

---

A SOFTWARE ENVIRONMENT FOR  
OIL & NATURAL GAS FIELD  
MANAGEMENT

By

XIYUAN CHEN

Bachelor of Science

Nankai University

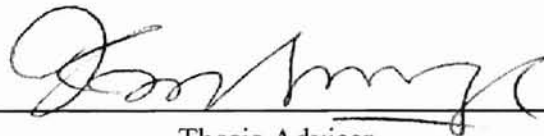
Tianjin, China

1988

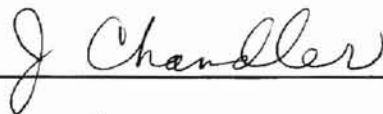
Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
May, 1997

A SOFTWARE ENVIRONMENT FOR  
OIL & NATURAL GAS FIELD  
MANAGEMENT

Thesis Approved:



Thesis Adviser



Dean of the Graduate College

## ACKNOWLEDGMENTS

I wish to express sincere and special thanks to my advisor, Dr. K.M. George, for his encouragement and expert guidance during the design and development of my thesis project. I would like also like to thank Dr. John P. Chandler and Dr. Huizhu Lu for their advice and encouragement.

I would also like to take this opportunity to show my appreciation to Mr. Xuehai Li and Mr. Shenghong Yao for their help, guidance, and friendship.

## TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION .....	1
II. LITERATURE REVIEW .....	3
Object-Oriented Programming .....	3
Event-Driven Programming .....	5
Object-Oriented Database Management System .....	8
Graphical User Interface .....	9
Microsoft Visual C++ .....	11
Graphic User Interface Applications .....	12
An Introduction of Oil and Natural Gas Field .....	14
III. PROBLEM STATEMENT AND PROGRAM DESIGN .....	19
IV. USER INTERFACE .....	34
VI. SUMMARY AND CONCLUSION .....	50
REFERENCES .....	51

## LIST OF FIGURES

Figure	Page
1. Inheritance	5
2. OOP database application design	9
3. Internal structure of the project program	23
4. System diagram	24
5. Structure of program implementation	35
6. The main window of the application for oil and natural gas	36
7. About Project Dialog box	36
8. The pop-up menu items in 'File' option	37
9. Open dialog box	38
10. The field level management window showing the information	39
11. The lease level management window showing the information	40
12. The well child window	41
13. The pop-up menu items in 'Information' option	42
14. The General information dialog box	43
15. The pop-up menu items in 'Daily Data' option	44
16. The Search data dialog box	45

17. The Input data dialog box	46
18. Success dialog box	46
19. The pop-up menu tiems in ‘Analysis’ option	47
20. The rate dialog box	48
21. The life dialog box	48
22. The history graph dialog box	49

## LIST OF TABLES

Table	Page
1. List of files and classes in the project	21
2. Explanation of parameters	22
3. Input Data file	25
4. NohandApp Class	26
5. CAboutDlg Class	26
6. Field Class	27
7. CMainFrame Class	28
8. Lease Class	28
9. Infor Class	29
10. NohandDoc Class	29
11. BlastitDlg Class	30
12. Inset Class	31
13. NohandView Class	32
14. Well Class	32
15. Analysis Class	33

## CHAPTER I

### INTRODUCTION

Information related to daily production of oil and natural gas fields are very important to oil companies. Several analyses are performed on this data. They include calculation of the field's life, rate of declines, and so on. A software system that manages the data is a valuable tool for the oil industry. This thesis addresses the problem of developing such a software system. Specifically, a database program with analysis capability is implemented. This program is a very useful tool for production engineers and non-professional users in oil and gas fields. This program is implemented using Visual C++. It provides a user friendly graphical user interface(GUI). It is designed for use on an IBM-compatible personal computer.

This thesis is organized as follows:

Chapter I, the current chapter, is a brief introduction to the project. In chapter II, the author reviews object-oriented programming, event-driven programming, object-oriented database management system, windows environment, describes Microsoft Visual C++ as a tool for graphic use interface creation, and the applications of Visual C++. An introduction of oil and natural gas field is also provided. In chapter III, the author states the objective for creating database management system for oil and gas field, describes the internal structure of the program in detail, and provides the design of the program. And



Chapter IV is devoted to the description of the GUI implemented by the program.

Chapter V provides a summary of the project and concludes this thesis.

## CHAPTER II

### LITERATURE REVIEW

#### Object-Oriented Programming

A programming methodology that supports classes, instances, and message passing is referred to as object-based programming, and which one that also supports inheritance is referred to as object-oriented programming (OOP). The fundamental idea of object-oriented programming is to combine data and functions into a single unit called an object or block. According to Yourdon, object-oriented techniques are the most important development since the introduction of structured techniques during the 1970s and 1980s [Yourdon, 1994]. The origin of OOP can be traced back to Simula in 1967. In [Timothy, 1991], Timothy states that this new technique will be the key to increased productivity, and improved reliability. During the mid 1980s it started to come into general use [Florentin, 1991]. OOP is a style of programming that models data in terms of real world objects, it is a combination and normalization of ideas that have been around for many years [Hu, 1990]. OOP is a new way of thinking about what it means to perform computation, about how programmer can structure information inside a computer. OOP is often referred to as a new programming paradigm. Some other programming paradigms are the imperative-programming paradigms ( Pascal language or C language support this paradigm), the logic-programming paradigm ( Prolog ), and the functional-programming paradigm ( FP or ML ) [Timothy, 1991].

---

The major characteristics of object-oriented programming are [Ananthaswamy, 1995]:

- **Abstraction:** Abstraction is a process that stresses the main features while hiding the unnecessary details. Data abstraction and procedural abstraction are two types of abstractions that are commonly used. A powerful mechanism to combine the two types are supported by object oriented languages.
- **Data encapsulation:** A mechanism that supports programmer to package together an object's data or attributes, and the functions that operate on the data such as methods. Implementation details of the operations that manipulate the stored data can be changed without affecting the interface, and they are independent of how the data is used, so programmer can "hide" the implementation of an object.
- **Inheritance:** Inheritance allows the creation of a new class by using an existing class as a base. It is a mechanism that allows an object to incorporate all or part of the definition of another object as part of its own definition. Inheritance supports the reusability of code. This is shown in Figure 1 [Huang, 1995].
- **Polymorphism:** A polymorphic function or operator can be used with many data types. Polymorphism is used to reduce a large number of code [Marcus, 1995]. This allows a program to provide the same interface to different objects. Combined with data abstraction, polymorphism gives more control to the programmer than the only use of an abstraction.

