page of D.O. plots. Figures 9 and 10 shows part of the dissolved oxygen plots and part of

the first page of the summarized report, respectively. A complete summarized report can be found in Appendix B.

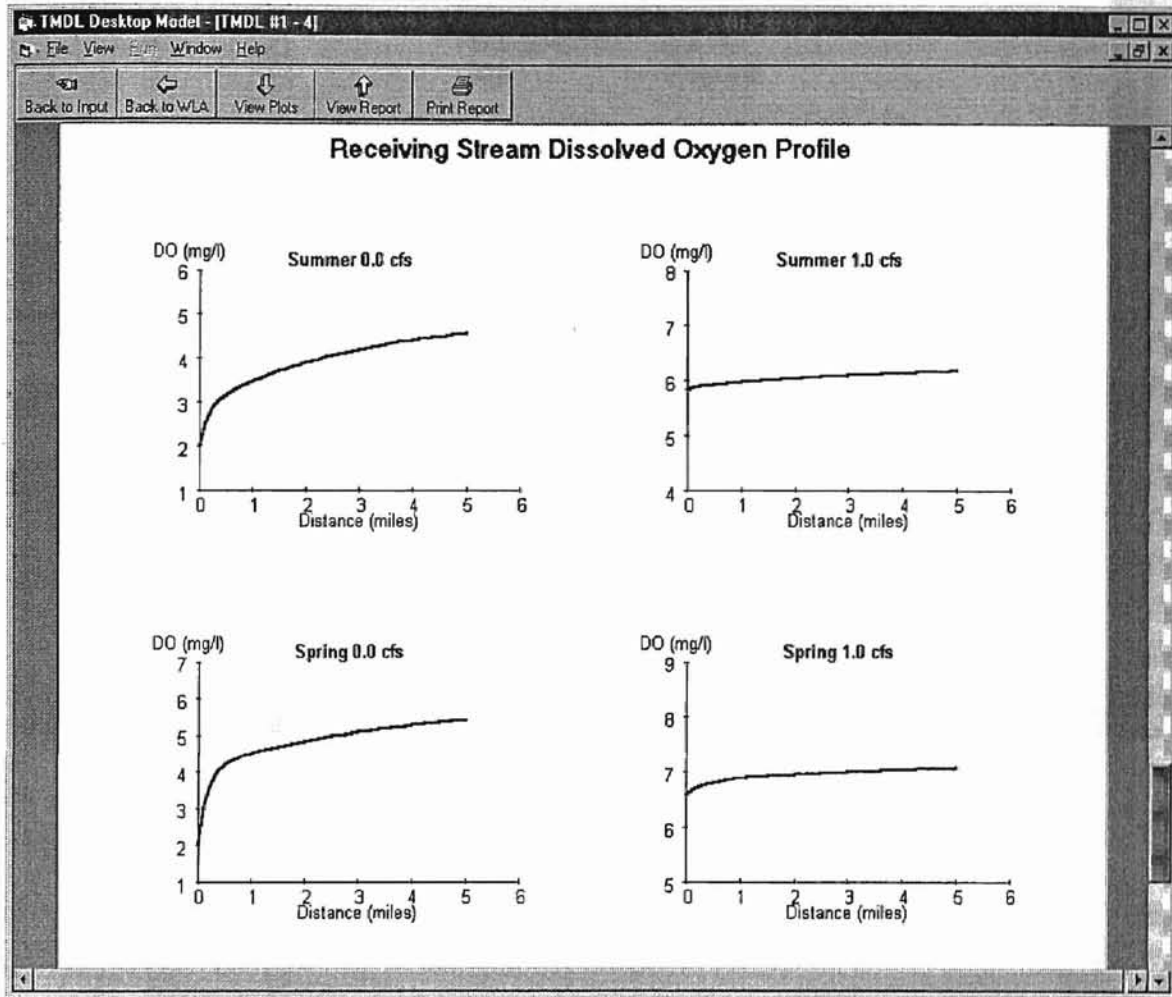


Figure 9. Dissolved Oxygen (D.O.) Plots

As shown in Figures 9 and 10, users can utilize either the icons on the top of the screen or the menu of this interface to navigate back to modify the model, if necessary, or to print out the whole report.

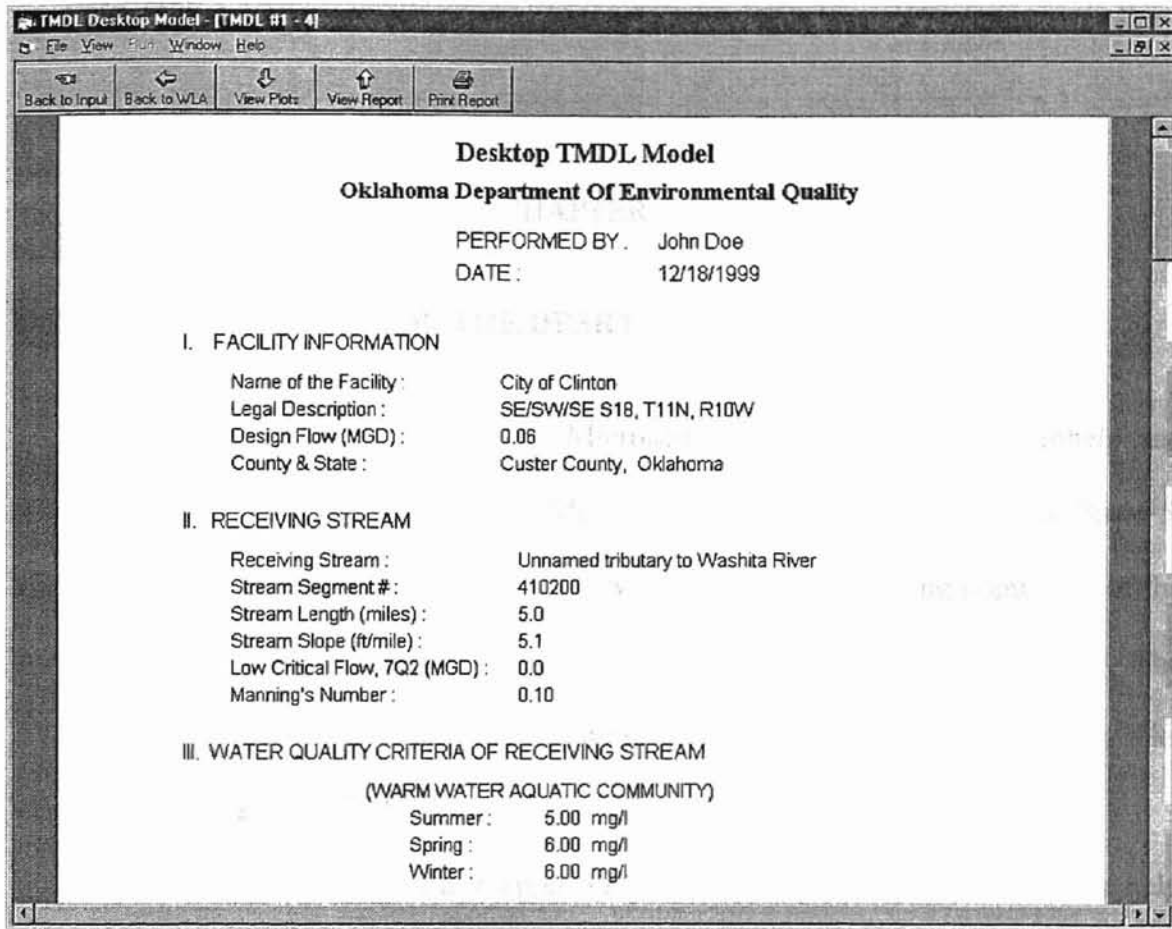


Figure 10. Summarized Report of the TMDL Model

CHAPTER IV

HELP FOR THE DESKTOP TMDL MODEL

There are two types of help files in Microsoft window applications: Winhelp and HTML help. Winhelp is used in many Microsoft applications, such as Visual Basic 5, Microsoft Word, Excel etc. in Office 97. However, with the increasing popularity of the Internet, Microsoft has changed its applications' help to an HTML help system. The HTML help system has now become the new Microsoft standard [18]. For example, Visual Basic 6 uses an HTML help system.

A help file for your Visual Basic application needs to be created with a separate help workshop. An HTML help workshop is a part of the standard package in Visual Basic 6.0 (Professional Edition or Enterprise Edition). For an earlier version of VB, for example VB 5.0, one could download an HTML Help Workshop from Microsoft's web site [21]. The help file for the Desktop TMDL Model was created using this workshop.

Before creating an HTML help file for a Visual Basic application, one has to create a standard HTML file (with the extension of "html") for each help topics he/she plans to provide in the application's help system. Figure 11 is a simple example of such standard HTML language, which is also one of the help topics for the Desktop TMDL model.

When creating an HTML help file, another very important issue is having a unique Help Context ID for each help topic. The Help Context ID can be any number, but must be unique. Based on the contents, help topics are often classified for different levels.

here. Interested readers can find such information in Howard Hawhee's book, *MCS D Visual Basic 6 Exams 70-175 and 70-176* [18].

Once the HTML help file is created, it must be connected to the TMDL application. This can be easily done in Visual Basic 6 by pointing the HelpFile of the TMDL project property to the help file being created with the HTML help workshop. Now, when the TMDL application is running, a user can press the F1 key to get help information, just like other Microsoft Windows applications such as Word, Excel etc.

A user can also get help information through a Help menu. There is a little trick when making a help menu work. If we can make the action of clicking on the help menu equivalent to the action of pressing the F1 key, the help information will show up when the user clicks on the help menu. This can be done by a function called "SendKey". Most Visual Basic books don't have any information about functions like "SendKey", but interested readers can find them in the Microsoft Visual Basic web site [19].

CHAPTER V

VALIDATION OF THE DESKTOP TMDL MODEL

Any application must be well tested before it can be delivered to the users. There will be no exceptions for the Desktop TMDL Model application. Because we have known what results we should expect from the Desktop TMDL Model, the validation for this application is relatively straightforward. Basically, the validation was done through two steps: Carefully debugging and comparing the results of the TMDL application with the original Excel model results.

First, all of the codes were carefully checked for any type of errors, from editorial errors to logical errors. Any inappropriate codes found were corrected. In addition, the application was well tested to make sure all icons, menu items, help information and so on worked correctly.

Secondly, the results of the Desktop TMDL Model were compared cautiously with that of the original Excel model with the same input information (Appendix B shows a complete result of this Desktop TMDL Model and a complete result of the original Excel model). For all different sets of input information we have ever tried, we always got the identical results from the above-mentioned two models. This fact proves that the Desktop TMDL Model application is working the way it was designed to and therefore is a good valid application to use.

CHAPTER V

VALIDATION OF THE DESKTOP TMDL MODEL

Any application must be well tested before it can be delivered to the users. There will be no exceptions for the Desktop TMDL Model application. Because we have known what results we should expect from the Desktop TMDL Model, the validation for this application is relatively straightforward. Basically, the validation was done through two steps: Carefully debugging and comparing the results of the TMDL application with the original Excel model results.

First, all of the codes were carefully checked for any type of errors, from editorial errors to logical errors. Any inappropriate codes found were corrected. In addition, the application was well tested to make sure all icons, menu items, help information and so on worked correctly.

Secondly, the results of the Desktop TMDL Model were compared cautiously with that of the original Excel model with the same input information (Appendix B shows a complete result of this Desktop TMDL Model and a complete result of the original Excel model). For all different sets of input information we have ever tried, we always got the identical results from the above-mentioned two models. This fact proves that the Desktop TMDL Model application is working the way it was designed to and therefore is a good valid application to use.

CHAPTER VI

SUMMARY, RESULTS AND POSSIBLE FUTURE WORK

Visual Basic 6 was used in this study to build a window interface for an Excel-based water quality model. The Excel-based model has been recognized by the Environmental Protection Agency and has been used by Oklahoma Department of Environmental Quality and by other consultants for water quality assessment in Oklahoma. However, this model was not very user-friendly and undesired changes could be made to the model accidentally by its users. To overcome these drawbacks, a Windows application was developed in this study. The developed application was carefully tested and verified to insure that it would produce the same results as the original Excel-based model. In this new application, users are not permitted to change the underlying model so that a more desirable level security has been achieved. In addition, the application is easier to use, especially for new users. The new application was packaged into two 1.44 MB floppy disks for installation.

Since the original Excel-based model was embedded into the application, a PC must have a legal Microsoft Excel application in order to run the developed application. This is a limitation of this application. Fortunately, Excel is widely available on PCs. However, if the Excel-based model could be converted to an executable C program, it would not only eliminate the above limitation but also speed up the application.

The Help system provided in this application was meant mainly for demonstration.

Therefore, the contents of the Help system in the application certainly can be expanded to make the application better for users.

As for the Visual Basic 6, the application developed in this study is only one example of what Visual Basic can do. However, its real power and ability are far beyond what was used in this study.

REFERENCES

- [1] Aitken, Peter G., *Visual Basic 6 Programming Blue Book*, The Coriolis Group, Inc., 1998.
- [2] Brisco, Romona M., *A Multiple-Windows Interface For Internet Tools*, Oklahoma State University, M.S. Thesis, 1996.
- [3] Cornell, Gary, *Visual Basic 5 from the Ground Up*, Osborne/McGraw-Hill, New York, 1997.
- [4] DelRossi, Robert A. and Mathews, Carla *Component-based Development*, Software and Systems, April 1993, Volume 15, Number 15, pp. 69-75.
- [5] Gurewich, Nathan and Gurewich, Ori, *Teach Yourself Visual Basic 5 In 21 Days*, Fourth edition, Sams Publishing, Indianapolis, 1997.
- [6] Hawhee, Howard, *Microsoft Certified Solution Developer Visual Basic 6 Exam Guide*, Que Corporation, January 1998.
- [7] Hawhee, Howard, *MCSD Visual Basic 6 Exams 70-175 and 70-176*, New Riders Publishing, March 1999.
- [8] Head, Alison J., *A Question of Interface Design: How do Online Service GUIs Measure Up*, Online (www.onlineinc.com/onlinemag), May 1997, pp. 20-29.
- [9] Isaacson, Portia, *Visual Basic: Huge Opportunity For Free*, Computer Reseller News (www.crn.com), July 1997.
- [10] Julien, Shelly, *The Future of GUI Development Tools: Clearing Up the Confusion*, Software and Systems, July 1992, Volume 5, Number 4, p. 46.
- [11] Laudon, Kenneth C. and Laudon, Jane P., *Essentials of Management Information Systems*, Prentice Hall, Inc., 1995.

- [12] Leinfuss, Emily, *GUI Application Development*, Software and Systems, June 1993, Volume 15, Number 25, pp. 63-64.
- [13] Mandelkern, Dave, *GUIs - The Next Generation*, Communications of the ACM, April 1993, Volume 36, Number 4, pp. 36-39.
- [14] McKelvy, Mike, Spotts, Jeff and Siler, Brian, *Special Edition Using Visual Basic 5*, Second Edition, Que Corporation, December 1997.
- [15] Microsoft Visual Basic Web Site, <http://www.microsoft.com/vbasic>
- [16] Microsoft Web Page, *Microsoft HTML Help Workshop*, <http://www.microsoft.com/workshop/author/htmhelp>.
- [17] Morse, Alan and Reynolds, George, *Overcoming Current Growth Limits In UI development*, Communication of the ACM, April 1993, Volume 36, Number 4, pp. 73-80.
- [18] Raskin, Jeff, *Looking for Humane Interface: Will Computers Ever Become Easy to Use*, Communications of the ACM, February 1997, Volume 40, Number 2, pp. 98-101.
- [19] Schneider, David I., *An Introduction to Programming Using Visual Basic 5.0*, Prentice-Hall, Inc., New Jersey, 1998.
- [20] Seymour, Jem, *The GUI An Interface You Won't Outgrow*, PC Magazine, September 12, 1989, Volume 8, Number 15, pp. 97-109.
- [21] Tenopir, Carol, *The User-System Interface*, Library Journal, August 1989, Volume 114, Number 13, pp. 80-81.
- [22] Valaer, Laura A. and Babb II, Robert G., *Choosing a User Interface Development Tool*, IEEE Software, August 1997, pp. 29-39.