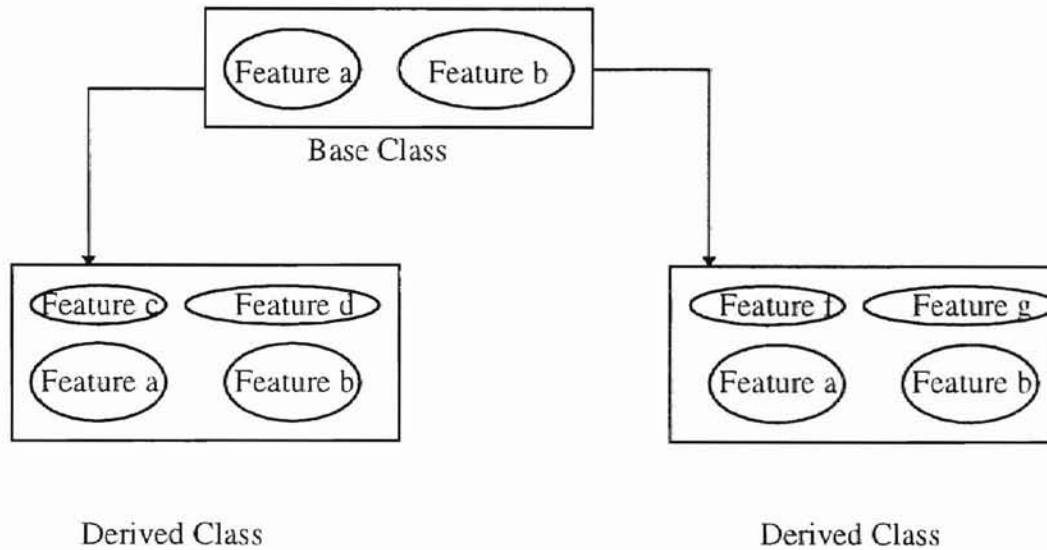


Figure-1 Inheritance.

- Modularity: A programmer can better manage a large complex system by breaking up the system into modules. Two types of modularity are provided by OOP -- class level modularity and file level modularity. Class level modularity brings the inherent advantages of data abstraction and encapsulation. Since the development can proceed on a class-by-class basis to a large extent, development of individual classes with well-defined interface provides a convenient method for implementing large software systems.

### Event-Driven Programming

What is an Event? According to the Oxford Dictionary, an event is a thing that happens or takes place, especially one of importance. How are events generated? Clicking the mouse generates a mouse event, pressing a key generates a keystroke event in fact, most everything the user does generates an event. Events may be generated by

devices or other external components. A user may define his/her own events depending on the application and use in programs.

In a traditional application, execution begins at the top of the program, then flows through function calls and control-flow statements. The behavior of the program is predictable. One may say that the program is in control, while the user plays a subordinate role of entering keystrokes when requested.

In an event-driven program, on the other hand, the user and external events control what parts of the program are executed. The user generates events using a mouse or other devices, and the program is expected to make an appropriate response to each click. An effective way of dealing with these events is to associate callback functions with events. A callback function is called automatically when an event occurs. Internal functions used for event-handling include [<http://www.cs.princeton.edu/courses/cs111/>]:

- `event_get`: Get the next event. If no events are available, wait for the next event.
- `event_process`: Get the next event and call the appropriate callback function. Return immediately if no events are available.
- `event_loop`: Set up an infinite loop to automatically process the event queue, calling the appropriate callback function for each event.
- `event_queue`: Queue an event for an object. The event may be placed either at the end of the queue or the head of the queue.
- `event_delete`: Delete an event or events from the event queue.
- `event_peek`: Look for a particular event or events in the event queue, without affecting the event queue.

There are five types of event-driven programming models[<http://www.sb.com.au/wayne>]: Character, Traditional, Hybrid, Callback (Object-based), and Callback (Object-oriented). Models are mostly used for simple applications or when converting traditional applications to a GUI environment. Callback methods are attached to events. User-defined classes are derived from pre-defined classes,so it is the most suitable for the vast majority of non-trivial GUI applications[<http://www.sb.com.au/wayne>].

An event-driven program is one in which the main components of the interaction with users(and with the system) are driven by a messaging system that provides messages, known as events to the program. The biggest problem with traditional programming, is as they grow linearly in complexity, they grow exponentially in size: the more complex they are, the bigger they are [Leavens, 1994]. This also tends to exacerbate the problem of bugs and maintenance. For an event-driven program, there is little growth of the global state, and the state is contained within each object instead of being scattered through the code in the form of global variables, so complexity is reduced. Also maintenance is easier for event-driven programs because all message-passing is centralized and rarely needs to be changed. There is another big difference between event-driven programs and traditional programs: an event-driven program is ready at any time to process any sort of event. but a traditional program is not [Leavens, 1994]. Because the event-driven interface can respond to events at any given time, it makes developing a program with this interface much simpler in the case where user interaction is required.

## Object-Oriented Database Management System

A database consists of some collection of persistent data (operational data and decision support data) that is used by application systems of some given enterprise.

A database system involves four major components [Henry & Abraham, 1991]: (1) Data (information), (2) Hardware, (3) Software: between the physical database and the user is a layer of software, called database management system(DBMS), and (4) User. Object-Oriented Databases are databases that support objects and classes. They are different from the more traditional relational databases because they allow structured sub-objects, each object having its own identity or object-id [Huang, 1995]. In addition, they provide support for object-oriented features such as methods and inheritance. It is also possible to provide relational operations on an object-oriented database. Rather than providing only a high-level language such as SQL for data manipulation, an ODBMS transparently integrates database capabilities with the application programming language. Briefly, OODM in contrast to traditional data models supports [Henry & Abraham, 1991]:

- Object identity.
- Constructors for complex values, which can be applied recursively and in arbitrary combinations without further restrictions. These constructors allow users to define operation for objects, and arbitrary classes.
- Object structures and complex objects.
- Class hierarchies with inheritance.

Hence, compared with traditional database system, OODM is more powerful and comfortable to users. An OOP database application design is shown as figure 2.

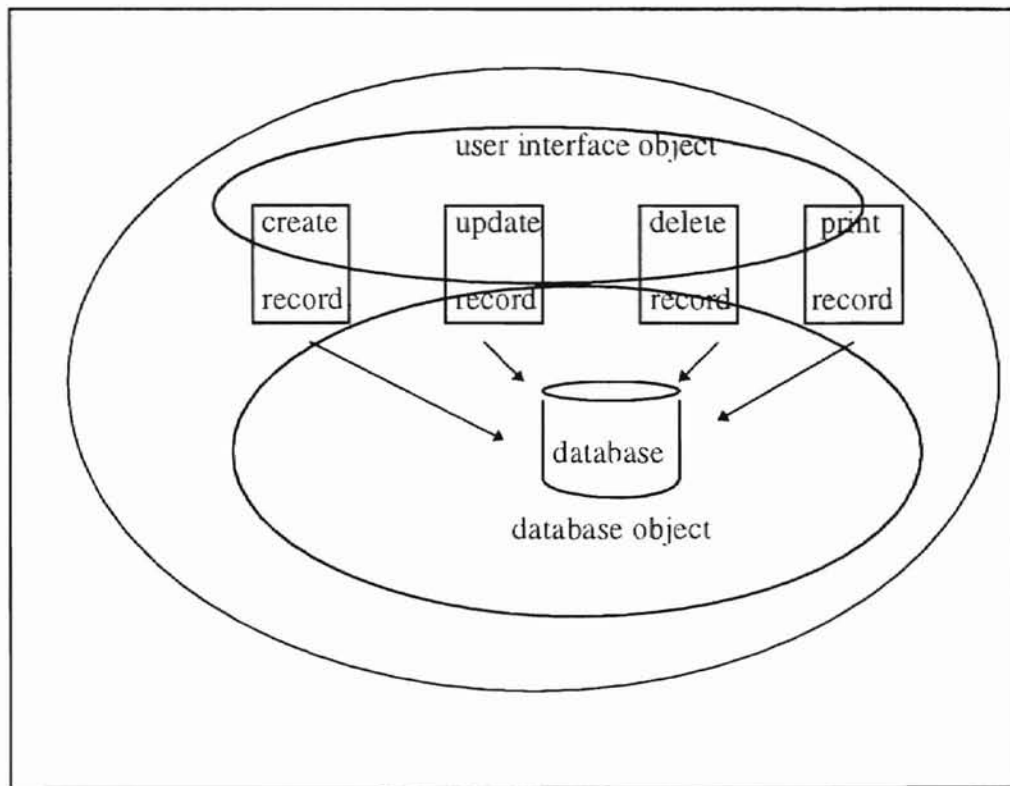


Figure-2 OOP Database Application Design.

(adopted from [Halladay, 1993] )

### Graphical User Interface (GUI)

A user interface is a software by which an application communicates with the user, and the user with the application. The most important principle is putting the user in control. The more the user feels in control, the more the user is comfortable and satisfied with the application. Graphical User Interface (GUI) allows users to see environment in a visual way. In particular, windowed interfaces provide more feedback about what the user can (and cannot) do [Leavens, 1994]. As Aaron Marcus points out "the GUIs reduce the user's information load in the following ways:



- Present commands, options, or data to the user on the appropriate application display.
- Display information appropriate to completing a task on the screen, so the user can be selective in attending to information relevant to his or her needs.
- Organize information in a meaningful way to help the user focus on essential task information. It provides immediate feedback, the error message should not imply that the user is at fault, but state the problem and offer possible solution, so the user feels free in control, should be able to explore without fear of causing irreversible mistake. This makes the decision-making process easier as well as reducing the potential of errors”[Marcus, 1995].

Interaction is the means by which the user controls the execution of an application. The concept of pointing to an object and then selecting it, often referred to as point-and-select, is an essential factor in achieving effective human application interaction [Marcus, 1995]. The interaction is achieved by the use of the keyboard or mouse. A GUI supports point-and-select interaction.

Almost every program for windows uses a dialog box to interact with the user. A dialog box is truly a window that receives message, which can be moved and closed, and can accept drawing instructions in its client area. The benefit of GUI is that it provides a user friendly programming environment. The users use a group of windows on screen, each of which contains standard controls (such as buttons, menus, scroll bars and list). Icons and objects can be easily manipulated with the mouse or the keyboard [Andrew, 1993]. So users, even with little knowledge of application area, can obtain an analysis of results by simply the use of mouse or keyboard. To create a GUI program, the programmer first designs all of the needed user interface controls into a window. As the user manipu-

lates the application controls, the program responds appropriately by sending and receiving messages.

### Microsoft Visual C++

Visual OOP is a programming paradigm that combines both objected-oriented and visual programming techniques [Burnett, 1995]. It improves the quality and accessibility of information exchange between programmers and computer, while the same time supporting programmers who solve large and complex problem. Visual C++ is a programming language that combines OOP and visual programming techniques. It is a complete application development environment, lets the programmer fully exploit the object oriented nature of C++ to create professional windows applications, it contains the most powerful window-based application framework to date [Kruglinski, 1995]. The Microsoft foundation class library (MFC) contains a library of C++ classes and global functions along with their source codes, it is designed mainly for windows applications [Kruglinski, 1995]. The document-view architecture is defined in Microsoft Visual C++. Document objects which are created by the document template objects manage the application's data. The base class for all the application specific document is CDocument class. Each application can derive it's own document class from CDocument class. A single document interface application (SDI) allows only one document window to be opened in the same instance of an application. A multiple document interface (MDI) application is a kind of application that allows multiple document frame window to be opened in the same instance of an application. It contains several MDI child windows. each child window contains a specific document. View represents the client area of a

frame window which is used to display, accept, edit or select the input for a document. The base class for all the application specific view is CView class. There are four types: CView, CScrollView, CFormView, and CEditView. Data are stored in document and displayed in view. Document and view communicate with each other by sending and receiving windows messages [Andrew,1993]. The components of Visual C++ also include[Kruglinkski, 1995]:

- AppWizard: AppWizard is a code generator which creates a working skeleton of a windows application with features, class names, and source code filenames that are specified through dialog boxes. The AppWizard approach is document and view oriented.
- ClassWizard: ClassWizard is a program that operates both inside the Visual Workbench and inside AppStudio, it is implemented as dynamic link libraries.
- AppStudio: AppStudio is a resource editor. Resources include accelerators, bitmaps, cursors, dialog boxes, icons, menus, version, and string table.
- Visual Workbench: windows-hosted interactive development environment that's a direct descendant of Microsoft QuickC for windows.
- C/C++ compiler and linker [Kruglinski, 1995].

### Graphical User Interface Applications

In recent years there had been a great deal of interest in the use of GUI for various applications. As Henry Lieberman says "graphical is about visually representing the world and visually representing our ideas" [ Henry, 1996]. And Rodey Bell says "GUIs are now

the norm for interactive applications and can be a key distinguishing characteristic” [Rodey, David, 1995]. The GUI, along with being a good medium of communication between user and application, also has another benefit which the character-based text does not have --Graphics. It provides an easy to use, intuitive, and consistent user environment. The graphical methods enable the data analyst to explore data thoroughly, to look for patterns and relationships, to confirm or disprove the expected, and to discover new phenomena. It is clear that displaying message in a dialog box gives more convenience than displaying text in a terminal, because dialog boxes provide scrolling. So by using GUI, the user can quickly and efficiently enter and retrieve data from databases, and do analysis. By using a sound graphical interface and user friendly environment, user can represent the spatial data structure [ Voisard, 1991]. Several research related to GUI are described below:

Stephen G. Eick points out that it is a hard, tedious, time-consuming, and error-prone task to analyze the Read Only Print (ROP) to determine which messages correspond to real problems, what the root causes are and their priority for resolution. Analyzing ROPs is a problem of size. By using graphical tool, user can quickly obtain insights into the problems and can characterize the patterns, decreasing analysis time 80% , and increasing analysis efficiency [Stephen, 1994].