APPENDICES:

APPENDIX A
SOURCE CODE FOR THE DESKTOP TMDL MODEL

MDIForm1 and frmModel

```
Private Sub frmExit_Click()  
    Unload Me  
End Sub  
'  
'
```

```
-----  
Private Sub frmNew_Click()  
    Load frmTmdll  
    frmTmdll.Show  
    frmTmdll.WindowState = 2  
    blnNewfile = True  
    blnChanged = False  
End Sub  
'  
'
```

```
-----  
Private Sub frmOpen_Click()  
    Call OpenFile  
End Sub  
'  
'
```

```
-----  
Private Sub MDIForm_Load()  
    Load frmSplash  
    DoEvents  
' Load the embedded model into memory and set the form invisible.  
    Load frmModel  
    Load frmTmdll  
    frmModel.Hide  
'  
    Unload frmSplash  
    Set frmSplash = Nothing  
'  
    blnNewfile = True  
End Sub  
'  
'
```

```
-----  
Private Sub MDIForm_QueryUnload(Cancel As Integer, UnloadMode As  
Integer)  
'This event will fire before any forms are actually unloaded.  
'Therefore, this is a good place to remind user to save changes  
'before exiting the application.  
    If Not blnasked Then  
        ExitApp Me.ActiveForm  
    End If  
End Sub  
'  
'
```

```
-----  
Private Sub MDIForm_Resize()  
' This routine ensures that the MDI parent form will not  
' be resized smaller than the following specified size.  
'  
    If Me.WindowState = 0 Then  
        If Me.width < 12000 Then Me.width = 12000  
        If Me.height < 10000 Then Me.height = 10000  
    End If  
End Sub
```

```

End If
End Sub
'
'-----
Private Sub midExit_Click()
    Dim Message As String
    Dim ButtonsAndIcons As Integer
    Dim title As String
    Dim Response As Integer

    Message = "Are you sure you want to quit?"
    ButtonsAndIcons = vbYesNo + vbQuestion
    title = "The message Program"
    Response = MsgBox(Message, ButtonsAndIcons, title)
    If Response = vbYes Then
        End
    End If
End Sub
'
'-----frmModel-----
'
Private Sub Form_Load()
    Me.OLE1.CreateEmbed App.Path & "\TMDL.xls"
End Sub
'
'-----
Private Sub frmConst_Click()
    frmTmdl2.Show
    Me.Hide
End Sub
'
'-----
Private Sub frmFacility_Click()
    frmTmdl1.Show
    Me.Hide
End Sub
'
'-----
Private Sub frmWLA_Click()
    frmTmdl3.Show
    Me.Hide
End Sub
'
'-----

```

Form FrmTmdl1

Option Explicit

```
Private Sub Form_Activate()  
  If Me.WindowState <> 2 Then  
    Me.WindowState = 2  
    If frmModel.Visible Then  
      frmModel.Hide  
    End If  
  End If  
  Me.SetFocus  
End Sub
```

```
'-----  
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)  
  If Not blnasked Then  
    Dim canceled As Boolean  
    canceled = CloseForm(Me)  
    Cancel = canceled  
  End If  
End Sub
```

```
'-----  
Private Sub Form_Resize()  
  ' This routine will make sure the form will have enough  
  ' room to hold all command buttons in the title bar.  
  If Me.ScaleWidth > 10000 Then  
    Me.lbltitlebar.width = Me.ScaleWidth - 6285  
    Me.cmdRun.left = Me.lbltitlebar.width + 4320  
    Me.cmdView.left = Me.cmdRun.left + 975  
  Else  
    If Me.WindowState <> 0 Then  
      Me.WindowState = 0  
    End If  
    Me.width = 10000  
    Me.lbltitlebar.width = Me.ScaleWidth - 6285  
    Me.cmdRun.left = Me.lbltitlebar.width + 4320  
    Me.cmdView.left = Me.cmdRun.left + 975  
  End If  
  ' This is to set the minimum height of the form.  
  If Me.height < 8500 Then  
    If Me.WindowState <> 0 Then  
      Me.WindowState = 0  
    End If  
    Me.height = 8500  
  End If  
End Sub
```

```
'-----  
Private Sub frm1Cascade_Click()  
  MDIForm1.Arrange vbCascade  
End Sub
```

```
'-----  
Private Sub frm1Close_Click()  
  CloseForm Me
```

```

End Sub
'-----
Private Sub frmContent_Click()
    SendKeys "{f1}"
End Sub
'-----
Private Sub frmRate_Click()
    frmTmdl2.Show
    Me.Hide
End Sub
'-----
Private Sub frmExit_Click()
    ExitApp Me
End Sub
'-----
Private Sub frmInput_Click()
    updateInput
End Sub
'-----
Private Sub frmModel_Click()
    Dim strMsg As String
    Dim button As Integer
    strMsg = "                Desktop Water Quality Model " & vbCrLf _
        & "                Version 1.0.0" & vbCrLf _
        & vbCrLf _
        & "                Zhengping Yan" & vbCrLf & vbCrLf _
        & "                This application was created for a Master's" & _
        & vbCrLf _
        & "                Degree thesis in Computer Science at OSU. " & _
        & vbCrLf _
        & "                December 18, 1999 "
    button = vbOKOnly
    MsgBox strMsg, button, "About TMDL Model"
End Sub
'-----
Private Sub frmModelview_Click()
    If Me.WindowState <> 0 Then
        Me.WindowState = 0
    '
        On Error Resume Next
        frmModel.Show
        frmModel.OLE1.SetFocus
    Else
        frmModel.Show
        frmModel.OLE1.SetFocus
    End If
End Sub
'-----
Private Sub frmNew_Click()
    If blnNewfile = False And blnChanged = True Then
        Dim strMsg As String
        Dim button As Integer
        Dim intResponse As Integer
        strMsg = "Do you want to save the current Model?"
        button = vbYesNoCancel + vbExclamation
        intResponse = MsgBox(strMsg, button, "Warning")
    End If
End Sub

```

```

    If intResponse = vbYes Then
        SaveData
        DoEvents
        Load frmTmdl1
    ElseIf intResponse = vbNo Then
        Load frmTmdl1
    End If
Else
    Load frmTmdl1
End If
End Sub

```

```

-----
Private Sub frm1Open_Click()
    Call OpenFile
End Sub

```

```

-----
Private Sub frm1Plot_Click()
    frmTmdl4.Visible = True
    DoEvents
    frmTmdl4.VScroll11.Value = 86
    If Not frmTmdl4.Justplot Then
        frmTmdl4.Picture2.Cls
        Figure frmTmdl4.Picture2
        DrawPlot frmTmdl4.Picture2
        frmTmdl4.Justplot = True
    End If
    Me.Hide
End Sub

```

```

-----
Private Sub frm1Print_Click()
    'Send Report to DEFAULT printer.
    Report Printer
    DoEvents
    'Send D.O. plots to default printer
    Figure Printer
    DoEvents
    DrawPlot Printer
End Sub

```

```

-----
Private Sub frm1Printer_Click()
    MDIForm1.CommonDialog1.ShowPrinter
End Sub

```

```

-----
Private Sub frm1Report_Click()
    frmTmdl4.Visible = True
    frmTmdl4.VScroll11.Value = 0
    If Not frmTmdl4.justreport Then
        frmTmdl4.Picture1.Cls
        Report frmTmdl4.Picture1
        frmTmdl4.justreport = True
    End If
    Me.Hide
End Sub

```

```

-----
Private Sub frm1Save_Click()
    If blnNewfile Then
        SaveDataAs MDIForm1.CommonDialog1
    Else
        SaveData
    End If
End Sub
-----

Private Sub frm1SaveAs_Click()
    SaveDataAs MDIForm1.CommonDialog1
End Sub
-----

Private Sub frm1WLA_Click()
    frmTmdl3.Show
    Me.Hide
End Sub
-----

Private Sub pages_Click(Index As Integer)
    Select Case (Index)
        Case 2:
            frmTmdl2.Visible = True
            If frmTmdl2.WindowState = 0 Then
                frmTmdl2.top = 0
                frmTmdl2.left = 0
            End If
            frmTmdl1.Hide

        Case 3:
            frmTmdl3.Visible = True
            If frmTmdl3.WindowState = 0 Then
                frmTmdl3.top = 0
                frmTmdl3.left = 0
            End If
            frmTmdl1.Hide

    End Select
End Sub
-----

Private Sub cmdRun_Click()
    If Not chkcount Then
        MsgBox "An aeration formula has not been selected yet!"
    End If

    Call updateWLA
    Call getDO
    Call SafetyMargin
    Call getSafety

    Me.Hide
    frmTmdl3.Show
End Sub
-----

Private Sub Text1_Change(Index As Integer)
    ' If the values regarding to flow, stream characteristics

```

```

' has been changed, the aeration rate might be changed too.
' This routine makes you select an aeration formula again before
' running the model.
If Index >= 6 And chkcount Then
    Dim I As Integer
    For I = 0 To 3
        If frmTmdl2.Option1(I).Value Then
            frmTmdl2.Option1(I).Value = False
        End If
    Next
    chkcount = False
End If
'
    blnChanged = True
End Sub
'
'-----
Private Sub Text1_GotFocus(Index As Integer)
' The routine will highlight text when the textbox get focus,
' so that user can immediately type information he/she wants
' to input without deleting the existing text first.
'
    Me.Text1(Index).Text = Trim(Me.Text1(Index).Text)
    Me.Text1(Index).SelLength = Len(Me.Text1(Index).Text)
'
' Change text to red when editing.
    Me.Text1(Index).ForeColor = vbRed
End Sub
'
'-----
Private Sub Text1_LostFocus(Index As Integer)
' Change text color back to normal when leaving the textbox.
    Me.Text1(Index).ForeColor = vbBlack
' This routine adjusts the position of text to make the
' user interface attractive.
    Select Case Index
        Case 0 To 6, 8, 9
            Me.Text1(Index).Text = "    " & Me.Text1(Index).Text
        Case 7, 10 To 14
            Me.Text1(Index).Text = "        " & Me.Text1(Index).Text
    End Select
End Sub
'
'-----
Private Sub Text1_KeyDown(Index As Integer, KeyCode As Integer, _
    Shift As Integer)
' It is a common practice to press the "Return" key after finishing
' inputting data in the text box. This routine is to let the text
' box recognize the Return key. When the RETURN key is pressed, the
next text
' will got focus, so the user can start next input.
    If KeyCode = vbKeyReturn And Index < 14 Then
        Me.Text1(Index + 1).SetFocus
    End If
End Sub

```

From FrmTmdl2

```
Private Sub cmdRun_Click()  
    If Not chkcount Then  
        MsgBox "An aeration formula has not been selected yet!"  
    End If  
    '  
    Call updateWLA  
    Call getDO  
    Call SafetyMargin  
    Call getSafety  
    '  
    Me.Hide  
    frmTmdl3.Show  
    '  
    frmTmdl4.Justplot = False  
End Sub  
'  
-----
```

```
Private Sub Form_Activate()  
    If Me.WindowState <> 2 Then  
        Me.WindowState = 2  
        If frmModel.Visible Then  
            frmModel.Hide  
        End If  
    End If  
End Sub  
'  
-----
```

```
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)  
    If Not blnasked Then  
        Dim canceled As Boolean  
        canceled = CloseForm(Me)  
        Cancel = canceled  
    End If  
End Sub  
'  
-----
```

```
Private Sub Form_Resize()  
    ' This routine will make sure the form will have enough  
    ' room to hold all command buttons in the title bar.  
    If Me.ScaleWidth > 10000 Then  
        Me.lbltitlebar.width = Me.ScaleWidth - 6285  
        Me.cmdRun.left = Me.lbltitlebar.width + 4320  
        Me.cmdView.left = Me.cmdRun.left + 975  
    Else  
        If Me.WindowState <> 0 Then  
            Me.WindowState = 0  
        End If  
        Me.width = 10000  
        Me.lbltitlebar.width = Me.ScaleWidth - 6285  
        Me.cmdRun.left = Me.lbltitlebar.width + 4320  
        Me.cmdView.left = Me.cmdRun.left + 975  
    End If  
    ' This is to set the minimum height of the form.  
    If Me.height < 8500 Then  
        If Me.WindowState <> 0 Then
```



```

        Me.WindowState = 0
    End If
    Me.height = 8500
End If
End Sub
'-----
Private Sub frm2Cascade_Click()
    MDIForm1.Arrange vbCascade
End Sub
'-----
Private Sub frm2Close_Click()
    Call CloseForm(Me)
End Sub
'-----
Private Sub frm2Content_Click()
    SendKeys "{F1}"
End Sub
'-----
Private Sub frm2Exit_Click()
    ExitApp Me
End Sub
'-----
Private Sub frm2Input_Click()
    updateInput
End Sub
'-----
Private Sub frm2Model_Click()
    If Me.WindowState <> 0 Then
        Me.WindowState = 0
        frmModel.Show
        frmModel.OLE1.SetFocus
    Else
        frmModel.Show
        frmModel.OLE1.SetFocus
    End If
End Sub
'-----
Private Sub frm2New_Click()
    If blnNewfile = False And blnChanged = True Then
        Dim strMsg As String
        Dim button As Integer
        Dim intResponse As Integer
        strMsg = "Do you want to save the current model?"
        button = vbYesNoCancel + vbExclamation
        intResponse = MsgBox(strMsg, button, "Warning")
        '
        If intResponse = vbYes Then
            SaveData
            DoEvents
            Load frmTmdl1
        ElseIf intResponse = vbNo Then
            Load frmTmdl1
        End If
    Else
        Load frmTmdl1
    End If
End Sub
End Sub

```

```

-----
Private Sub frm2Open_Click()
  RetrieveData MDIForm1.CommonDialog1
End Sub
-----
Private Sub frm2Plot_Click()
  frmTmdl4.Visible = True
  DoEvents
  frmTmdl4.VScroll1.Value = 86
  If Not frmTmdl4.Justplot Then
    frmTmdl4.Picture2.Cls
    Figure frmTmdl4.Picture2
    DrawPlot frmTmdl4.Picture2
    frmTmdl4.Justplot = True
  End If
  Me.Hide
End Sub
-----
Private Sub frm2Print_Click()
  'Send Report to DEFAULT printer
  Report Printer
  DoEvents
  'Send D.O. plots to the default printer.
  Figure Printer
  DoEvents
  DrawPlot Printer
End Sub
-----
Private Sub frm2Printer_Click()
  MDIForm1.CommonDialog1.ShowPrinter
End Sub
-----
Private Sub frm2Report_Click()
  frmTmdl4.Visible = True
  frmTmdl4.VScroll1.Value = 0
  If Not frmTmdl4.justreport Then
    frmTmdl4.Picture1.Cls
    Report frmTmdl4.Picture1
    frmTmdl4.justreport = True
  End If
  Me.Hide
End Sub
-----
Private Sub frm2Runmodel_Click()
  If Not chkcount Then
    MsgBox "An aeration formula has not been selected yet!"
  End If
  '
  Call updateWLA
  DoEvents
  Call getDO
  DoEvents
  Call SafetyMargin
  DoEvents
  Call getSafety
  '
  frmTmdl4.Justplot = False

```

```

    frmTmdl4.justreport = False
End Sub
'-----
Private Sub frm2Save_Click()
    If blnNewfile Then
        SaveDataAs MDIForm1.CommonDialog1
    Else
        SaveData
    End If
End Sub
'-----
Private Sub frm2SaveAs_Click()
    SaveDataAs MDIForm1.CommonDialog1
End Sub
'-----
Private Sub frm2Stream_Click()
    frmTmdl1.Show
    Me.Hide
End Sub
'-----
Private Sub frm2WLA_Click()
    frmTmdl3.Show
    Me.Hide
End Sub
'-----
Private Sub Option1_GotFocus(Index As Integer)
    Dim obj As Object
    Set obj = frmModel.OLE1.object.sheets("Data")
    '
    obj.range("F57").Value = Index + 1
    ' If it is the first time to selected the aeration formula,
    ' get the values for each formula from the embedded Excel
    ' model and set chkcount nonzero. Otherwise skip the
    ' following part of the code.
    '
    If Not chkcount Then
        Call updateInput
        ' Return aeration rate by Tsivoglu's formula.
        Me.txtk2(1).Text = obj.range("F46").Text
        Me.txtk2(0).Text = obj.range("H46").Text
        '
        ' Return aeration rate by the Turney-Harris formula.
        '
        Me.txtk2(3).Text = obj.range("F49").Text
        Me.txtk2(2).Text = obj.range("H49").Text
        '
        ' Return aeration rate by the Texas formula.
        '
        Me.txtk2(5).Text = obj.range("F52").Text
        Me.txtk2(4).Text = obj.range("H52").Text
        '
        ' Return aeration rate by the Owens et al. Formula.
        '
        Me.txtk2(7).Text = obj.range("F55").Text
        Me.txtk2(6).Text = obj.range("H55").Text
        '
        chkcount = True
    End If
End Sub

```

```

End If
' Make the selected aeration rate boldface.
For I = 0 To 7
    Me.txtk2(I).Font.Bold = False
Next
I = Index + 2
Me.txtk2(I).Font.Bold = True
Me.txtk2(I + 1).Font.Bold = True
'
Set obj = Nothing
End Sub
'-----