An object design framework and methodology, which utilizes the object-oriented concepts, for planning, organizing and designing structural engineering design objects has been recently reported. The application domain considered is structural engineering, specifically the design of reinforced concrete buildings. The elements are

classified into object classes, with detailed identification of their properties, behavior, relationship and constraints. And a well-defined communications channel is identified. A framework for object design is presented [Jamal, 1995].

Educational programs are often used by school children, and teachers. For example teachers design math worksheets and students enter answers through a graphic editor that recognizes shapes, handwriting, and gestures [Takayuki, 1995]. As Zhang and Chen pointed out that GUI is a drawing tools and is used in several educational programs [Zhang 1996, and Chen 1995].

In geologic area, for example, GUI can be used in the generation of two dimensional gridded models [Dean, Brett, 1996]. Also, it can be used to modular organization and manipulation of various types of data when used with the numerical models [Michel, 1995].

Visual C++ programming technique is so powerful and convenient for users, it has been used widely in several research fields. Computer application in petroleum industry data management is one of the areas.

#### An Introduction of Oil And Natural Gas Field

Petroleum deposits have been known since antiquity and have been studied for at least two centuries. The word "petroleum" appears to have been used with some frequency by Agricola as early as 1546 [Landon, 1996]. As the global economy grows, petroleum is widely used, and world wide interest in oil technology is also increased [Paul, 1994].

It is generally agreed that oil usually is not formed in the rocks from which it is obtained, but somehow migrates into the reservoir rocks [ Speight, 1980]. After a field has been discovered, exploratory wells and delineation wells are drilled to evaluate the field for reserves and design criteria. After the actual drilling of development wells, production engineers need to initiate, recommend, and monitor these wells while they are producing oil or gas [Landon, 1996].

We can obtain lots of data from a well, such as the daily production of oil, natural gas, and water, the daily injection of water and gas, the daily fluctuation of the liquid gas level in a well, and the monthly pressure readings [Troutman, 1958]. By the use of these data, geologists and production engineers can identify potential source rocks and evaluate the maturation, generation, and migration of oil [Arps, 1994]. Also they can map the geological structures together with the development history of an oil field. For example, the actual composition of the oil obtained from the well is variable and depends not only on the original composition of the oil in situ but also on the manner of production and the stage reached in the life of the well or reservoir. As Robert London says “For a newly opened formation and under ideal condition the productions of gas may be so high that the oil is, in fact, a solution of liquid in gas which leaves the reservoir rock so that a rock sample will not show any obvious oil content” [Landon, 1996]. So by calculating the actual composition, engineers can know the stage of life for the well. Engineers calculate the effective decline rate, convert an effective decline rate to a nominal rate, estimate the flow rate for a well after a period of production. They also calculate time economic limit, total volume of oil or gas produced during the life of the well [Landon, 1996]. These data

are very useful for purchase of commercial project. The formulas used for these calculation are given below[Landon, 1996]:

$$d = (qt - qt') / qt \quad [ 2-1 ]$$

where  $d$  is effective decline rate,  
 $qt$  is flow rate at any point in time, and  
 $qt'$  is flow rate at some time later.

$$a = - \ln(1-d) \quad [ 2-2 ]$$

where  $a$  is nominal decline rate, and  
 $d$  is effective decline rate.

$$Q = (q_0 - q_c) / a \quad [ 2-3 ]$$

where  $Q$  is reserve,  
 $q_0$  is initial flow rate,  
 $a$  is nominal decline rate, and  
 $q_c$  is economic limit flow rate equivalent.

$$t = ( \ln q_0 - \ln q_c ) / a \quad [ 2-4 ]$$

where  $t$  is time to economic limit,  
 $q_c$  is economic limit flow rate equivalent,  
 $a$  is nominal decline rate, and  
 $q_0$  is initial flow rate.

After a discovery has been made, a company has a number of options. One is to sell the field to another company before it is developed. For the business people, there are generally many different investment opportunities to choose from, and the economic

evaluation is a decision making tool to help management determine the best choice. When they are buying or selling oil field, they need to compare fields, well's production information, storage life, location and other information. The petroleum industry is characterized by large front-end capital investments. In order to make good investment decisions, it is important that the time value of money be considered so that profit-to-investment ratios can be properly evaluated. And most individual wells will decline along a curve that can be modeled by hyperbolic decline curve, like the one shown below [Landon,1996].

$$q_t = q_0 / [(1 + (a_0 * t/h)) ^ h] \quad [2-5]$$

where  $q_t$  is flow rate at any given time,  
 $q_0$  is initial production rate,  
 $a_0$  is initial nominal decline rate,  
 $t$  is production time, and  
 $h$  is hyperbolic factor.

As  $h$  goes to infinitely, the exponential decline equation is becomes

$$q_t = q_0 * e^{(-at)} \quad [2-6]$$

In the case of an oil field, most data are on the log books before 1980, therefore, lots of time is needed to collect, classify and calculate these data for any project. Since about 1960, geologists have been in the forefront of computer data processing. They were among the first to take advantage of the computer's ability to do complex mathematical processing of large data files. By the mid-1950s, petroleum companies were considering computer-oriented geologic data files. Commercial data files began in



1957 with the foundation of the Petroleum Information Corporation [Robinson, 1982]. The oil fields data base is the foundation of information available to the computer for retrieval and computing. But calculations in oil field, involve large numbers of operations which if attempted by hand are inherently very tedious and error-prone [Dawson, 1976]. So far, there are numerous data files available for project use, but file designers did not want their data contaminated by the data from other files, so there is no file to file compatibility. Most programs which are being used so far for data analysis are written in high-level programming languages, such as BASIC, COBOL, FORTRAN II, IV, and V [Nobles, 1994]. These programs are very inconvenient to use and not friendly to people who do not have sufficient background in computer and petroleum technology.

## CHAPTER III

### PROBLEM STATEMENT AND PROGRAM DESIGN

As described in chapter II, object-oriented programming (OOP) is a style of programming that models data in terms of real world objects, the major contributions of OOP are inheritance, encapsulation, and polymorphism. Object-Oriented Database Management System is one of the major application areas of OO concept. And OOP is one of the best ways to build and maintain a GUI application.