```

```

Private Sub pages_Click(Index As Integer)
    Select Case (Index)
        Case 0:
            frmTmdl1.Visible = True
            If frmTmdl1.WindowState = 0 Then
                frmTmdl1.top = 0
                frmTmdl1.left = 0
            End If
            frmTmdl2.Hide
        Case 1:
            frmTmdl1.Visible = True
            If frmTmdl1.WindowState = 0 Then
                frmTmdl1.top = 0
                frmTmdl1.left = 0
            End If
            frmTmdl2.Hide
        Case 2:
            frmTmdl3.Visible = True
            If frmTmdl3.WindowState = 0 Then
                frmTmdl3.top = 0
                frmTmdl3.left = 0
            End If
            frmTmdl2.Hide
        Case 3:
            frmTmdl3.Visible = True
            If frmTmdl3.WindowState = 0 Then
                frmTmdl3.top = 0
                frmTmdl3.left = 0
            End If
            frmTmdl2.Hide
    End Select
End Sub
'-----

```

```

Private Sub txtPara_Change(Index As Integer)
    Dim I As Integer
    For I = 0 To 3
        If Me.Option1(I).Value = True Then
            Me.Option1(I).Value = False
        End If
    Next
    chkcount = False
    blnChanged = True
End Sub
'-----

```

```

Private Sub txtPara_GotFocus(Index As Integer)
    Me.txtPara(Index).Text = Trim(Me.txtPara(Index).Text)
    'Select the whole text and change color to red.
    Me.txtPara(Index).SelLength = Len(Me.txtPara(Index).Text)
    Me.txtPara(Index).ForeColor = vbRed
End Sub
'
'-----
Private Sub txtPara_KeyDown(Index As Integer, KeyCode As Integer, _
    Shift As Integer)
    ' It is a common practice to press the "Return" key after finishing
    ' inputing data in the text box. This routine is to let the text
    ' box recognize the Return key. When the RETURN key is pressed, the
    ' next text will get focus, so the user can start the next input.
    If KeyCode = vbKeyReturn And Index < 9 Then
        Me.txtPara(Index + 1).SetFocus
    End If
End Sub
'
'-----
Private Sub txtPara_LostFocus(Index As Integer)
    Me.txtPara(Index).Text = "          " & Me.txtPara(Index).Text
    Me.txtPara(Index).ForeColor = vbBlack
End Sub

```

Form frmTmdl3

Option Explicit

```
'-----  
Private Sub cmdRun_Click()  
    If Not chkcount Then  
        Dim strMsg As String  
        Dim buttons As Integer  
        strMsg = "An aeration formula has not been selected yet!"  
        buttons = vbOKOnly + vbExclamation  
        MsgBox strMsg, buttons, "Reminder"  
        Exit Sub  
    End If  
  
    Dim I As Integer  
    For I = 21 To 41  
        frmTmdl3.txtwla(I).Text = ""  
    Next I  
  
    Call updateWLA  
    DoEvents  
    Call getDO  
    DoEvents  
    Call SafetyMargin  
    DoEvents  
    Call getSafety  
  
    frmTmdl4.Justplot = False  
    frmTmdl4.justreport = False  
End Sub
```

```
'-----  
Private Sub cmdView_Click()  
    frmTmdl4.Visible = True  
    DoEvents  
    frmTmdl4.VScroll11.Value = 86  
    If Not frmTmdl4.Justplot Then  
        frmTmdl4.Picture2.Cls  
        Figure frmTmdl4.Picture2  
        DrawPlot frmTmdl4.Picture2  
        frmTmdl4.Justplot = True  
    End If  
    frmTmdl3.Hide  
End Sub
```

```
'-----  
Private Sub Form_Activate()  
    If Me.WindowState <> 2 Then  
        Me.WindowState = 2  
        If frmModel.Visible Then  
            frmModel.Hide  
        End If  
    End If  
End Sub
```

```

-----
Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    If Not blnasked Then
        Dim canceled As Boolean
        canceled = CloseForm(Me)
        Cancel = canceled
    End If
End Sub
'
-----

Private Sub Form_Resize()
    If Me.ScaleWidth > 10000 Then
        Me.lbltitlebar.width = Me.ScaleWidth - 6285
        Me.cmdRun.left = Me.lbltitlebar.width + 4320
        Me.cmdView.left = Me.cmdRun.left + 975
    Else
        If Me.WindowState <> 0 Then
            Me.WindowState = 0
        End If
        Me.width = 10000
        Me.lbltitlebar.width = Me.ScaleWidth - 6285
        Me.cmdRun.left = Me.lbltitlebar.width + 4320
        Me.cmdView.left = Me.cmdRun.left + 975
    End If
    ' This is to set the minimum height of the form.
    If Me.height < 8500 Then
        If Me.WindowState <> 0 Then
            Me.WindowState = 0
        End If
        Me.height = 8500
    End If
End Sub
'
-----

Private Sub frm3Cascade_Click()
    MDIForm1.Arrange vbCascade
End Sub
'
-----

Private Sub frm3Const_Click()
    frmTmdl2.Show
    Me.Hide
End Sub
'
-----

Private Sub frm3Content_Click()
    SendKeys "{F1}"
End Sub
'
-----

Private Sub frm3Input_Click()
    updateInput
End Sub
'
-----

Private Sub frm3Model_Click()
    If Me.WindowState <> 0 Then

```

```

    Me.WindowState = 0
    frmModel.Show
    frmModel.OLE1.SetFocus
Else
    frmModel.Show
    frmModel.OLE1.SetFocus
End If
End Sub
'
'-----
Private Sub frm3New_Click()
    If blnNewfile = False And blnChanged = True Then
        Dim strMsg As String
        Dim button As Integer
        Dim intResponse As Integer
        strMsg = "Do you want to save the current Model?"
        button = vbYesNoCancel + vbExclamation
        intResponse = MsgBox(strMsg, button, "Warning")
        '
        If intResponse = vbYes Then
            SaveData
            DoEvents
            Load frmTmdl1
        ElseIf intResponse = vbNo Then
            Load frmTmdl1
        End If
    Else
        Load frmTmdl1
    End If
End Sub
'
'-----
Private Sub frm3Plot_Click()
    frmTmdl4.Visible = True
    DoEvents
    frmTmdl4.VScroll1.Value = 86
    If Not frmTmdl4.Justplot Then
        frmTmdl4.Picture2.Cls
        Figure frmTmdl4.Picture2
        DrawPlot frmTmdl4.Picture2
        frmTmdl4.Justplot = True
    End If
    Me.Hide
End Sub
'
'-----
Private Sub frm3Print_Click()
    'Send Report to the DEFAULT printer.
    Report Printer
    DoEvents
    'Send D.O. plots to the default printer.
    Figure Printer
    DoEvents
    DrawPlot Printer
End Sub
'
'-----

```



```

Private Sub frm3Printer_Click()
    MDIForm1.CommonDialog1.ShowPrinter
End Sub
'
'-----
Private Sub frm3Report_Click()
    frmTmdl4.Visible = True
    frmTmdl4.VScroll1.Value = 0
    If Not frmTmdl4.justreport Then
        frmTmdl4.Picture1.Cls
        Report frmTmdl4.Picture1
        frmTmdl4.justreport = True
    End If
    Me.Hide
End Sub
'
'-----
Private Sub frm3Close_Click()
    Call CloseForm(Me)
End Sub
'
'-----
Private Sub frm3Exit_Click()
    ExitApp Me
End Sub
'
'-----
Private Sub frm3Open_Click()
    RetrieveData MDIForm1.CommonDialog1
End Sub
'
'-----
Private Sub frm3Runmodel_Click()
    If Not chkcount Then
        MsgBox "An aeration formula has not been selected yet!"
    End If
    '
    Dim I As Integer
    For I = 21 To 41
        frmTmdl3.txtwla(I).Text = ""
    Next I
    '
    Call updateWLA
    DoEvents
    Call getDO
    DoEvents
    Call SafetyMargin
    DoEvents
    Call getSafety
    '
    frmTmdl4.Justplot = False
    frmTmdl4.justreport = False
End Sub
'
'-----
Private Sub frm3Save_Click()
    If blnNewfile Then

```

```

        SaveDataAs MDIForm1.CommonDialog1
    Else
        SaveData
    End If
End Sub
'
'-----
Private Sub frm3SaveAs_Click()
    SaveDataAs MDIForm1.CommonDialog1
End Sub
'
'-----
Private Sub frm3Stream_Click()
    frmTmdl1.Show
    Me.Hide
End Sub
'
'-----
Private Sub pages_Click(Index As Integer)
    Select Case (Index)
        Case 0:
            frmTmdl1.Visible = True
            If frmTmdl1.WindowState = 0 Then
                frmTmdl1.top = 0
                frmTmdl1.left = 0
            End If
            frmTmdl3.Hide
        Case 1:
            frmTmdl2.Visible = True
            If frmTmdl2.WindowState = 0 Then
                frmTmdl2.top = 0
                frmTmdl2.left = 0
            End If
            frmTmdl3.Hide
    End Select
End Sub
'
'-----
Private Sub txtwla_Change(Index As Integer)
    blnChanged = True
End Sub
'
'-----
Private Sub txtwla_GotFocus(Index As Integer)
    Dim intSpacePos As Integer
    Dim intLength As Integer
    If Index < 9 Then
        intSpacePos = InStr(Me.txtwla(Index), " ")
        Do While intSpacePos > 0 And intSpacePos < 2
            intLength = Len(Me.txtwla(Index)) - 1
            Me.txtwla(Index) = Right(Me.txtwla(Index), intLength)
            intSpacePos = InStr(Me.txtwla(Index), " ")
        Loop
    End If
    Me.txtwla(Index).SelLength = Len(Me.txtwla(Index).Text)
    Me.txtwla(Index).ForeColor = vbRed
End Sub

```

```

'-----
Private Sub txtwla_KeyDown(Index As Integer, KeyCode As Integer, _
    Shift As Integer)
    If KeyCode = vbKeyReturn And Index < 8 Then
        Me.txtwla(Index + 1).SetFocus
    End If
End Sub
'-----

Private Sub txtwla_LostFocus(Index As Integer)
    Me.txtwla(Index).ForeColor = vbBlack
    '
    Dim intSpacePos As Integer
    Dim intLength As Integer

    If Index < 9 Then
        intSpacePos = InStr(Me.txtwla(Index), " ")
        Do While intSpacePos > 0 And intSpacePos < 2
            intLength = Len(Me.txtwla(Index)) - 1
            Me.txtwla(Index) = Right(Me.txtwla(Index), intLength)
            intSpacePos = InStr(Me.txtwla(Index), " ")
        Loop
    '
        Me.txtwla(Index).Text = "          " & Me.txtwla(Index).Text
    End If
End Sub

```



```

'
'-----
Private Sub frm4Const_Click()
    frmTmdl2.Show
    Me.Hide
End Sub
'
'-----
Private Sub frm4Content_Click()
    SendKeys "{F1}"
End Sub
'
'-----
Private Sub frm4Exit_Click()
    ExitApp Me
End Sub
'
'-----
Private Sub frm4Model_Click()
    If Me.WindowState <> 0 Then
        Me.WindowState = 0
        frmModel.Show
        frmModel.OLE1.SetFocus
    Else
        frmModel.Show
        frmModel.OLE1.SetFocus
    End If
End Sub
'
'-----
Private Sub frm4Print_Click()
    'Send Report to DEFAULT printer
    Report Printer
    DoEvents
    'Send D.O. plots to default printer
    Figure Printer
    DoEvents
    DrawPlot Printer
End Sub
'
'-----
Private Sub frm4Printer_Click()
    MDIForm1.CommonDialog1.ShowPrinter
End Sub
'
'-----
Private Sub frm4Save_Click()
    If blnNewfile Then
        SaveDataAs MDIForm1.CommonDialog1
    Else
        SaveData
    End If
End Sub
'
'-----
Private Sub frm4SaveAs_Click()
    SaveDataAs MDIForm1.CommonDialog1

```

```

End Sub
'
'-----
Private Sub frm4Stream_Click()
    frmTmdl1.Show
    Me.Hide
End Sub
'
'-----
Private Sub frm4WLA_Click()
    frmTmdl3.Show
    Me.Hide
End Sub
'
'-----
Private Sub HScroll1_Change()
    Dim intLeft As Integer
    Dim intRight As Integer
    Static intHvalue As Integer
    Dim intChange As Integer
    '
    intLeft = Me.Picture1.left
    intRight = intLeft + Me.Picture1.width
    intChange = intHvalue - Me.HScroll1.Value
    '
    Select Case Abs(intChange)
        Case Is > 10:
            If intChange > 0 Then
                Me.Picture1.left = 100
                Me.Picture2.left = 100
            Else
                intLeft = Me.ScaleWidth - Me.Picture1.width - 400
                Me.Picture1.left = intLeft
                Me.Picture2.left = intLeft
            End If
        Case Is <= 10:
            If intChange < 0 Then
                If intRight > Me.ScaleWidth - 400 Then
                    Me.Picture1.left = intLeft + intChange * 150
                    Me.Picture2.left = intLeft + intChange * 150
                End If
            Else
                If Me.Picture1.left < 100 Then
                    Me.Picture1.left = intLeft + intChange * 150
                    Me.Picture2.left = intLeft + intChange * 150
                End If
            End If
    End Select
    '
    intHvalue = Me.HScroll1.Value
End Sub
'
'-----
Private Sub Picture1_KeyDown(KeyCode As Integer, Shift As Integer)
    Select Case KeyCode
        Case vbKeyPageDown:

```

```

    If Me.VScroll11.Value <= 83 Then
        'must ensure the sum is less than 100
        Me.VScroll11.Value = Me.VScroll11.Value + 17
    Else
        Me.VScroll11.Value = 100
    End If
Case vbKeyPageUp:
    If Me.VScroll11.Value >= 17 Then
        'must ensure value after subtraction is >= 0
        Me.VScroll11.Value = Me.VScroll11.Value - 17
    Else
        Me.VScroll11.Value = 0
    End If
Case vbKeyUp:
    If Me.VScroll11.Value > 0 Then
        Me.VScroll11.Value = Me.VScroll11.Value - 1
    End If
Case vbKeyDown:
    If Me.VScroll11.Value < 100 Then
        Me.VScroll11.Value = Me.VScroll11.Value + 1
    End If
End Select
End Sub
'
'-----
Private Sub Picture2_KeyDown(KeyCode As Integer, Shift As Integer)
    Select Case KeyCode
        Case vbKeyPageDown:
            If Me.VScroll11.Value <= 83 Then
                'the max of value is defined as 100, if starting value is
                'greater than 83, the sum will be over 100 --> runtime error
                Me.VScroll11.Value = Me.VScroll11.Value + 17
            Else
                Me.VScroll11.Value = 100
            End If
        Case vbKeyPageUp:
            If Me.VScroll11.Value >= 17 Then
                'for the same reason, we must ensure the value never go
                'down below zero. otherwise, cause a runtime error
                Me.VScroll11.Value = Me.VScroll11.Value - 17
            Else
                Me.VScroll11.Value = 0
            End If
        Case vbKeyUp:
            If Me.VScroll11.Value > 0 Then
                Me.VScroll11.Value = Me.VScroll11.Value - 1
            End If
        Case vbKeyDown:
            If Me.VScroll11.Value < 100 Then
                Me.VScroll11.Value = Me.VScroll11.Value + 1
            End If
    End Select
End Sub
'
'-----
Private Sub Toolbar1_ButtonClick(ByVal button As MSComctlLib.button)
    Select Case button.Index

```

Standard Module

```
Case 1:
    frmTmdl1.Visible = True
    Me.Hide
Case 2:
    frmTmdl3.Visible = True
    Me.Hide
Case 3:
    Me.VScroll1.Value = 86
    If Not Justplot Then
        Me.Picture2.Cls
        Figure Me.Picture2
        DoEvents
        DrawPlot Me.Picture2
        Justplot = True
    End If
Case 4:
    Me.VScroll1.Value = 0
    If Not justreport Then
        Report Me.Picture1
        justreport = True
    End If
Case 5:
    'Send Report to DEFAULT printer
    Report Printer
    DoEvents
    'Send D.O. plots to default printer
    Figure Printer
    DoEvents
    DrawPlot Printer

End Select
End Sub
'
'-----
Private Sub VScroll1_Change()
    Dim lngTop As Long
    Dim lngTop1 As Long
    Static intValue As Integer
    Dim intChange As Long

    lngTop = Me.Picture1.top
    lngTop1 = Me.Picture2.top
    intChange = intValue - Me.VScroll1.Value
    '
    Me.Picture1.top = lngTop + intChange * 925
    Me.Picture2.top = lngTop1 + intChange * 925

    intValue = Me.VScroll1.Value
End Sub
'
```