We use the Object-oriented method to develop a Window-based program to create an oil and gas field database and to analyze well decline rate automatically. This program is implemented using Microsoft Visual C++ as a Visual C++ project. It provides graphical user interface (GUI) to do data analysis. Users can also select display of daily oil, natural gas, and water production, and daily natural gas or water injection data information. Field rate information could be viewed as graphs. By this way, people without computer programming and petroleum knowledge can use this program easily.

To develop this project, first the author uses AppWizard to create the C++ Microsoft Foundation Classes Library source files for the project. Then AppStudio is used to create and edit resource files: ClassWizard is used to add C++ framework code for classes and message maps for document and view classes.

The list of files and classes in the project is shown in Table 1. The internal structure of the software is shown in Figure 2, it gives the document and view architecture information and class relation information about the program. The system diagram is shown in Figure 3, it gives the user interface information about this system. Input data is stored in a file named data.txt. It contains the field name, location, investment, and also contains the lease name, well name, discovery date, and well daily information, such as oil production, gas production, water production, gas injection, water injection, and liquid level. The production is in barrels per day. The explanation of these parameters is provided in Table2. As sample input data is shown in Table3.

Header File	Source Code File	Class	Description
field.h	field.cpp	Field	Field dialog box
lease.h	lease.cpp	Lease	Lease dialog box
well.h	well.cpp	Well	Well dialog box
NOhand.h	NOhand.cpp	CNOhandApp	Application class
Nohanddoc.h	NOhanddoc.cpp	CNOhandDoc	Project document
NOhandvw.h	NOhandvw.cpp	CNOhandView	Project view
MainFrm.h	ManinFrm.cpp	CMainFrame	SDI main frame
Analysis.h	Analysis.cpp	Analysis	Analyze dialog
sidata.h	sidata.cpp	InserDlg	Input data dialog class
inset.h	inset.cpp	InsetDlg	Search data dialog class
infor.h	infor.cpp	InforDlg	Information dialog class
StdAfx.h	StdAfx.cpp	---	Precompiled headers
resource.h	---	---	Contains macro definitions
blastit.h	blastit.cpp	CBlastit	Input list class
coil.h	---	---	Field information
data.txt	---	---	Data file

Table-1. The files and classes in the project.

<b>Parameter</b>	<b>Meaning</b>
oil production	amount of oil produced by a well per day.
gas production	amount of natural gas produced by a well per day.
water production	amount of water output in a well per day.
gas injection	amount of natural gas injected into a well per day.
water injection	amount of water injected into a well per day.
liquid level	the depth of a well.
initial flow rate	oil production at the initial time.
economic limit	oil production at the end of the well life.
discovery date	the date when a well was found.
location	the two-dimensional location of a field.
investment	amount of investment in the field.
yearly production	total production within a year.

Table-2. Explanation of parameters.

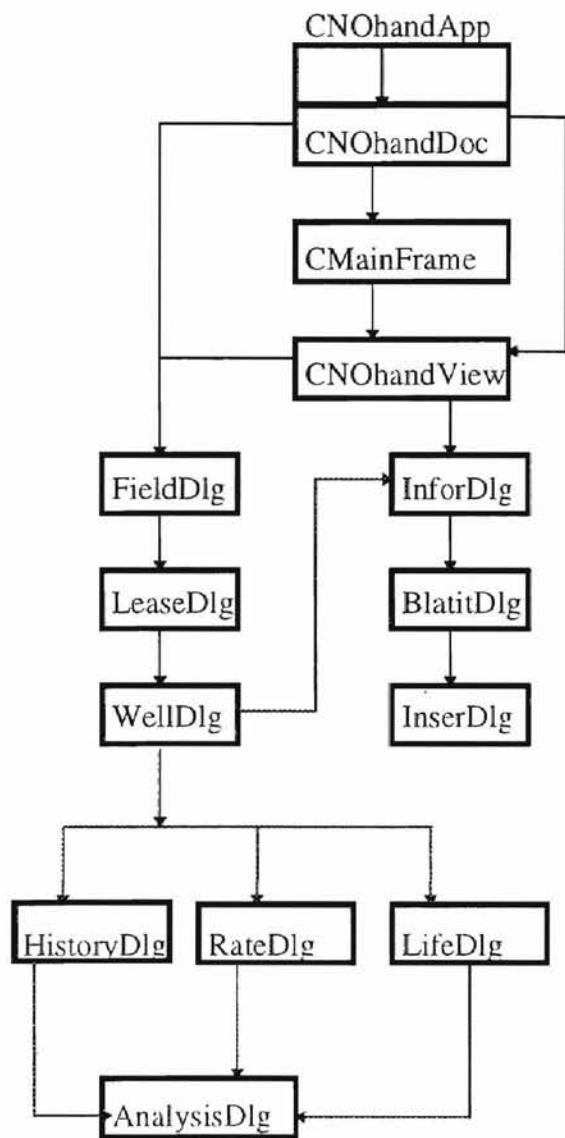


Figure-3 Internal structure of the software.