Standard Module

```
Option Explicit
'
' Define Form level variables.
' These variables will be used to open, close, save or exit a file.
'
Public blnNewfile As Boolean      'new file
'
Public CurrentName As String      'name of opened file
'
' True if a text being edited has changed
Public blnChanged As Boolean
'
Public blnasked As Boolean        'used before exit
'
' This is to check if aeration rates in form frmTmdl2 needs to be
' updated.
Public chkcount As Boolean
'
'-----
Sub checkInput()
    Dim I As Integer, J As Integer
    Dim title As String
    Dim msg As String
    Dim flag As Integer
    Dim reply

    Dim obj As Object
    Set obj = frmModel.OLE1.object.sheets("Data")

    ' This routine will first test if all required data input
    ' are provided by the user.  If all data are inputted, the
    ' routine will display the modeling results.  Otherwise,
    ' the program halts, gives a warning message and returns
    ' to the input data page.
    '
    ' Step 1: test if all entries in the Modeler section have inputs.
    For I = 0 To 2
        If Len(frmTmdl1.Text1(I)) = 0 Then
            title = "Insufficient Input"
            msg = "No data was inputted in one of the entries in the" _
                & vbCrLf & "Modeler section. Please fill out all " & _
                "entries then try again."
            flag = vbOKOnly + vbExclamation
            reply = MsgBox(msg, flag, title)
        End If
    Next

    ' Step 2: test if all entries in the Facility & Dischrge section
    ' have inputs.
    For I = 3 To 8
        If Len(frmTmdl1.Text1(I)) = 0 Then
            title = "Insufficient Input"
            msg = "No data was inputted in one of the entries in the" _
```

```

        & vbCrLf & "Facility & Discharge section. Please " & _
        "fill out" & vbCrLf & "all entries then try again."
    flag = vbOKOnly + vbExclamation
    reply = MsgBox(msg, flag, title)
End If
Next

' step 3: test if all entries in the Stream section
'         have inputs.
For I = 9 To 14
    If Len(frmTmdl1.Text1(I)) = 0 Then
        title = "Insufficient Input"
        msg = "No data was inputed in one of the entries in the" _
            & vbCrLf & "Receiving Stream section. Please " & _
            "fill out all entries" & vbCrLf & "then try again."
        flag = vbOKOnly + vbExclamation
        reply = MsgBox(msg, flag, title)
    End If
Next

' Step 4: test if all entries in the Rate Constants section
'         have inputs.
For I = 0 To 9
    If Len(frmTmdl2.txtPara(I)) = 0 Then
        title = "Insufficient Input"
        msg = "No data was inputed in one of the entries in the" _
            & vbCrLf & "Rate Constants at 20° C section. Please " & _
            "fill out all entries" & vbCrLf & "then try again."
        flag = vbOKOnly + vbExclamation
        reply = MsgBox(msg, flag, title)
    End If
Next
Set obj = Nothing
End Sub

'-----
Sub updateInput()
    Dim I As Integer
    Dim obj As Object
    Set obj = frmModel.OLE1.object.sheets("Data")

    'Update model with information on Form #1.
    With frmTmdl1
        obj.range("E4").Value = .Text1(0).Text           '- Engineer
        obj.range("E3").Value = .Text1(3).Text           '- Facility name
        obj.range("E5").Value = Val(.Text1(2).Text)      '- Date performed
        obj.range("C9").Value = .Text1(4).Text           '- Location
        obj.range("C10").Value = .Text1(5).Text          '- County
        obj.range("H9").Value = Val(.Text1(6).Text)      '- Segment #
        obj.range("D12").Value = Val(.Text1(7).Text)    '- Design flow
        obj.range("D13").Value = .Text1(9).Text          '- Strea name
        obj.range("F15").Value = Val(.Text1(10).Text)    '- 7Q2
        obj.range("H14").Value = Val(.Text1(11).Text)   '- Stream length
        obj.range("F17").Value = Val(.Text1(12).Text)   '- Stream slope
        obj.range("F18").Value = Val(.Text1(13).Text)   '- Side slope
        obj.range("F19").Value = Val(.Text1(14).Text)   '- Mannings' #
    End With
End Sub

```

```

'
  For I = 0 To 2      ' Water Quality Standards
    If .Option1(I).Value Then
      obj.range("E30").Value = I + 1
    End If
  Next
'
End With
'
'Update embedded model with data from Form #2.
With frmTmdl2
  obj.range("F43").Value = Val(.txtPara(0).Text)  '- K1
  '
  obj.range("H65").Value = Val(.txtPara(1).Text)  '- Ks, summer
  obj.range("G65").Value = Val(.txtPara(2).Text)  '- Ks, spring
  obj.range("F65").Value = Val(.txtPara(3).Text)  '- Ks, winter
  '
  obj.range("H61").Value = Val(.txtPara(4).Text)  '- SOD, summer
  obj.range("G61").Value = Val(.txtPara(5).Text)  '- SOD, spring
  obj.range("F61").Value = Val(.txtPara(6).Text)  '- SOD, winter
  '
  obj.range("H63").Value = Val(.txtPara(7).Text)  '- KN, summer
  obj.range("G63").Value = Val(.txtPara(8).Text)  '- KN, spring
  obj.range("F63").Value = Val(.txtPara(9).Text)  '- KN, winter
  '
End With
Set obj = Nothing
End Sub

```

```

'-----
'
Sub updateWLA()
  Dim obj As Object
  Set obj = frmModel.OLE1.object.sheets("Data")
  '
  With frmTmdl3
    ' Update CBOD5
    obj.range("B72").Value = Val(.txtwla(0).Text) ' summer
    obj.range("B73").Value = Val(.txtwla(1).Text) ' spring
    obj.range("B74").Value = Val(.txtwla(2).Text) ' winter
    ' update NH3-N
    obj.range("C72").Value = Val(.txtwla(3).Text) ' summer
    obj.range("C73").Value = Val(.txtwla(4).Text) ' spring
    obj.range("C74").Value = Val(.txtwla(5).Text) ' winter
    ' update D.O.
    obj.range("D72").Value = Val(.txtwla(6).Text) ' summer
    obj.range("D73").Value = Val(.txtwla(7).Text) ' spring
    obj.range("D74").Value = Val(.txtwla(8).Text) ' winter
    '
  End With
  '
  Set obj = Nothing
  '
End Sub
'-----
'

```

```

Sub getDO()
''
Dim obj As Object
Set obj = frmModel.OLE1.object.sheets("Data")
,
With frmTmdl3
' Minimum D.O. under 0.0 cfs condition
,
.txtwla(9).Text = obj.range("C109").Text ' summer
.txtwla(10).Text = obj.range("C110").Text ' spring
.txtwla(11).Text = obj.range("C111").Text ' winter
' D.O. sags under 0.0 cfs condition
.txtwla(12).Text = obj.range("E109").Text ' summer
.txtwla(13).Text = obj.range("E110").Text ' spring
.txtwla(14).Text = obj.range("E111").Text ' winter
,
' Minimum D.O. under 1.0 cfs condition
,
.txtwla(15).Text = obj.range("F109").Text ' summer
.txtwla(16).Text = obj.range("F110").Text ' spring
.txtwla(17).Text = obj.range("F111").Text ' winter
' D.O. sags under 1.0 cfs condition
.txtwla(18).Text = obj.range("H109").Text ' summer
.txtwla(19).Text = obj.range("H110").Text ' spring
.txtwla(20).Text = obj.range("H111").Text ' winter
,
End With
,
Set obj = Nothing
,
End Sub
'-----
Sub getSafety()
''
Dim obj As Object
Set obj = frmModel.OLE1.object.sheets("Data")
,
With frmTmdl3
' Get safety factor for the Summer.
,
.txtwla(21).Text = obj.range("B96").Text ' CBOD max
.txtwla(22).Text = obj.range("C96").Text ' NH3-N max
.txtwla(23).Text = obj.range("D96").Text ' CBOD alloc
.txtwla(24).Text = obj.range("E96").Text ' NH3-N alloc
.txtwla(25).Text = obj.range("F96").Text ' CBOD resved
.txtwla(26).Text = obj.range("G96").Text ' NH3-N resved
.txtwla(27).Text = obj.range("H96").Text ' Safety
,
' Get safety factor for the spring.
,
.txtwla(28).Text = obj.range("B97").Text ' CBOD max
.txtwla(29).Text = obj.range("C97").Text ' NH3-N max
.txtwla(30).Text = obj.range("D97").Text ' CBOD alloc
.txtwla(31).Text = obj.range("E97").Text ' NH3-N alloc
.txtwla(32).Text = obj.range("F97").Text ' CBOD resved
.txtwla(33).Text = obj.range("G97").Text ' NH3-N resved
,

```

```

.txtwla(34).Text = obj.range("H97").Text ' Safety
'
' Get safety factor for the winter.
'
.txtwla(35).Text = obj.range("B98").Text ' CBOD max
.txtwla(36).Text = obj.range("C98").Text ' NH3-N max
.txtwla(37).Text = obj.range("D98").Text ' CBOD alloc
.txtwla(38).Text = obj.range("E98").Text ' NH3-N alloc
.txtwla(39).Text = obj.range("F98").Text ' CBOD resved
.txtwla(40).Text = obj.range("G98").Text ' NH3-N resved
.txtwla(41).Text = obj.range("H98").Text ' Safety
'
End With
Set obj = Nothing
End Sub
'
'-----
' This general procedure will generate a nicely oragnized report of the
' TMDL model based upon the information from each input form and
' the results of the embedded model.
'
Public Sub Report(prtObj As Object)
Dim intPos As Integer, intNum As Integer
Dim intMargin As Integer
Dim intWidth As Integer
Dim N As Long
Dim y As Long
Dim strText As String
Dim strWord As String

intMargin = 1440
With prtObj
' If Not prtObj Is Printer Then .Cls
prtObj.Font.Bold = False
.Font.Size = 10
If Not prtObj Is Printer Then ' Show page separator on the Form.
.Cls
For N = 0 To 3
y = 15840 * (N + 1) - 50
prtObj.Line (0, y)-(12240, y + 100), 8, B
Next N

.CurrentX = intMargin
.CurrentY = 720
prtObj.Print "Page 1 of 6"
Else ' No page separator on Printer.
.CurrentX = intMargin
.CurrentY = 720
prtObj.Print "Page "; Str(1); " of 6"
End If
'
' Print Title
.Font.Name = "New Times Roman"
.Font.Size = 16
.Font.Bold = True
strText = "Desktop TMDL Model"
.CurrentX = (12240 - .TextWidth(strText)) / 2

```

CHAPTER IV

HELP FOR THE DESKTOP TMDL MODEL

There are two types of help files in Microsoft window applications: Winhelp and HTML help. Winhelp is used in many Microsoft applications, such as Visual Basic 5, Microsoft Word, Excel etc. in Office 97. However, with the increasing popularity of the Internet, Microsoft has changed its applications' help to an HTML help system. The HTML help system has now become the new Microsoft standard [18]. For example, Visual Basic 6 uses an HTML help system.

A help file for your Visual Basic application needs to be created with a separate help workshop. An HTML help workshop is a part of the standard package in Visual Basic 6.0 (Professional Edition or Enterprise Edition). For an earlier version of VB, for example VB 5.0, one could download an HTML Help Workshop from Microsoft's web site [21]. The help file for the Desktop TMDL Model was created using this workshop.

Before creating an HTML help file for a Visual Basic application, one has to create a standard HTML file (with the extension of "html") for each help topics he/she plans to provide in the application's help system. Figure 11 is a simple example of such standard HTML language, which is also one of the help topics for the Desktop TMDL model.

When creating an HTML help file, another very important issue is having a unique Help Context ID for each help topic. The Help Context ID can be any number, but must be unique. Based on the contents, help topics are often classified for different levels.

```

Do
    intPos = InStr(strText, " ")
    strWord = left(strText, intPos)
    strText = Mid(strText, intPos + 1)
    intWidth = .CurrentX + .TextWidth(strWord)
    N = N + 1
    If intWidth > 12240 - intMargin Then
        prtObj.Print
        prtObj.Print Tab(52);
    End If
    prtObj.Print strWord;
    Loop While (N < 20000 And intPos <> 0)
Else
    ' The name fits on one line.
    prtObj.Print strText
End If
'
.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "Stream Segment # :"; Tab(52);
prtObj.Print Trim(frmTmdl1.Text1(6).Text)
'
.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "Stream Length (miles) :"; Tab(52);
prtObj.Print Trim(frmTmdl1.Text1(11).Text)
'
.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "Stream Slope (ft/mile) :"; Tab(52);
prtObj.Print Trim(frmTmdl1.Text1(12).Text)
'
.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "Low Critical Flow, 7Q2 (MGD) :";
prtObj.Print Tab(52); Trim(frmTmdl1.Text1(10).Text)
'
.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "Manning's Number :"; Tab(52);
prtObj.Print Trim(frmTmdl1.Text1(14).Text)
'
'Print Water Quality Standards in Stream.
.Font.Size = 12
.CurrentX = intMargin
.CurrentY = .CurrentY + 400
prtObj.Print "III. WATER QUALITY CRITERIA OF RECEIVING STREAM"
'
Dim Excelobj As Object
Set Excelobj = frmModel.OLE1.object.sheets("Data")
'
.CurrentY = .CurrentY + 200
.Font.Size = 11
prtObj.Print Tab(35); "(";
prtObj.Print Excelobj.range("B29").Text;
prtObj.Print ")"
'
.CurrentY = .CurrentY + 60
prtObj.Print Tab(40); "Summer : "; Tab(55);
prtObj.Print Excelobj.range("G30").Text; " mg/l"
'
.CurrentY = .CurrentY + 60
prtObj.Print Tab(40); "Spring : "; Tab(55);

```

```

prtObj.Print Excelobj.range("G31").Text; " mg/l"
,
.CurrentY = .CurrentY + 60
prtObj.Print Tab(40); "Winter : "; Tab(55);
prtObj.Print Excelobj.range("G32").Text; " mg/l"
,
'Print Coefficiennts at 20 degree C.
.Font.Size = 12
.CurrentX = intMargin
.CurrentY = .CurrentY + 400
prtObj.Print "IV. RATE CONSTANTS AT 20 DEGREE C"
,
.CurrentY = .CurrentY + 200
.Font.Size = 11
prtObj.Print Tab(20); "CBOD Decay Rate, K1 (/day) :"; Tab(60);
prtObj.Print Trim(frmTmdl2.txtPara(0).Text)
,
.CurrentY = .CurrentY + 60
If Excelobj.range("F15").Value = 0 Then
prtObj.Print Tab(20); "Rearation Rate, K2 (/day) :"; Tab(60);
prtObj.Print Excelobj.range("H58").Text;
prtObj.Print Tab(68); "-- under 0.0 cfs"
,
.CurrentY = .CurrentY + 60
prtObj.Print Tab(60); Excelobj.range("F58").Text;
prtObj.Print Tab(68); "-- under 1.0 cfs"
Else
prtObj.Print Tab(20); "Rearation Rate, K2 (/day) :"; Tab(60);
prtObj.Print Excelobj.range("F58").Text; " -- under 7Q2 flow"
End If
,
y = .CurrentY + 100
.CurrentY = .CurrentY + 200
prtObj.Print Tab(64); "Summer"; Tab(81); "Spring";
prtObj.Print Tab(96); "Winter"
,
.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "CBOD Settling Rate, KS (/day) :";
prtObj.Print Tab(67); Trim(frmTmdl2.txtPara(4).Text);
prtObj.Print Tab(82); Trim(frmTmdl2.txtPara(5).Text);
prtObj.Print Tab(97); Trim(frmTmdl2.txtPara(6).Text);
,
.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "NBOD Decay Rate, KN (/day) :";
prtObj.Print Tab(67); Trim(frmTmdl2.txtPara(7).Text);
prtObj.Print Tab(82); Trim(frmTmdl2.txtPara(8).Text);
prtObj.Print Tab(97); Trim(frmTmdl2.txtPara(9).Text);
,
.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "Sediment Oxygen Demand, SOD (g/ft2/d) :";
prtObj.Print Tab(67); Trim(frmTmdl2.txtPara(1).Text);
prtObj.Print Tab(82); Trim(frmTmdl2.txtPara(2).Text);
prtObj.Print Tab(97); Trim(frmTmdl2.txtPara(3).Text);
,
'prtObj.Line (6400, y)-(10800, y)
'prtObj.Line -(10800, y + 1400)
'prtObj.Line -(6400, y + 1400)

```



```

    prtObj.Line -(6400, y)
    '
End With
'+++++
'This is the end of Page #1
'
With prtObj
    .Font.Bold = False
    .Font.Size = 10
    If prtObj Is Printer Then
        prtObj.NewPage
        .CurrentX = intMargin
        .CurrentY = 720
        prtObj.Print "Page 2 of 6"
    Else
        .CurrentX = intMargin
        .CurrentY = 15840 * 1# + 720
        prtObj.Print "Page 2 of 6"
    End If
    '
    .Font.Size = 12
    .CurrentX = intMargin
    .CurrentY = .CurrentY + 780
    prtObj.Print "V.      PROPOSED WASTELOAD ALLOCATIONS (WLA)"
    '
    .Font.Size = 11
    .CurrentY = .CurrentY + 600
    y = .CurrentY - 100
    '
    'Draw a simple table for WLA.
    '
    prtObj.Line (2400, y)-(9500, y)
    prtObj.Line -(9500, y + 1400)
    prtObj.Line -(2400, y + 1400)
    prtObj.Line -(2400, y)
    '
    prtObj.Line (2400, y + 400)-(9500, y + 400)
    prtObj.Line (3800, y)-(3800, y + 1400)

    'Fill the table
    .Font.Size = 10
    .CurrentX = intMargin
    .CurrentY = y + 100
    prtObj.Print Tab(48); "CBOD"; Tab(67); "NH3-N"; Tab(85); "D.O."
    '
    .CurrentY = .CurrentY + 100
    prtObj.Print Tab(30); "Summer"; Tab(50);
    prtObj.Print Trim(frmTmdl3.txtwla(0).Text); Tab(69);
    prtObj.Print Trim(frmTmdl3.txtwla(3).Text); Tab(87);
    prtObj.Print Trim(frmTmdl3.txtwla(6).Text)
    '
    .CurrentY = .CurrentY + 60
    prtObj.Print Tab(30); "Spring"; Tab(50);
    prtObj.Print Trim(frmTmdl3.txtwla(1).Text); Tab(69);
    prtObj.Print Trim(frmTmdl3.txtwla(4).Text); Tab(87);
    prtObj.Print Trim(frmTmdl3.txtwla(7).Text)
    '

```

```

.CurrentY = .CurrentY + 60
prtObj.Print Tab(30); "Winter"; Tab(50);
prtObj.Print Trim(frmTmdl3.txtwla(2).Text); Tab(69);
prtObj.Print Trim(frmTmdl3.txtwla(5).Text); Tab(87);
prtObj.Print Trim(frmTmdl3.txtwla(8).Text)
'
'Print Reserved Capacity or Safety Factor.
.CurrentX = intMargin
.CurrentY = .CurrentY + 1000
.Font.Size = 12
prtObj.Print "VI. RESERVED CAPACITY AND SAFETY FACTOR"
'
.Font.Size = 10
.CurrentY = .CurrentY + 600
y = .CurrentY - 100
'
'Draw a rectangular table for Reserved capacity and safety factor.
prtObj.Line (1500, y)-(10700, y)
prtObj.Line -(10700, y + 2000)
prtObj.Line -(1500, y + 2000)
prtObj.Line -(1500, y)
'
'horizontal lines
prtObj.Line (2800, y + 650)-(9300, y + 650)
prtObj.Line (1500, y + 1000)-(10700, y + 1000)
'vertical lines
prtObj.Line (2800, y)-(2800, y + 2000)
prtObj.Line (5000, y)-(5000, y + 2000)
prtObj.Line (7200, y)-(7200, y + 2000)
prtObj.Line (9300, y)-(9300, y + 2000)
'
.CurrentX = intMargin
.CurrentY = y + 100
prtObj.Print Tab(35); "Maximum Loading"; Tab(60);
prtObj.Print "Allocated Loading"; Tab(85);
prtObj.Print "Reserved Loading"; Tab(105);
prtObj.Print "Reserved/Max"
'
.CurrentY = .CurrentY + 40
prtObj.Print Tab(20); "Season";
prtObj.Print Tab(40); "(lbs/day)"; Tab(65); "(lbs/day)";
prtObj.Print Tab(90); "(lbs/day)"; Tab(110); "(%)"
'
.CurrentY = .CurrentY + 100
prtObj.Print Tab(35); "CBOD5"; Tab(47); "NH3-N"; Tab(59);
prtObj.Print "CBOD5"; Tab(71); "NH3-N"; Tab(83);
prtObj.Print "CBOD5"; Tab(95); "NH3-N"; Tab(107);
'
.CurrentY = .CurrentY + 100
prtObj.Print Tab(20); "Summer"; Tab(37);
prtObj.Print Trim(frmTmdl3.txtwla(21).Text); Tab(49);
prtObj.Print Trim(frmTmdl3.txtwla(22).Text); Tab(61);
prtObj.Print Trim(frmTmdl3.txtwla(23).Text); Tab(73);
prtObj.Print Trim(frmTmdl3.txtwla(24).Text); Tab(85);
prtObj.Print Trim(frmTmdl3.txtwla(25).Text); Tab(97);
prtObj.Print Trim(frmTmdl3.txtwla(26).Text); Tab(109);
prtObj.Print Trim(frmTmdl3.txtwla(27).Text)

```

```

,
        Page-01: "Main"
        Page-02: "Index"
.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "Spring"; Tab(37);
prtObj.Print Trim(frmTmdl3.txtwla(28).Text); Tab(49);
prtObj.Print Trim(frmTmdl3.txtwla(29).Text); Tab(61);
prtObj.Print Trim(frmTmdl3.txtwla(30).Text); Tab(73);
prtObj.Print Trim(frmTmdl3.txtwla(31).Text); Tab(85);
prtObj.Print Trim(frmTmdl3.txtwla(32).Text); Tab(97);
prtObj.Print Trim(frmTmdl3.txtwla(33).Text); Tab(109);
prtObj.Print Trim(frmTmdl3.txtwla(34).Text)
,

.CurrentY = .CurrentY + 60
prtObj.Print Tab(20); "Winter"; Tab(37);
prtObj.Print Trim(frmTmdl3.txtwla(35).Text); Tab(49);
prtObj.Print Trim(frmTmdl3.txtwla(36).Text); Tab(61);
prtObj.Print Trim(frmTmdl3.txtwla(37).Text); Tab(73);
prtObj.Print Trim(frmTmdl3.txtwla(38).Text); Tab(85);
prtObj.Print Trim(frmTmdl3.txtwla(39).Text); Tab(97);
prtObj.Print Trim(frmTmdl3.txtwla(40).Text); Tab(109);
prtObj.Print Trim(frmTmdl3.txtwla(41).Text)
,

'Print the Location of DO sags.
'Print Reserved Capacity or Safety Factor.
.CurrentX = intMargin
.CurrentY = .CurrentY + 1000
.Font.Size = 12
prtObj.Print "VII. LOCATIONS OF D.O. SAGS"
,

.Font.Size = 10
.CurrentY = .CurrentY + 600
y = .CurrentY - 100
,

'Draw a rectangular table for Reserved capacity and safety factor.
prtObj.Line (2500, y)-(9500, y)
prtObj.Line -(9500, y + 1950)
prtObj.Line -(2500, y + 1950)
prtObj.Line -(2500, y)
,

'Horizontal lines
prtObj.Line (2500, y + 950)-(9500, y + 950)
prtObj.Line (3700, y + 350)-(9500, y + 350)
,

'Vertical lines
,

prtObj.Line (3700, y)-(3700, y + 1950)
prtObj.Line (6500, y)-(6500, y + 1950)
,

.CurrentX = intMargin
.CurrentY = y + 100
prtObj.Print Tab(55); "0.0 cfs"; Tab(85); "1.0 cfs"
,

.CurrentY = .CurrentY + 100
prtObj.Print Tab(32); "Season";
prtObj.Print Tab(45); "Min D.O."; Tab(60); "River Mile";
prtObj.Print Tab(78); "Min D.O."; Tab(93); "River Mile"
,

'.CurrentY = .CurrentY + 60

```

```

prtObj.Print Tab(46); "(mg/l)"; Tab(62); "(miles)";
prtObj.Print Tab(79); "(mg/l)"; Tab(95); "(miles)"
,
.CurrentY = .CurrentY + 100
prtObj.Print Tab(32); "Summer"; Tab(46);
prtObj.Print Excelobj.range("C109").Text; Tab(62);
prtObj.Print Excelobj.range("E109").Text; Tab(79);
prtObj.Print Excelobj.range("F109").Text; Tab(95);
prtObj.Print Excelobj.range("H109").Text
,
.CurrentY = .CurrentY + 60
prtObj.Print Tab(32); "Spring"; Tab(46);
prtObj.Print Excelobj.range("C110").Text; Tab(62);
prtObj.Print Excelobj.range("E110").Text; Tab(79);
prtObj.Print Excelobj.range("F110").Text; Tab(95);
prtObj.Print Excelobj.range("H110").Text
,
.CurrentY = .CurrentY + 60
prtObj.Print Tab(32); "Winter"; Tab(46);
prtObj.Print Excelobj.range("C111").Text; Tab(62);
prtObj.Print Excelobj.range("E111").Text; Tab(79);
prtObj.Print Excelobj.range("F111").Text; Tab(95);
prtObj.Print Excelobj.range("H111").Text
,
End With
Set Excelobj = Nothing
'This is the end of page #2.
'++++++
With prtObj
    .Font.Size = 10
    If prtObj Is Printer Then
        prtObj.NewPage
        .CurrentX = intMargin
        .CurrentY = 720
        prtObj.Print "Page 3 of 6"
    Else
        .CurrentX = intMargin
        .CurrentY = 15840 * 2# + 720
        prtObj.Print "Page 3 of 6"
    End If
    .Font.Size = 12
End With
,
Set Excelobj = frmModel.OLE1.object.sheets("Summer")
Table prtObj, Excelobj, 2
Set Excelobj = Nothing
,
'This is the end of page #3.
'++++++
With prtObj
    .Font.Size = 10
    If prtObj Is Printer Then
        prtObj.NewPage
        .CurrentX = intMargin
        .CurrentY = 720
        prtObj.Print "Page 4 of 6"
    Else

```

```

        .CurrentX = intMargin
        .CurrentY = 15840 * 3# + 720
        prtObj.Print "Page 4 of 6"
    End If
    .Font.Size = 12
End With

Set Excelobj = frmModel.OLE1.object.sheets("Spring")
Table prtObj, Excelobj, 3
Set Excelobj = Nothing

'This is the end of page #4.
'+++++
With prtObj
    .Font.Size = 10
    If prtObj Is Printer Then
        prtObj.NewPage
        .CurrentX = intMargin
        .CurrentY = 720
        prtObj.Print "Page 5 of 6"
    Else
        .CurrentX = intMargin
        .CurrentY = 15840 * 4# + 720
        prtObj.Print "Page 5 of 6"
    End If
    .Font.Size = 12
End With

Set Excelobj = frmModel.OLE1.object.sheets("Winter")
Table prtObj, Excelobj, 4
Set Excelobj = Nothing
End Sub

```

```

-----
Private Sub Table(prtObj As Object, Excelobj As Object, I As Long)
    Dim N As Integer
    Dim y As Long
    Dim intMargin As Integer

    intMargin = 1440

    With prtObj
        .Font.Size = 12
        .Font.Bold = True
        If prtObj Is Printer Then
            'prtObj.NewPage
            .CurrentX = intMargin
            .CurrentY = 1240
        Else
            .CurrentX = intMargin
            .CurrentY = 15840 * I + 1440
        End If
    End With

    Select Case I
        Case 2:
    
```

```

    prtObj.Print Tab(45); "SUMMER"
Case 3:
    prtObj.Print Tab(45); "SPRING"
Case 4:
    prtObj.Print Tab(45); "WINTER"
End Select
'
.CurrentY = .CurrentY + 300
y = .CurrentY - 100
'
'Draw a simple table for WLA.
'
prtObj.Line (1440, y)-(10800, y)
prtObj.Line -(10800, y + 12450)
prtObj.Line -(1440, y + 12450)
prtObj.Line -(1440, y)
'
prtObj.Line (2550, y)-(2550, y + 12450)
'Fill the table.
.Font.Bold = False
If frmModel.OLE1.object.sheets("Data").range("F15").Value Then
' Do following if 7Q2 <> 0 (non-zero --> true).
    prtObj.Line (1440, y + 700)-(10800, y + 700)
    .CurrentX = intMargin
    .CurrentY = y + 100
    '
    prtObj.Print Tab(16); "Distance"; Tab(35); "CBOD5";
    prtObj.Print Tab(60); "NH3-N"; Tab(85); "D.O."
    .Font.Size = 10
    prtObj.Print Tab(20); "(miles)"; Tab(43); "(mg/l)";
    prtObj.Print Tab(71); "(mg/l)"; Tab(99); "(mg/l)"
    .CurrentY = .CurrentY + 100
    .Font.Size = 11
    '
    For N = 0 To 40
        .CurrentY = .CurrentY + 20
        prtObj.Print Tab(17); Excelobj.cells(33 + N, 2).Text;
        prtObj.Print Tab(37); Excelobj.cells(33 + N, 4).Text;
        prtObj.Print Tab(61); Excelobj.cells(33 + N, 8).Text;
        prtObj.Print Tab(85); Excelobj.cells(33 + N, 12).Text;
    Next N
    '
Else 'If 7Q2 = 0.0 cfs
'
    prtObj.Line (6650, y)-(6650, y + 12450)
    prtObj.Line (2550, y + 300)-(10800, y + 300)
    prtObj.Line (1440, y + 900)-(10800, y + 900)
    '
    .CurrentX = intMargin
    .CurrentY = y + 20
    '
    prtObj.Print Tab(42); "0.0 cfs"; Tab(80); "1.0 cfs"
    '
    .CurrentY = .CurrentY + 40
    prtObj.Print Tab(16); "Distance"; Tab(28); "CBOD5";
    prtObj.Print Tab(42); "NH3-N"; Tab(55); "D.O."; Tab(68);
    prtObj.Print "CBOD5"; Tab(81); "NH3-N"; Tab(94); "D.O."

```

Margin of Safety & Plots

```
'  
    .Font.Size = 10  
    prtObj.Print Tab(20); "(miles)"; Tab(34); "(mg/l)";  
    prtObj.Print Tab(50); "(mg/l)"; Tab(64); "(mg/l)";  
    prtObj.Print Tab(81); "(mg/l)"; Tab(95); "(mg/l)";  
    prtObj.Print Tab(109); "(mg/l)"  
    .CurrentY = .CurrentY + 100  
    .Font.Size = 11  
'  
For N = 0 To 40  
    .CurrentY = .CurrentY + 20  
    prtObj.Print Tab(17); Excelobj.cells(33 + N, 2).Text;  
    prtObj.Print Tab(29); Excelobj.cells(33 + N, 19).Text;  
    prtObj.Print Tab(43); Excelobj.cells(33 + N, 23).Text;  
    prtObj.Print Tab(55); Excelobj.cells(33 + N, 27).Text;  
  
    prtObj.Print Tab(69); Excelobj.cells(33 + N, 4).Text;  
    prtObj.Print Tab(82); Excelobj.cells(33 + N, 8).Text;  
    prtObj.Print Tab(94); Excelobj.cells(33 + N, 12).Text;  
Next N  
End If  
End With  
End Sub  
'
```