Oil & Natural Gas Database Management System

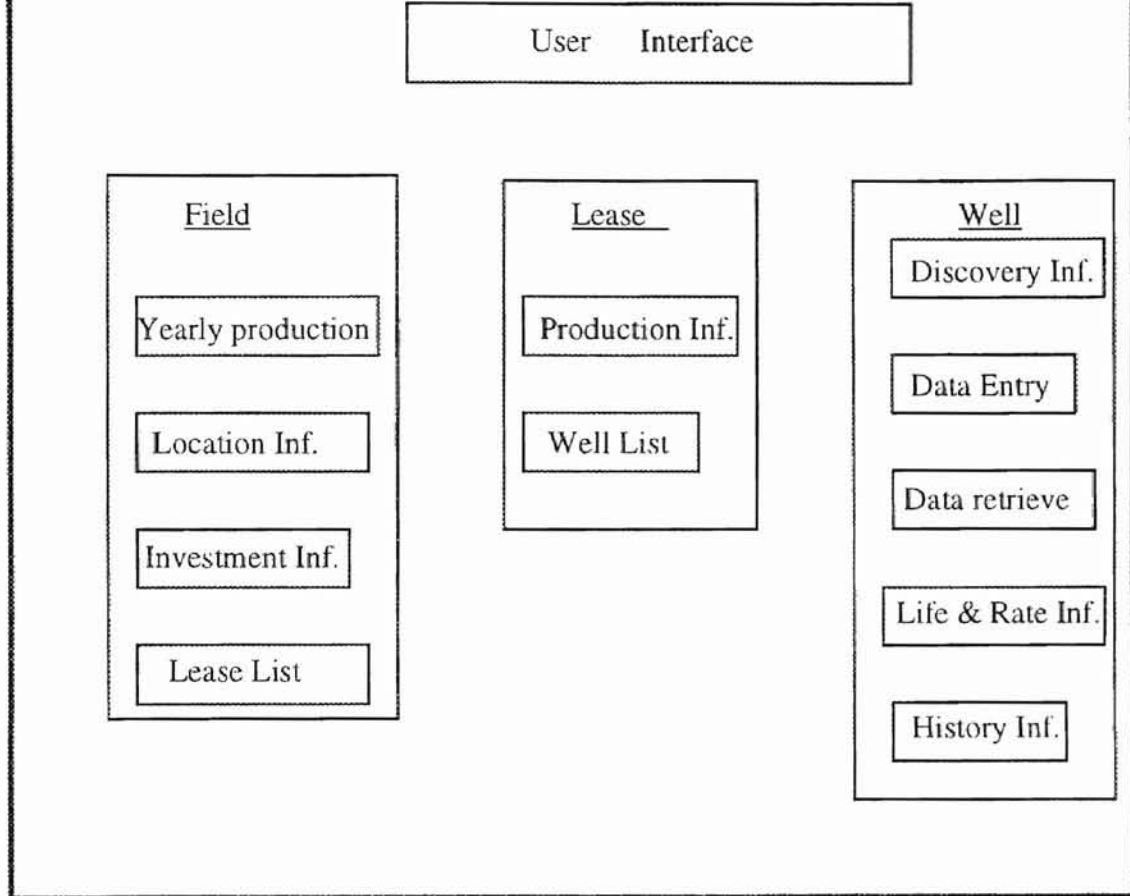


Figure 4 System Diagram.

```
##North N-135W-32 3600 150
**North/RedRock/NewYork 700 20 900201
&&North/RedRock/Dallas
901206 850 200 30 0 0 1000
901212 848 200 30 0 0 1005
901223 844 190 34 0 0 1015
901224 844 190 34 0 0 1015
901225 844 190 34 0 0 1015
**North/RedRock/Atalanta 900 15 920910
**Central/Moon/GoldDream 980 20 910219
##South S-120 E-45 130 4020
##West W-60S-90 180 5000
&&Central/Moon/GoldDream
901206 850 200 30 0 0 1000
901212 848 200 30 0 0 1005
901223 844 190 34 0 0 1015
901224 844 190 34 0 0 1015
```

Table-3. Input data file.



Tables 4-15 give the description of the classes implemented in this project. Class name, description, data member(member variable), and member function are described.

<b>Class Name</b>	NOhandApp
<b>Description</b>	The NOhandApp class is derived from CWinApp which is one of the MFC classes. Each Windows application that uses MFC only has one object derived from CWinApp. The object is constructed when other C++ global objects are constructed and is available when Windows calls the WinMain function which is supported by MFC. CWinApp provides member functions for initializing and running Windows application. NohandApp overwrites the member function InitInstance() to create the application's main window object, and to register document templates.
<b>Member Functions</b>	NohandApp(), InitInstance() , and OnAppAbout()

Table-4. NohandApp Class

<b>Class Name</b>	CAboutDlg
<b>Description</b>	The CAboutDlg Class is derived from CDialog class. The CAboutDlg class is used for project introduction.
<b>Member Functions</b>	CAboutDlg(). DoDataExchange()

Table-5. CAboutDlg Class

<b>Class Name</b>	Field	
<b>Description</b>	The field class is derived from CDialog which is one of the MFC classes. The field class is used by a user to search information of a specific field.	
<b>Controls</b>	Name	Droplist for user to select a specific field name.
	Location	Edit control for user to get location of a specific field.
	Storage	Edit control for user to get oil storage of a specific field.
	Investment	Edit control for user to get investment of a specific field.
	Lease list	Combo box control for user to choose a lease from a lease list.
	Yearly production	Edit control for user to get oil and gas yearly production record of a specific field.
	OK	Button control. When users push it, the action is to search the field information in database, and using the information obtained from the database display the Name, Location, Storage, Yearly production, and lease list in the appropriate control boxes.
	Cancel	Button control. Close the field dialog box.
<b>Member functions</b>	DoDataExchange(), field(), OnDblclkCombol(), OnDblclklist2(), CircleCount_F, ShowFilemess(), OnCancel(), OnOk(), and OnSelendOkCombol().	

Table-6. Field Class.

<b>Class Name</b>	CMainFrame
<b>Description</b>	The CMainFrame Class is derived from CMDIFrameWnd (one of the MFC classes), it provides the member functions for executing the user's options.
<b>Member Function</b>	CMainFrame(), ~CMainFrame(), AssertValid(), Dump(), and OnFileBlastit().

Table-7. CMainFrame Class.

<b>Class Name</b>	Lease	
<b>Description</b>	The lease class is derived from CDialog which is one of the MFC classes. CDialog is the base class for displaying dialog boxes on screen. The lease class is used for user to search information about a specific lease.	
<b>Controls</b>	Well list	Combo box control for user to choose a lease from a well list.
	Yearly production	Edit control for user to get oil and gas yearly production record of a specific lease.
	OK	Button control. When users push it, get the corresponding well management child window.
	Cancel	Button control. when users push it, close the lease dialog box.
<b>Member function</b>	DoDataExchange(), lease(), OnDlhelkwell(), CircleCount_L(), and ShowLeasemess().	

Table-8. Lease Class.

<b>Class Name</b>	Infor	
<b>Description</b>	The infor class is derived from CDialog which is one of the MFC classes. The infor class is used for user to get general information of a specific well.	
<b>Controls</b>	Discover Date	Edit control for user to set discover date of a specific well.
	Economic Limit	Edit control for user to get estimate economic limit of a specific well.
	Initial Flow Rate	Edit control for user to get initial flow rate of a specific well.
<b>Member Variables</b>	m_Date, m_Limit, m_Flow.	
<b>Member functions</b>	infor(), and DoDataExchange().	

Table-9. Infor Class.

<b>Class Name</b>	NOhandDoc
<b>Description</b>	The CNOhandDoc class is derived form CDocument which is one of the MFC classes. CDocument provides the basic functionality for user-defined document class. User interacts with the document through the CView object associated with it. CProgDoc provides the functions for saving all user input data to a file and loading saved data from a file to user's application.
<b>Member functions</b>	ProgDoc(), ~CProgDoc(), OnNewDocument(), Dump(), AssertValid(), and Serialize().

Table-10. NohandDoc Class.