Margin of Safety & Plots

```
' *****  
' This routine is to draw a single Dissolved Oxygen (D.O.) profile  
' (D.O. vs River miles). According to the passed parameters, this  
' routine will get XY coordinates from the Embedded Water Quality  
' Model, and draw the graph the specified location.  
' *****  
'  
Private Sub PlotDO(ByVal prtObj2 As Object, gtop As Integer, _  
                  gleft As Integer, gheight As Integer, _  
                  gwidth As Integer, colnum As Integer)  
-----  
' Given a set of XY data, this routine will automatically choose  
' appropriate x and y coordinates (min & max) and plot the data  
' at the location specified by the passed routine parameters.  
-----  
'  
' prtObj2 -- object on which plot is drawn  
' gtop -- coordinate of the top of the graph  
' gleft -- coordinate of the left of the graph  
' gheight -- the height of the graph  
' gwidth -- the width of the graph  
' colnum -- the column # which contains D.O. data in OLE1  
'  
Dim I As Integer  
Dim xnum As Integer, ynum As Integer  
Dim LM As Single      ' left margin of the graph  
Dim BM As Single      ' bottom margin of the graph  
Dim xscale As Integer ' max number of x axis  
Dim yscalemain As Integer ' max number of y axis  
Dim yscalemax As Integer ' min number of y axis  
Dim x(0 To 40) As Single ' x(i) - x coordinate of point i  
Dim y(0 To 40) As Single ' y(i) - y coordinate of point i  
Dim xmax As Single     ' max value of x (stream length)  
Dim ymax As Single     ' max value of y (D.O.)  
Dim ymin As Single     ' min value of y (D.O.)  
Dim xratio As Single   ' x ratio for plot x(i)  
Dim yratio As Single   ' y ratio for plot y(i)  
Dim temp As Single  
  
Dim obj As Object  
Dim picdo As Object  
Set obj = frmModel.OLE1.object.sheets("Plot")  
Set picdo = prtObj2  
  
Let LM = 300  
Let BM = 450  
  
' Get stream DO data from the Excel sheet embedded in Form1.  
  
For I = 0 To 40  
    Let x(I) = obj.cells(4 + I, 1).Value  
    Let y(I) = obj.cells(4 + I, colnum).Value  
Next I  
Let xmax = obj.range("A44").Value
```



```

Let ymin = obj.cells(46, colnum).Value
Let ymax = obj.cells(47, colnum).Value

' Define X and Y axis (x always starts at 0 and ends at xscale).
'
Let xscale = Int(xmax) + 1
Let yscalemin = Int(ymin) - 1
Let yscalemax = Int(ymax) + 2
Let xnum = xscale
Let ynum = (yscalemax - yscalemin)
Let xratio = (gwidth - LM) / xnum
Let yratio = (gheight - BM) / ynum
-----
' Draw axes, including ticks and numbers.
-----
' Draw y axis, ticks and numbers.

picdo.DrawWidth = 1
picdo.Line ((gleft + LM), gtop)-((gleft + LM), (gtop + gheight - BM))
picdo.CurrentX = gleft
For I = 0 To ynum
    picdo.CurrentX = gleft
    picdo.CurrentY = gtop + gheight - BM - yratio * I - 80
    temp = yscalemin + I
    picdo.Print Str(temp)
    picdo.CurrentY = gtop + gheight - BM - yratio * I
    picdo.Line ((gleft + LM - 30), picdo.CurrentY)-
                ((gleft + LM + 30), picdo.CurrentY)
Next I

' Label y axis.

picdo.CurrentX = gleft - 0.3 * picdo.TextWidth("DO (mg/l)")
picdo.CurrentY = gtop - picdo.TextHeight("DO (mg/l)") - 80
picdo.Print "DO (mg/l)"

' Draw x axis, ticks and numbers.

picdo.CurrentY = gtop + gheight - BM
temp = picdo.CurrentY
picdo.Line (gleft + LM, picdo.CurrentY)-
            (gleft + gwidth, picdo.CurrentY)

For I = 0 To xnum
    picdo.CurrentX = gleft + LM + xratio * I - 80
    picdo.CurrentY = gtop + gheight - BM + 40
    picdo.Print Str(I)
    picdo.CurrentX = gleft + LM + xratio * I
    picdo.Line (picdo.CurrentX, temp - 30)-
                (picdo.CurrentX, temp + 30)
Next

' Label x axis.

picdo.CurrentX = gleft + 0.5 *
                (gwidth - picdo.TextWidth("Distance (miles)"))
picdo.CurrentX = picdo.CurrentX + LM / 2

```

```

picdo.CurrentY = gtop + gheight - picdo.TextHeight("Distance
(miles)")
picdo.Print "Distance (miles)"
,
'-----
' Draw dissolved oxygen profile.
'-----
,
For I = 0 To 40
    Let x(I) = x(I) * xratio + LM + gleft
    Let y(I) = gheight + gtop - BM - (y(I) - yscalemin) * yratio
Next
,
picdo.DrawWidth = 2
picdo.Line (x(0), y(0))-(x(1), y(1)), vbBlue
For I = 2 To 40
    picdo.Line -(x(I), y(I)), vbBlue
Next I
Set picdo = Nothing
Set obj = Nothing
End Sub
,
'*****
Public Sub DrawPlot(prtObj1 As Object)
,
    Dim top As Integer
    Dim left As Integer
    Dim width As Integer
    Dim height As Integer
    Dim label As String
    Dim x As Single, y As Single
,
    Dim obj As Object
    Set obj = frmModel.OLE1.object.sheets("Data")
    Dim pic As Object
    Set pic = prtObj1
,
    If obj.range("F15") = 0 Then
        ' Determine the width and height of each DO plot.
        width = pic.ScaleWidth / 3
        height = (pic.ScaleHeight - 3000) / 4.5
,
        ' Draw DO plot under Summer 0.0 cfs condition.
        top = 0.35 * height + 2000
        left = 0.25 * width + 300
        Call PlotDO(pic, top, left, height, width, 2)
        DoEvents
        'Write title of the plot.
        x = left + width / 2 - pic.TextWidth("Summer 0.0 cfs") / 2
        y = top - pic.TextHeight("Summer 0.0 cfs")
        pic.CurrentX = x
        pic.CurrentY = y
        pic.Font.Bold = True
        pic.Print "Summer 0.0 cfs"
        pic.Font.Bold = False
,
        ' Draw DO plot under Spring 0.0 cfs condition.

```

```

top = 1.85 * height + 2000
Call PlotDO(pic, top, left, height, width, 4)
DoEvents
'Write title of the plot.
x = left + width / 2 - pic.TextWidth("Spring 0.0 cfs") / 2
y = top - pic.TextHeight("Spring 0.0 cfs")
pic.CurrentX = x
pic.CurrentY = y
pic.Font.Bold = True
pic.Print "Spring 0.0 cfs"
pic.Font.Bold = False
,
' Draw DO plot under Winter 0.0 cfs condition.
top = 3.35 * height + 2000
Call PlotDO(pic, top, left, height, width, 6)
DoEvents
'Write title of the plot.
x = left + width / 2 - pic.TextWidth("Winter 0.0 cfs") / 2
y = top - pic.TextHeight("Winter 0.0 cfs")
pic.CurrentX = x
pic.CurrentY = y
pic.Font.Bold = True
pic.Print "Winter 0.0 cfs"
pic.Font.Bold = False
,
' Draw DO plot under Summer 1.0 cfs condition.
top = 0.35 * height + 2000
left = 1.75 * width - 100
Call PlotDO(pic, top, left, height, width, 3)
DoEvents
'Write title of the plot.
x = left + width / 2 - pic.TextWidth("Summer 1.0 cfs") / 2
y = top - pic.TextHeight("Summer 1.0 cfs")
pic.CurrentX = x
pic.CurrentY = y
pic.Font.Bold = True
pic.Print "Summer 1.0 cfs"
pic.Font.Bold = False
,
' Draw DO plot under Spring 1.0 cfs condition.
top = 1.85 * height + 2000
Call PlotDO(pic, top, left, height, width, 5)
DoEvents
x = left + width / 2 - pic.TextWidth("Spring 1.0 cfs") / 2
y = top - pic.TextHeight("Spring 1.0 cfs")
pic.CurrentX = x
pic.CurrentY = y
pic.Font.Bold = True
pic.Print "Spring 1.0 cfs"
pic.Font.Bold = False
,
' Draw DO plot under Summer 1.0 cfs condition.
top = 3.35 * height + 2000
Call PlotDO(pic, top, left, height, width, 7)
DoEvents
'Write title of the plot.
x = left + width / 2 - pic.TextWidth("Winter 1.0 cfs") / 2

```

```

y = top - pic.TextHeight("Winter 1.0 cfs")
pic.CurrentX = x
pic.CurrentY = y
pic.Font.Bold = True
pic.Print "Winter 1.0 cfs"
pic.Font.Bold = False
,
Else
,
width = pic.ScaleWidth / 1.5
height = (pic.ScaleHeight - 3000) / 4.5
label = obj.range("F15").Text
,
' Draw DO plot under Summer 7Q2 condition.
top = 0.35 * height + 2000
left = 0.25 * width
Call PlotDO(pic, top, left, height, width, 3)
DoEvents
'Write title of the plot.
label = "Summer " & label & " cfs"
x = left + width / 2 - pic.TextWidth(label) / 2
y = top - pic.TextHeight(label)
pic.CurrentX = x
pic.CurrentY = y
pic.Font.Bold = True
pic.Font.Size = pic.Font.Size + 2
pic.Print label
pic.Font.Bold = False
pic.Font.Size = pic.Font.Size - 2
,
' Draw DO plot under Spring 7Q2 condition.
top = 1.85 * height + 2000
Call PlotDO(pic, top, left, height, width, 5)
DoEvents
'write title of the plot
label = obj.range("F15").Text
label = "Spring " & label & " cfs"
x = left + width / 2 - pic.TextWidth(label) / 2
y = top - pic.TextHeight(label)
pic.CurrentX = x
pic.CurrentY = y
pic.Font.Bold = True
pic.Font.Size = pic.Font.Size + 2
pic.Print label
pic.Font.Bold = False
pic.Font.Size = pic.Font.Size - 2
,
' Draw DO plot under Summer 7Q2 condition.
top = 3.35 * height + 2000
Call PlotDO(pic, top, left, height, width, 7)
DoEvents
'write title of the plot
label = obj.range("F15").Text
label = "Winter " & label & " cfs"
x = left + width / 2 - pic.TextWidth(label) / 2
y = top - pic.TextHeight(label)
pic.CurrentX = x

```

```

pic.CurrentY = y
pic.Font.Bold = True
pic.Font.Size = pic.Font.Size + 2
pic.Print label
pic.Font.Bold = False
pic.Font.Size = pic.Font.Size - 2
'
End If
Set pic = Nothing
Set obj = Nothing
End Sub
'
'*****
Public Sub Figure(prtObj As Object)
  If prtObj Is Printer Then
    prtObj.NewPage
  End If
  '
  With prtObj
    .CurrentY = 1500
    .Font.Bold = True
    .Font.Size = 14
    prtObj.Print Tab(22);
    prtObj.Print "Receiving Stream Dissolved Oxygen Profile"
    '
    .CurrentX = 1440
    .CurrentY = 720
    .Font.Name = "Arial"
    .Font.Bold = False
    .Font.Size = 10
    prtObj.Print "Page "; Str(6); " of 6"
  End With
End Sub
'
'*****
Sub SafetyMargin()
  Dim I As Integer, flag As Integer
  Dim count As Integer
  Dim temp As Double
  Dim wqs(1 To 3) As Double
  Dim streamDO(1 To 3) As Double
  Dim DODiff(1 To 3) As Double
  Dim SF0(1 To 3) As Double
  Dim SF(1 To 3) As Double
  Dim msg As String
  Dim button As Integer
  Dim ok
  '
  ' wqs -- water quality standards
  ' streamDO -- Dissolved Oxygen concentration in the stream
  ' DODiff - difference between streamDO and wqs
  ' SF -- Safety Factor in the current round of calculation
  ' SF0 -- Safety factor in the previous round of calculation
  '
  Dim obj As Object
  Set obj = frmModel.OLE1.object.sheets("Data")
  '

```

```

' Obj refers to which sheet we want to access
'
obj.range("F83").Value = 0
obj.range("F84").Value = 1
obj.range("F85").Value = 1
obj.range("F86").Value = 1
'
For I = 1 To 3
    wqs(I) = obj.cells(29 + I, 7).Value
    streamDO(I) = obj.cells(71 + I, 8).Value
    DOdiff(I) = streamDO(I) - wqs(I)
    SF0(I) = 1
    SF(I) = SF0(I)
'
-----
' Check to see if instream DO, under 0.0 cfs and 1.0 cfs condition,
' meets water quality standards.  If instream DO meets standards,
' continue to calculate Safety Margin.  Otherwise, prompt a warning
' message and quit.
'-----
' Tests on 1.0 cfs condition.
'
    If DOdiff(I) < 0 Then
        Select Case (I)
            Case 1
                msg = "Instream DO does not meet Water Quality" & _
                    " Standards for the summer " & vbCrLf & _
                    "under 1.0 cfs condition. Safety " & _
                    "Factor can not be calculated. " & vbCrLf & _
                    "Reduce the waste load and try again."
                button = vbOKOnly + vbExclamation
                ok = MsgBox(msg, button, "Info Message")
                If ok = vbOK Then Exit Sub
            Case 2
                msg = "Instream DO does not meet Water Quality" & _
                    " Standards for the spring " & vbCrLf & _
                    "under 1.0 cfs condition. Safety " & _
                    "Factor can not be calculated. " & vbCrLf & _
                    "Reduce the waste load and try again."
                button = vbOKOnly + vbExclamation
                ok = MsgBox(msg, button, "Info Message")
                If ok = vbOK Then Exit Sub
            Case 3
                msg = "Instream DO does not meet Water Quality" & _
                    " Standards for the winter " & vbCrLf & _
                    "under 1.0 cfs condition. Safety " & _
                    "Factor can not be calculated. " & vbCrLf & _
                    "Reduce the waste load and try again."
                button = vbOKOnly + vbExclamation
                ok = MsgBox(msg, button, "Info Message")
                If ok = vbOK Then Exit Sub
        End Select
    End If
'
' Tests on 0.0 cfs condition.
'
    If obj.range("F15") = 0 Then

```

```

If obj.cells(71 + I, 6) < 2 Then
  Select Case (I)
    Case 1
      msg = "Instream DO does not meet Water Quality" & _
            " Standards for the summer " & vbCrLf & _
            "under 0.0 cfs condition. Safety " & _
            "Factor can not be calculated. " & vbCrLf & _
            "Reduce the waste load and try again."
      button = vbOKOnly + vbExclamation
      ok = MsgBox(msg, button, "Info Message")
      If ok = vbOK Then Exit Sub
    Case 2
      msg = "Instream DO does not meet Water Quality" & _
            " Standards for the spring " & vbCrLf & _
            "under 0.0 cfs condition. Safety " & _
            "Factor can not be calculated. " & vbCrLf & _
            "Reduce the waste load and try again."
      button = vbOKOnly + vbExclamation
      ok = MsgBox(msg, button, "Info Message")
      If ok = vbOK Then Exit Sub
    Case 3
      msg = "Instream DO does not meet Water Quality" & _
            " Standards for the winter " & vbCrLf & _
            "under 0.0 cfs condition. Safety " & _
            "Factor can not be calculated. " & vbCrLf & _
            "Reduce the waste load and try again."
      button = vbOKOnly + vbExclamation
      ok = MsgBox(msg, button, "Info Message")
      If ok = vbOK Then Exit Sub
  End Select
End If
End If
Next I
'
' -----
' Calculate safety margin for each season based on the following
' strategies:
' 1. Start from 1, increase by the increment of 0.5 until instream DO
'    is no longer greater than Water Quality Standards.
' 2. Divide the last interval into two, determine which part the
'    safety margin will fall into. Record that part as last interval.
' 3. Repeat step 2 until a satisfactory solution is found or until 50
'    iterations have been used up.
' -----
'
For I = 1 To 3
  count = 0
  flag = 0
  Do
    If DOdiff(I) > 0.001 Then
      SF0(I) = SF(I)
      SF(I) = SF0(I) + 0.5
      obj.cells(83 + I, 6).Value = SF(I)
      streamDO(I) = obj.cells(71 + I, 8).Value
      DOdiff(I) = streamDO(I) - wqs(I)
      count = count + 1
    ElseIf DOdiff(I) < -0.001 Then

```

```

Do
    SF(I) = SF0(I) + 0.5 * (SF(I) - SF0(I))    ' update safety
factor
    obj.cells(83 + I, 6).Value = SF(I)
    streamDO(I) = obj.cells(71 + I, 8).Value
    DOdiff(I) = streamDO(I) - wqs(I)
    '
    ' This if-block determines the latest interval.
    '
    If DOdiff(I) > 0 Then
        temp = SF0(I)
        SF0(I) = SF(I)
        SF(I) = temp + 2 * (SF(I) - temp)
    End If
    count = count + 1
    Loop Until (Abs(DOdiff(I)) < 0.001 Or count > 50)
Else
    flag = 1
End If
Loop Until (count > 50 Or flag = 1)

If (count > 50) Then
    ok = MsgBox("The program cannot find a satisfactory solution "& _
        "(error < 0.001) within 50 iterations. " & vbCrLf & " Do" & _
        " you want to display the best solution the program found?", _
        vbYesNo + vbQuestion, "Info Message")
    If ok = vbNo Then obj.cells(83 + I, 6).Value = 1
End If
Next I
obj.range("F83").Value = 1
Set obj = Nothing
End Sub

```


Menu Options

Option Explicit

```
'-----  
'This general procedure will store data  
'from the OLE container to the designated file.  
'  
Public Sub SaveDataAs(dlgBox As Object)  
    'Get file name from user (common dialog box).  
    dlgBox.Filter = "All Files|*.*|Binary File|*.bin"  
    dlgBox.FilterIndex = 1  
    dlgBox.ShowSave  
  
    Dim strFileName As String  
    strFileName = dlgBox.FileName  
  
    'Check if a valid file name is provided.  
    'the file name is not empty.  
    If strFileName <> "" Then  
        'Get handle for file.  
        Dim lHandle As Long  
        lHandle = FreeFile  
        'Open it as binary.  
        Open strFileName For Binary As #lHandle  
        'Write to file.  
        frmModel.OLE1.SaveToFile lHandle  
        'Close file.  
        Close #lHandle  
    End If  
    blnChanged = False  
End Sub  
'
```

```
'-----  
'This general procedure will update data  
'from OLE container with current file name.  
'Current file name is stored in a public variable  
'available through the whole application.  
Public Sub SaveData()  
    Dim strFileName As String  
    strFileName = CurrentName  
  
    'Get handle for file.  
    Dim lHandle As Long  
    lHandle = FreeFile  
    'Open it as binary.  
    Open strFileName For Binary As #lHandle  
    'Write to file.  
    frmModel.OLE1.SaveToFile lHandle  
    'Close file.  
    Close #lHandle  
    blnChanged = False  
End Sub  
'
```

```
'-----  
'This general procedure will retrieve data  
'from a designated file into the OLE container.
```

```

Private Function RetrieveData(dlgBox As Object) As Boolean
    Dim blnCancel As Boolean
    'Get file name for user.
    dlgBox.Filter = "All Files|*.*|Binary File|*.bin"
    dlgBox.FilterIndex = 1
    dlgBox.ShowOpen

    Dim strFileName As String
    strFileName = dlgBox.FileName
    CurrentName = strFileName

    'Check if a valid file name is provided.
    'The file name is not empty.
    If strFileName <> "" Then
        'Get handle for file
        Dim lHandle As Long
        lHandle = FreeFile
        'Open it as binary
        Open strFileName For Binary As #lHandle
        'Read from the file
        frmModel.OLE1.ReadFromFile lHandle
        'Close the file
        Close #lHandle
    Else
        blnCancel = True
    End If
    RetrieveData = blnCancel
End Function

```

```

'-----
'This procedure will extract the information from the
'just-opened TMDL model, and fill various input slots
'in the user interface.
'The model is embedded in the application; it is normally
'hidden from the user unless the user chose to view the model.

```

```

Private Sub ReadData()
    Dim I As Integer
    Dim obj As Object
    Set obj = frmModel.OLE1.object.sheets("Data")

    'Update information on form #1.
    With frmTmdl1
        .Text1(0).Text = " " & obj.range("E4").Text '- Engineer
        .Text1(3).Text = " " & obj.range("E3").Text '- Facility name
        .Text1(2).Text = " " & obj.range("E5").Text '- Date performed
        .Text1(4).Text = " " & obj.range("C9").Text '- Location
        .Text1(5).Text = " " & obj.range("C10").Text '- County
        .Text1(6).Text = " " & obj.range("H9").Text '- Segment #
        .Text1(7).Text = " " & obj.range("D12").Text '- Design flow
        .Text1(9).Text = " " & obj.range("D13").Text '- Stream name
        .Text1(10).Text = " " & obj.range("F15").Text '- 7Q2
        .Text1(11).Text = " " & obj.range("H14").Text '- Stream length
        .Text1(12).Text = " " & obj.range("F17").Text '- Stream slope
        .Text1(13).Text = " " & obj.range("F18").Text '- Side slope
        .Text1(14).Text = " " & obj.range("F19").Text '- Mannings' #
    End With

```

```

For I = 0 To 2      ' Water Quality Standards
  If obj.range("E30").Value = I + 1 Then
    .Option1(I).Value = True
  Else
    .Option1(I).Value = False
  End If
Next
'
End With
'Update information of form #2.
With frmTmdl2
  .txtPara(0).Text = "      " & obj.range("F43").Text  '- K1
  'update Ks
  .txtPara(1).Text = "      " & obj.range("H65").Text ' summer
  .txtPara(2).Text = "      " & obj.range("G65").Text ' spring
  .txtPara(3).Text = "      " & obj.range("F65").Text ' winter
  'update SOD
  .txtPara(4).Text = "      " & obj.range("H61").Text ' summer
  .txtPara(5).Text = "      " & obj.range("G61").Text ' spring
  .txtPara(6).Text = "      " & obj.range("F61").Text ' winter
  'update KN
  .txtPara(7).Text = "      " & obj.range("H63").Text ' summer
  .txtPara(8).Text = "      " & obj.range("G63").Text ' spring
  .txtPara(9).Text = "      " & obj.range("F63").Text ' winter
  '
For I = 0 To 3      ' Aeration coefficient
  If obj.range("F57").Value = I + 1 Then
    .Option1(I).Value = True
  Else
    .Option1(I).Value = False
  End If
Next
End With
'Update information on Form #3.
With frmTmdl3
  ' Update CBOD5.
  .txtwla(0).Text = "      " & obj.range("B72").Text ' summer
  .txtwla(1).Text = "      " & obj.range("B73").Text ' spring
  .txtwla(2).Text = "      " & obj.range("B74").Text ' winter
  ' Update NH3-N.
  .txtwla(3).Text = "      " & obj.range("C72").Text ' summer
  .txtwla(4).Text = "      " & obj.range("C73").Text ' spring
  .txtwla(5).Text = "      " & obj.range("C74").Text ' winter
  ' Update D.O.
  .txtwla(6).Text = "      " & obj.range("D72").Text ' summer
  .txtwla(7).Text = "      " & obj.range("D73").Text ' spring
  .txtwla(8).Text = "      " & obj.range("D74").Text ' winter
End With
Set obj = Nothing
Call getDO
Call getSafety
End Sub
'
'-----
Public Sub OpenFile()
  Dim blnCancel As Boolean
  blnCancel = RetrieveData(MDIForm1.CommonDialog1)

```

```

'If user clicked cancel in dialog box, do not open anything.
If blnCancel Then Exit Sub
'Call subroutine ReadData to read info from saved
'file onto user interface (text box, option button...).

```

```

ReadData

```

```

'Update the file name appeared on every form.
frmTmdl1.Caption = CurrentName & " - 1"
frmTmdl2.Caption = CurrentName & " - 2"
frmTmdl3.Caption = CurrentName & " - 3"
frmTmdl4.Caption = CurrentName & " - 4"
frmModel.Caption = CurrentName & " - Model"
'Hide other forms except the one from which the
'user opens the saved file.
frmTmdl2.Hide
frmTmdl3.Hide
frmTmdl4.Hide
frmModel.Hide
blnNewfile = False 'Open an existing file
blnChanged = False
End Sub

```

```

-----
Private Sub PrepareSave(myForm As Object)
'If user made any changes in the input sheets,
'update the embeded model first before saving.
If blnChanged Then
updateInput
updateWLA
Select Case myForm.Name
Case "frmTmdl1"
frmTmdl2.Hide
frmTmdl3.Hide
Case "frmTmdl2"
frmTmdl1.Hide
frmTmdl3.Hide
Case "frmTmdl3"
frmTmdl1.Hide
frmTmdl2.Hide
Case "frmTmdl4"
frmTmdl1.Hide
frmTmdl2.Hide
frmTmdl3.Hide

End Select
End If
End Sub

```

```

-----
Function CloseForm(myForm As Object) As Boolean
If blnChanged Then
Dim strMsg As String
Dim buttons As Integer
Dim Response As Integer
Dim canceled As Boolean
strMsg = "Inputs have been Changed." & vbCrLf _
& "Do you want to save the changes before Close?"

```

```

buttons = vbYesNoCancel + vbQuestion
Response = MsgBox(strMsg, buttons, "Confirmation")
If Response = vbYes Then
    If blnNewfile Then
        'Call SaveAs procedure.
        Call PrepareSave(myForm)
        Call SaveDataAs(MDIForm1.CommonDialog1)
    Else
        'Call Save procedure.
        Call PrepareSave(myForm)
        Call SaveData
    End If
    blnChanged = False
    Unload myForm
ElseIf Response = vbNo Then
    blnasked = True
    Unload myForm
Else
    canceled = True
End If
Else
    Unload myForm
End If
CloseForm = canceled
End Function

```

```

Sub ExitApp(myApp As Object)
    If blnChanged Then
        Dim strMsg As String
        Dim buttons As Integer
        Dim Response As Integer
        strMsg = "Inputs have been Changed." & vbCrLf _
            & "Do you want to save the changes before Exit?"
        buttons = vbYesNoCancel + vbQuestion
        Response = MsgBox(strMsg, buttons, "Confirmation")
        If Response = vbYes Then
            If blnNewfile Then
                'Call SaveAs procedure.
                Call PrepareSave(myApp)
                Call SaveDataAs(MDIForm1.CommonDialog1)
            Else
                'Call Save procedure.
                Call PrepareSave(myApp)
                Call SaveData
            End If
            Unload MDIForm1
        ElseIf Response = vbNo Then
            blnasked = True
            Unload MDIForm1
        End If
    Else
        Unload MDIForm1
    End If
End Sub

```

Desktop TMDL Model

Department Of Environmental Quality

Prepared BY: John Doe

4/28/00

APPENDIX B

**REPORT AND PLOT FROM THE DESKTOP TMDL
MODEL AND FROM THE ORIGINAL EXCEL MODEL**

Desktop TMDL Model

Oklahoma Department Of Environmental Quality

PERFORMED BY : John Doe
DATE : 4/28/99

I. FACILITY INFORMATION

Name of the Facility : City of Clinton
 Legal Description : SE/SW/SE S18, T11N, R10W
 Design Flow (MGD) : 0.060
 County & State : Custer County, Oklahoma

II. RECEIVING STREAM

Receiving Stream : Unnamed Tributary to Washita River
 Stream Segment # : 410200
 Stream Length (miles) : 5.00
 Stream Slope (ft/mile) : 5.100
 Low Critical Flow, 7Q2 (MGD) : 0.00
 Manning's Number : 0.100

III. WATER QUALITY CRITERIA OF RECEIVING STREAM

(WARM WATER AQUATIC COMMUNITY)

Summer : 5.00 mg/l
 Spring : 6.00 mg/l
 Winter : 6.00 mg/l

IV. RATE CONSTANTS AT 20 DEGREE C

CBOD Decay Rate, K1 (/day) : 0.4
 Rearation Rate, K2 (/day) : 9.78 -- under 0.0 cfs
 9.78 -- under 1.0 cfs

	Summer	Spring	Winter
CBOD Settling Rate, KS (/day) :	0.03	0.03	0.03
NBOD Decay Rate, KN (/day) :	0.3	0.3	0.3
Sediment Oxygen Demand, SOD (g/ft ² /d) :	0.06	0.075	0.1

V. PROPOSED WASTELOAD ALLOCATIONS (WLA)

	CBOD	NH3-N	D O
Summer	10.0	6.0	2.0
Spring	15.0	8.0	2.0
Winter	18.0	12.0	2.0

VI. RESERVED CAPACITY AND SAFETY FACTOR

Season	Maximum Loading (lbs/day)		Allocated Loading (lbs/day)		Reserved Loading (lbs/day)		Reserved/Max (%)
	CBOD5	NH3-N	CBOD5	NH3-N	CBOD5	NH3-N	
Summer	29.3	17.6	5.0	3.0	24.3	14.6	82.9%
Spring	38.5	20.5	7.5	4.0	31.0	16.5	80.5%
Winter	89.3	59.5	9.0	6.0	80.3	53.5	89.9%

VII. LOCATIONS OF D.O. SAGS

Season	0.0 cfs		1.0 cfs	
	Min D.O. (mg/l)	River Mile (miles)	Min D.O. (mg/l)	River Mile (miles)
Summer	2.00	0.00	5.84	0.00
Spring	2.00	0.00	5.59	0.00
Winter	2.00	0.00	7.52	0.00

SUMMER

Distance: (miles)	0.0 cfs			1.0 cfs		
	CBOD5 (mg/l)	NH3-N (mg/l)	D.O. (mg/l)	CBOD5 (mg/l)	NH3-N (mg/l)	D.O. (mg/l)
0.00	10.00	6.00	2.00	2.68	0.65	5.84
0.13	9.55	5.75	2.20	2.61	0.63	5.85
0.25	9.13	5.50	2.34	2.55	0.62	5.86
0.38	8.72	5.27	2.47	2.49	0.60	5.87
0.50	8.33	5.04	2.58	2.43	0.59	5.88
0.63	7.96	4.83	2.68	2.37	0.58	5.89
0.75	7.61	4.62	2.78	2.31	0.56	5.90
0.88	7.27	4.43	2.87	2.26	0.55	5.92
1.00	6.94	4.24	2.96	2.20	0.54	5.93
1.13	6.63	4.06	3.05	2.15	0.52	5.94
1.25	6.34	3.89	3.13	2.10	0.51	5.95
1.38	6.06	3.72	3.20	2.04	0.50	5.95
1.50	5.79	3.56	3.28	1.99	0.49	5.96
1.63	5.53	3.41	3.35	1.95	0.48	5.97
1.75	5.28	3.27	3.42	1.90	0.47	5.98
1.88	5.05	3.13	3.48	1.85	0.46	5.99
2.00	4.82	3.00	3.54	1.81	0.44	6.00
2.13	4.61	2.87	3.60	1.76	0.43	6.01
2.25	4.40	2.75	3.66	1.72	0.42	6.02
2.38	4.21	2.63	3.71	1.68	0.41	6.02
2.50	4.02	2.52	3.76	1.64	0.40	6.03
2.63	3.84	2.41	3.81	1.60	0.40	6.04
2.75	3.67	2.31	3.86	1.56	0.39	6.05
2.88	3.50	2.21	3.91	1.52	0.38	6.05
3.00	3.35	2.12	3.95	1.48	0.37	6.06
3.13	3.20	2.03	3.99	1.45	0.36	6.07
3.25	3.06	1.94	4.03	1.41	0.35	6.07
3.38	2.92	1.86	4.07	1.38	0.34	6.08
3.50	2.79	1.78	4.10	1.35	0.34	6.09
3.63	2.67	1.70	4.14	1.31	0.33	6.09
3.75	2.55	1.63	4.17	1.28	0.32	6.10
3.88	2.43	1.56	4.20	1.25	0.31	6.10
4.00	2.32	1.50	4.23	1.22	0.31	6.11
4.13	2.22	1.43	4.26	1.19	0.30	6.12
4.25	2.12	1.37	4.29	1.16	0.29	6.12
4.38	2.03	1.31	4.32	1.13	0.28	6.13
4.50	1.94	1.26	4.34	1.10	0.28	6.13
4.63	1.85	1.20	4.37	1.08	0.27	6.14
4.75	1.77	1.15	4.39	1.05	0.27	6.14
4.88	1.69	1.10	4.41	1.03	0.26	6.15
5.00	1.61	1.06	4.43	1.00	0.25	6.15

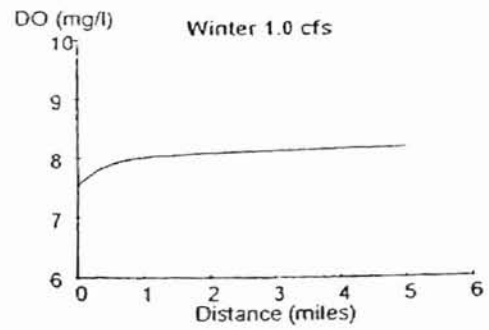
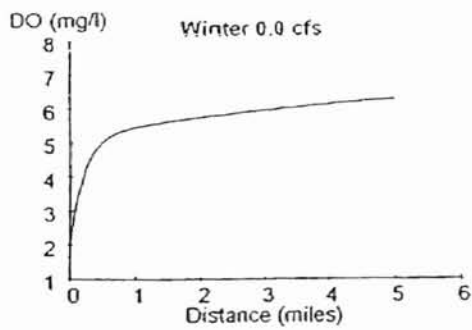
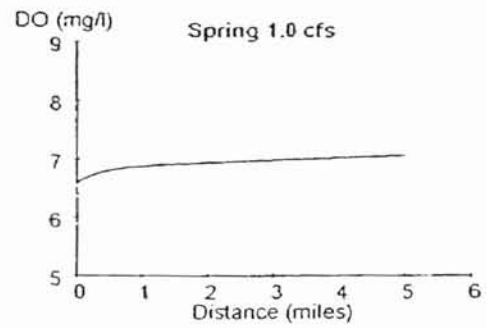
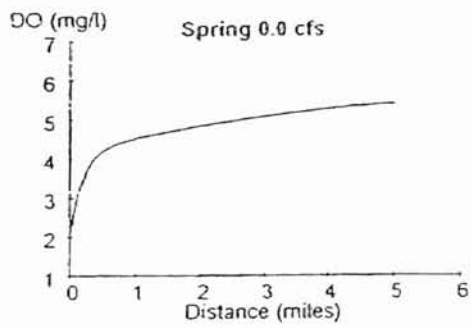
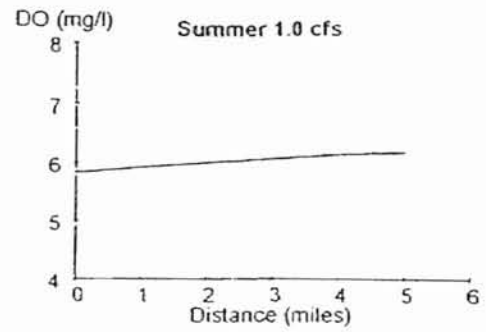
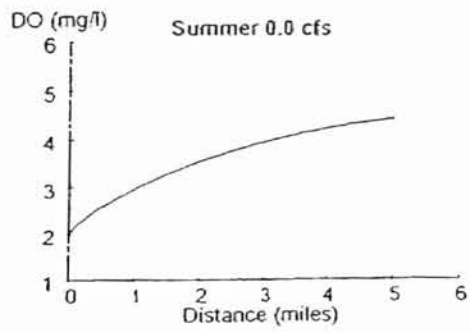
SPRING