<b>Class Name</b>		BlastitDlg
<b>Description</b>		The Blastitdlg class is derived from CDialog which is one of the MFC classes. CDialog is the base class for displaying dialog boxes on screen. The BlastitDlg class is used for user to enter the daily reading data of a well in a field.
<b>Controls</b>	Date	Edit control for user to enter the specific date for input.
	Water Production	Edit control for user to enter water production on a specific day.
	Water Injection	Edit control for user to enter water injection on a specific day.
	Oil Production	Edit control for user to enter oil production on a specific day.
	Gas Production	Edit control for user to enter gas production on a specific day.
	Gas Injection	Edit control for user to enter gas injection on a specific day.
	Liquid Level	Edit control for user to enter liquid level on a specific day.
	Enter	Button control. When user presses the button, a new data record is added into the database.
Cancel	Button control. When user presses it, the input reading information dialog box is closed.	
<b>Member Variables</b>		m_Day, m_Oilp, m_Waterp, m_Wateri, m_Gasp, m_Gasi, m_Level.
<b>Member functions</b>		blastit(), DoDataExchange(), OnInser(), OnCancel(), OnOK(), w_to_struct1(), DeleStr(), w_to_struct2(),and add_data().

Table-11. BlastitDlg Class.

<b>Class Name</b>		Inset
<b>Description</b>		The Inset class is derived from CDialog which is one of the MFC classes. The Inset class is used for user to search the daily reading data of a well in a field.
<b>Controls</b>	Date	Edit control for user to search the specific date for search.
	Water Production	Edit control for user to get water production on a specific day.
	Water Injection	Edit control for user to get water injection on a specific day.
	Oil Production	Edit control for user to get oil production on a specific day.
	Gas Production	Edit control for user to get gas production on a specific day.
	Gas Injection	Edit control for user to get gas injection on a specific day.
	Liquid Level	Edit control for user to get liquid level on a specific day.
	Search	Button control. When user presses the button, start to search a specific day's reading data in the database, using the data obtained from the database write water production, water injection, oil production, gas production, gas injection, and liquid level in appropriate control boxes.
Cancel	Button control. When user presses it, close the search information dialog box.	
<b>Member Variables</b>		m_Day, m_Oilp, m_Waterp, m_Wateri, m_Gasp, m_Gasi, m_Level.
<b>Member functions</b>		OnInitDialog(), DoDataExchange(), OnOk(), OnInser(), InsertData(), CInpDlg(), and OnCancel().

Table-12. Inset Class.

<b>Class Name</b>	NOhandView
<b>Description</b>	The CProgView class is derived form CView which is one of the MFC classes. CView provides the basic functionality for user-defined view class. Users interact with the view through the CDocument object associated with it.
<b>Member functions</b>	CProgView(), GetDocument(), AssertValid(). Dump(), OnPreparePrinting(), OnBeginPrinting(), OnEndPrinting(), ~CProgView(), DoDataExchange(), and OnDraw().

Table-13. NohandView Class.

<b>Class Name</b>	CWell
<b>Description</b>	The CWell class is derived from CDialog which is one of the MFC classes. The CWell class is used for user to search the general information, daily data, life, rate and history data of a well in a field.
<b>Member functions</b>	CWell(). OnInfrGeneralinformation(), OnDailydataSearch(). OnDailydataInputdata(), OnAnalysisHistory(). OnInfrExit(). GetTheDrawPoint(), OnAnalysisLife(), OnAnalysisRate(). w_to_struct32(). and DoDataExchange().

Table-14. Well Class.

<b>Class Name</b>	Analysis
<b>Description</b>	The Analysis class is derived from CDialog which is one of the MFC classes. The Analysis class is used for user to draw history data of a well in a field.
<b>Member functions</b>	Analysis(), DoDataExchange(), OnPaint(), DrawAxes(), and DrawGraph().

Table-15. Analysis Class.



## CHAPTER IV

### USER INTERFACE

This chapter describes the user interface features of the software developed in this thesis. The structure of the implementation of the program is shown as figure 5. It shows the control flow of the program when a user follows the menus provided as user interface. The program is executed under Microsoft Visual C++ 2.0. Under "Project" menu, open the execution file, and then run the execution file. The main window, titled 'Oil And Natural Gas Field Management System' will show up. It has a menu bar with two items and each item in the menu bar has a pull down menu. The two items are File, and Help. This is shown in figure 6.