Distance (miles)	0.0 cfs			1.0 cfs		
	CBOD5 (mg/l)	NH3-N (mg/l)	D.O. (mg/l)	CBOD5 (mg/l)	NH3-N (mg/l)	D.O. (mg/l)
0.00	15.00	8.00	2.00	3.10	0.82	6.59
0.13	14.51	7.79	3.10	3.05	0.81	6.67
0.25	14.03	7.59	3.68	2.99	0.79	6.73
0.38	13.57	7.39	4.01	2.94	0.78	6.78
0.50	13.13	7.20	4.19	2.89	0.77	6.81
0.63	12.70	7.01	4.31	2.84	0.76	6.84
0.75	12.28	6.82	4.39	2.79	0.75	6.86
0.88	11.88	6.65	4.46	2.74	0.74	6.87
1.00	11.49	6.47	4.51	2.69	0.73	6.88
1.13	11.11	6.30	4.56	2.64	0.72	6.90
1.25	10.75	6.14	4.61	2.59	0.71	6.90
1.38	10.39	5.98	4.65	2.55	0.70	6.91
1.50	10.05	5.82	4.69	2.50	0.69	6.92
1.63	9.72	5.67	4.73	2.46	0.68	6.93
1.75	9.40	5.52	4.77	2.41	0.67	6.94
1.88	9.09	5.38	4.81	2.37	0.66	6.94
2.00	8.80	5.23	4.84	2.33	0.65	6.95
2.13	8.51	5.10	4.88	2.29	0.64	6.96
2.25	8.23	4.96	4.91	2.25	0.63	6.96
2.38	7.95	4.83	4.94	2.20	0.62	6.97
2.50	7.70	4.71	4.97	2.17	0.61	6.97
2.63	7.44	4.59	5.00	2.13	0.61	6.98
2.75	7.20	4.47	5.03	2.09	0.60	6.99
2.88	6.96	4.35	5.06	2.05	0.59	6.99
3.00	6.74	4.23	5.09	2.02	0.58	7.00
3.13	6.51	4.12	5.11	1.98	0.57	7.00
3.25	6.30	4.02	5.14	1.94	0.56	7.01
3.38	6.09	3.91	5.16	1.91	0.56	7.01
3.50	5.89	3.81	5.19	1.87	0.55	7.02
3.63	5.70	3.71	5.21	1.84	0.54	7.02
3.75	5.51	3.61	5.23	1.81	0.53	7.03
3.88	5.33	3.52	5.26	1.78	0.52	7.03
4.00	5.16	3.43	5.28	1.74	0.52	7.04
4.13	4.99	3.34	5.30	1.71	0.51	7.04
4.25	4.82	3.25	5.32	1.68	0.50	7.05
4.38	4.67	3.16	5.34	1.65	0.50	7.05
4.50	4.51	3.08	5.35	1.62	0.49	7.06
4.63	4.37	3.00	5.37	1.59	0.48	7.06
4.75	4.22	2.92	5.39	1.57	0.47	7.07
4.88	4.08	2.85	5.41	1.54	0.47	7.07
5.00	3.95	2.77	5.42	1.51	0.46	7.07

WINTER

Distance (miles)	0.0 cfs			1.0 cfs		
	CBOD5 (mg/l)	NH3-N (mg/l)	D.O. (mg/l)	CBOD5 (mg/l)	NH3-N (mg/l)	D.O. (mg/l)
0.00	18.00	12.00	2.00	3.36	1.16	7.52
0.13	17.57	11.81	3.43	3.32	1.15	7.66
0.25	17.14	11.62	4.26	3.27	1.14	7.76
0.38	16.73	11.43	4.73	3.23	1.13	7.84
0.50	16.32	11.25	5.02	3.19	1.12	7.90
0.63	15.93	11.07	5.19	3.15	1.11	7.94
0.75	15.54	10.89	5.31	3.10	1.10	7.97
0.88	15.17	10.71	5.39	3.06	1.09	8.00
1.00	14.80	10.54	5.45	3.02	1.08	8.02
1.13	14.45	10.37	5.50	2.98	1.07	8.03
1.25	14.10	10.21	5.54	2.94	1.06	8.05
1.38	13.76	10.04	5.58	2.91	1.05	8.06
1.50	13.42	9.88	5.61	2.87	1.04	8.07
1.63	13.10	9.72	5.65	2.83	1.03	8.07
1.75	12.78	9.57	5.68	2.79	1.02	8.08
1.88	12.47	9.41	5.71	2.76	1.01	8.09
2.00	12.17	9.26	5.74	2.72	1.01	8.09
2.13	11.88	9.11	5.77	2.68	1.00	8.10
2.25	11.59	8.97	5.80	2.65	0.99	8.10
2.38	11.31	8.82	5.83	2.61	0.98	8.11
2.50	11.04	8.68	5.86	2.58	0.97	8.11
2.63	10.77	8.54	5.88	2.55	0.96	8.12
2.75	10.51	8.40	5.91	2.51	0.95	8.12
2.88	10.26	8.27	5.93	2.48	0.95	8.13
3.00	10.01	8.14	5.96	2.45	0.94	8.13
3.13	9.77	8.01	5.98	2.42	0.93	8.13
3.25	9.53	7.88	6.01	2.38	0.92	8.14
3.38	9.30	7.75	6.03	2.35	0.91	8.14
3.50	9.08	7.63	6.05	2.32	0.91	8.15
3.63	8.86	7.50	6.08	2.29	0.90	8.15
3.75	8.65	7.38	6.10	2.26	0.89	8.15
3.88	8.44	7.27	6.12	2.23	0.88	8.16
4.00	8.23	7.15	6.14	2.20	0.87	8.16
4.13	8.03	7.03	6.16	2.17	0.87	8.17
4.25	7.84	6.92	6.18	2.14	0.86	8.17
4.38	7.65	6.81	6.20	2.12	0.85	8.17
4.50	7.47	6.70	6.22	2.09	0.84	8.18
4.63	7.29	6.59	6.23	2.06	0.84	8.18
4.75	7.11	6.49	6.25	2.03	0.83	8.18
4.88	6.94	6.38	6.27	2.01	0.82	8.19
5.00	6.77	6.28	6.29	1.98	0.82	8.19

Receiving Stream Dissolved Oxygen Profile



JUSTIFICATION FOR SELECTION OF INPUT PARAMETERS

CITY: **City of Clinton** (Original Excel Model)
 PERFORMED BY: **John Doe**
 DATE: **4/28/99**

I DISCHARGE INFORMATION

LOCATION: **SE¼/NW¼/SE¼ - S18 - T11N - R10W** BASIN: **410200**
 COUNTY: **Custer County, Oklahoma**

DESIGN FLOW: **0.060 MGD**

II RECEIVING STREAM: **Unnamed Tributary to Washita River**

	7Q2=	0.00	MGD	4.30	MILES
INTERMITTENT STREAM					
NUMBER OF SEGMENTS		40			
STREAM SLOPE (S) =		5.100	FT/MILE		
SIDE SLOPE (P) =		0.100	FT/FT		
MANNING'S N =		0.100			
VELOCITY COEFFICIENT Cv =		2.207			
DEPTH COEFFICIENT Ch =		0.163			
1.0 CFS	VELOCITY =	3.73	MI/DAY	0.23	FPS
	DEPTH =	0.11	METERS	0.35	FT
0.0 CFS	VELOCITY =	2.01	MI/DAY	0.123	FPS
	DEPTH =	0.04	METERS	0.14	FT

III WATER QUALITY CRITERIA OF RECEIVING STREAM

WARM WATER AQUATIC COMMUNITY
 AVERAGE D.O. REQUIREMENT

SUMMER	5.00 MG/L
SPRING	6.00 MG/L
WINTER	6.00 MG/L

IV UPSTREAM CONDITIONS

ANY POINT/NPS POLLUTION SOURCES?
 D.O. SATURATION 85.00 %
 UPSTREAM CBOD5 2.00 MG/L
 UPSTREAM NH3-N 0.15 MG/L

V RATE CONSTANTS at 20° C

	<u>FLOW</u>	<u>1.0 CFS</u>	<u>0.0 CFS</u>
<u>CBOD DECAY RATE (K1)</u> (0.2*(H/8) ^{1.434})	K1	0.40 /DAY	0.40 /DAY
<u>REAERATION RATES (K2)</u>			
TSIVOGLU K2=CVS; C: 1.8		2.09 /DAY	1.13 /DAY
TURNERY-HARRIS K2=1.33*S ^{0.32} /r ^{0.64}		9.78 /DAY	9.78 /DAY
TEXAS K2=4.022*V ^{0.273} /H ^{0.894}		6.93 /DAY	13.38 /DAY
OWENS ET AL K2=21.7*V ^{0.67} /H ^{1.85}		57.22 /DAY	209.41 /DAY
K2 SELECTED TURNERY-HARRIS	K2	9.78 /DAY	9.78 /DAY
<u>CBOD SETTLING RATE</u>	KS	0.03	0.03 /DAY
<u>NBOD DECAY</u> Sediment	KN	0.30	0.30 /DAY

SEDIMENT OXYGEN DEMAND

SOD	0.100	0.075	0.060	G/FT ² /D
-----	-------	-------	-------	----------------------

VII PROPOSED WASTELOAD ALLOCATIONS (WLA)

	EFFLUENT				MINIMUM	MINIMUM
	CBOD5	NH3-N	D.O.	TEMP	D.O.	D.O.
	(MG/L)	(MG/L)	(MG/L)	(°C)	0.00 CFS	1.0 CFS
SUMMER	10.0	6.0	2.0	32	2.00 MG/L	5.84 MG/L
SPRING	15.0	8.0	2.0	25	2.00 MG/L	6.59 MG/L
WINTER	18.0	12.0	2.0	18	2.00 MG/L	7.52 MG/L

Calculations of Safety Factor

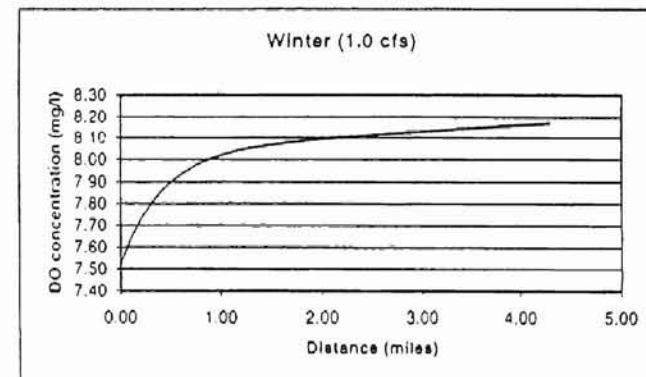
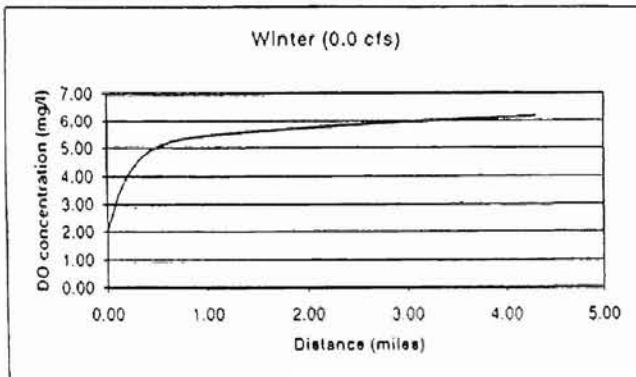
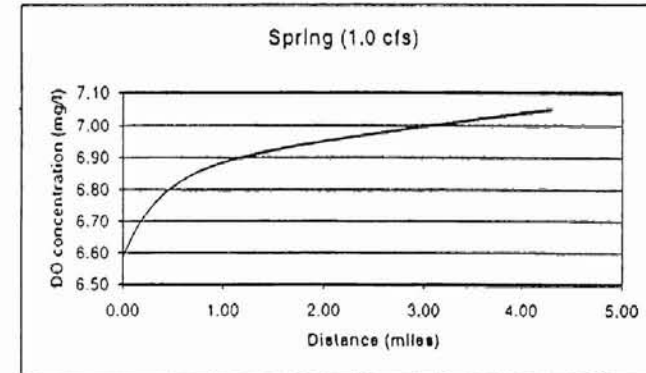
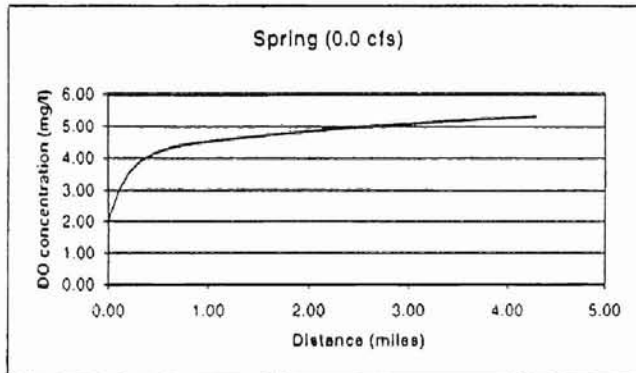
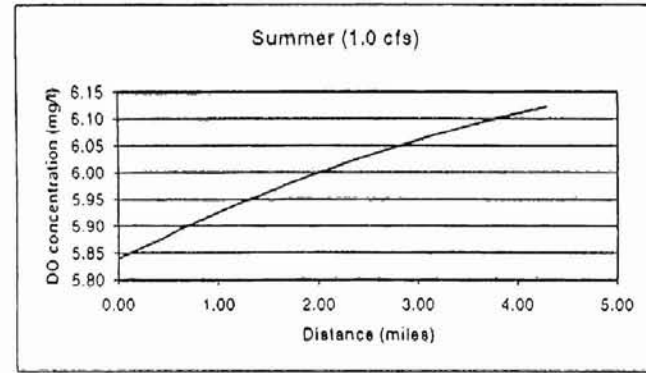
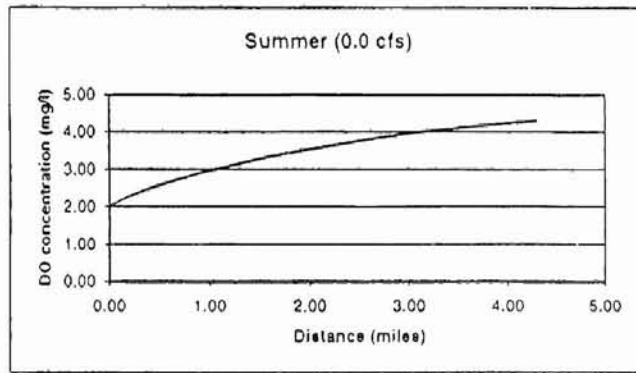
	CBOD5 (MG/L)	NH3-N (MG/L)	D.O. (MG/L)	Factor	Safety Factor
SUMMER	10	6	2.0	5.85	82.9%
SPRING	15	8	2.0	5.12	80.5%
WINTER	18	12	2.0	9.93	89.9%

Waste Load Allocations

SEASON	Maximum Loading (lb/day)		Allocated Loading (lb/day)		Reserve Loading (lb/day)		Reserve/Max (%)
	CBOD5	NH3 - N	CBOD5	NH3 - N	CBOD5	NH3 - N	
SUMMER	29.3	17.6	5.0	3.0	24.3	14.6	82.9%
SPRING	38.4	20.5	7.5	4.0	30.9	16.5	80.5%
WINTER	89.4	59.6	9.0	6.0	80.4	53.6	89.9%

Locations of D.O. Sags

	MINIMUM	RIVER	MINIMUM	RIVER
	D.O.	MILE	D.O.	MILE
	0.0 CFS		1.0 CFS	
SUMMER	2.00 MG/L	0.00	5.84 MG/L	0.00
SPRING	2.00 MG/L	0.00	6.59 MG/L	0.00
WINTER	2.00 MG/L	0.00	7.52 MG/L	0.00



VITA

Zhengping Yan

Candidate for the Degree of

Master of Science

Thesis: DEVELOPING A WINDOWS GUI APPLICATION FOR A WATER QUALITY
MODEL WITH VISUAL BASIC 6

Major Field: Computer Science

Biographical:

Personal Data: Born in Yunnan, China, on March 12, 1965.

Education: Graduate from Qiujin First High School, Yunnan, China in 1983; received a Bachelor of Science degree in Civil Engineering from Tianjin University, Tianjin, China in July 1987. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December, 2000.

Experience: Raised on Qiujin City, Yunnan; employed by Kunming Hydroelectric and Investigation Institute as an engineer, Kunming, Yunnan, 1987 to 1993; employed by Oklahoma State University, Department of Computer Science as a graduate teaching assistant, 1998 to 1999; employed by Wal-Mart Stores, Incorporation as a programmer, Wal-Mart Stores, Inc. Home Office, Information System Division, 1999 to present.