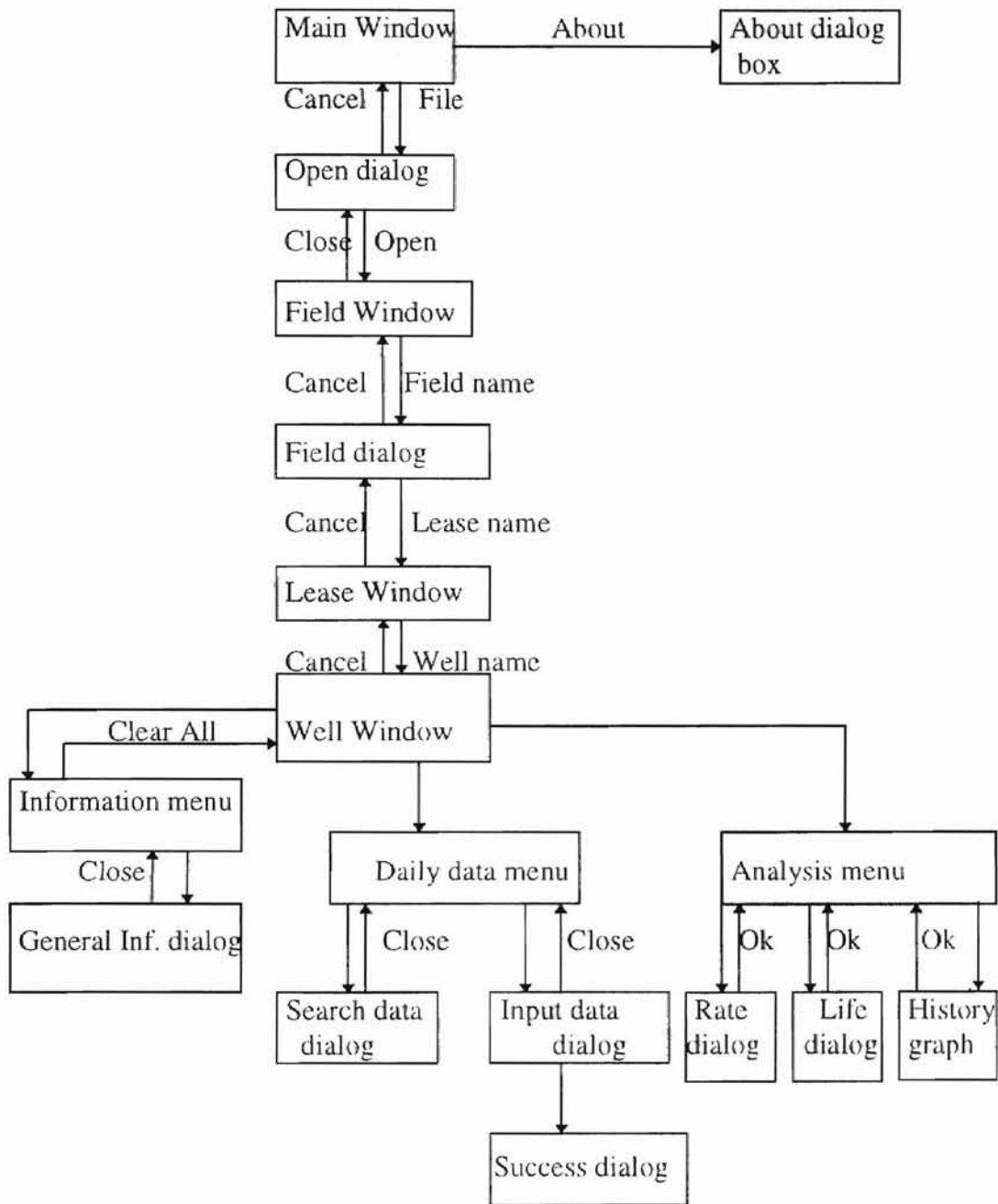


Figure-5 Structure of program implementation.

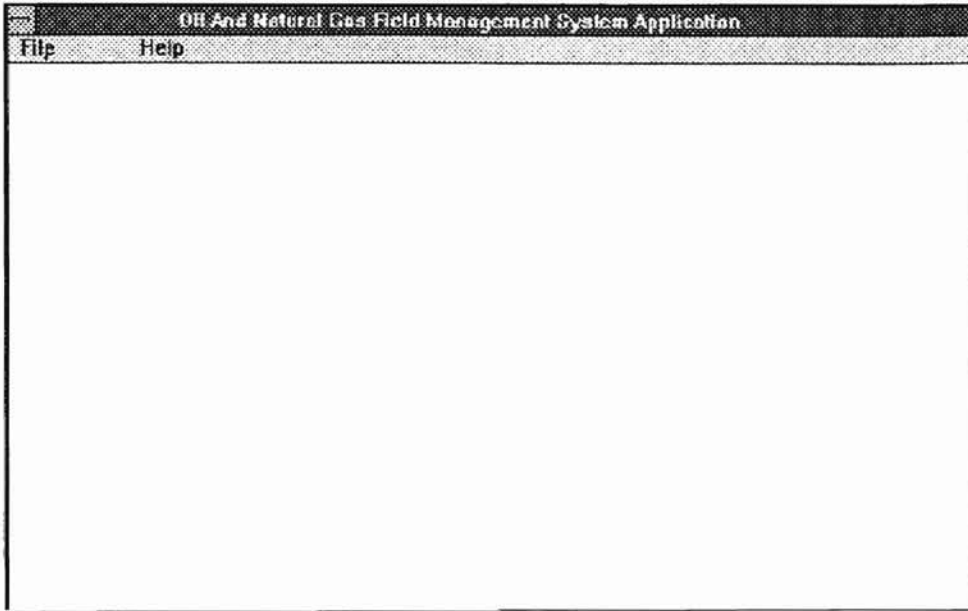


Figure-6 The Main frame window.

If the user chooses 'About Project' from the help menu item, a dialog box titled 'About Project' will show up, it introduces the project, this is shown in figure 7.

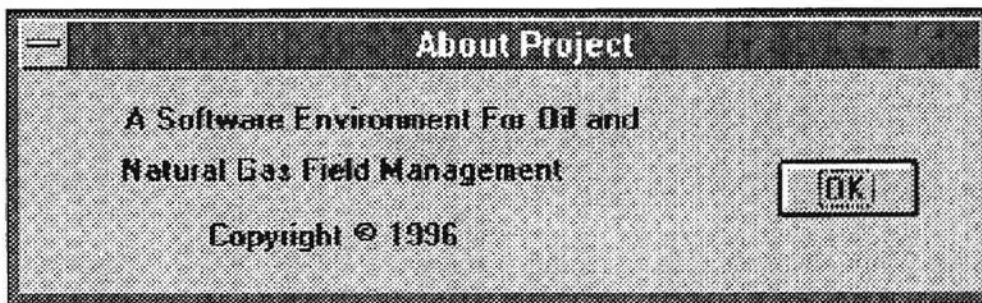


Figure-7 The about dialog box

Once the user clicks on 'File menu', it brings a pull down menu with the options of Open, Print, and Exit. This is shown in figure 8.



Figure-8 File menu.

Clicking on Open will open a dialog with two option buttons-OK and Cancel, this is shown in figure 9.

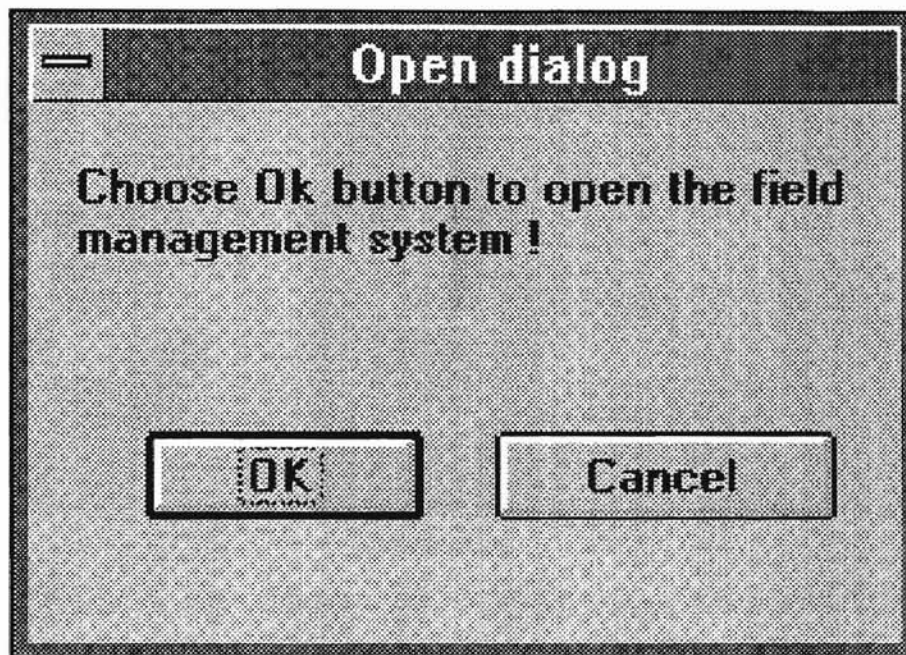


Figure-9 Open dialog box.

If the user clicks on the 'OK' button, it will open another window titled 'Oil And Natural Gas Field Management System'. Contents of this window are a droplist of all the oil and gas field names, with several edit boxes of location, investment, storage, and yearly production, and a scrolled list of leases within the selection field. This is shown in figure 10.

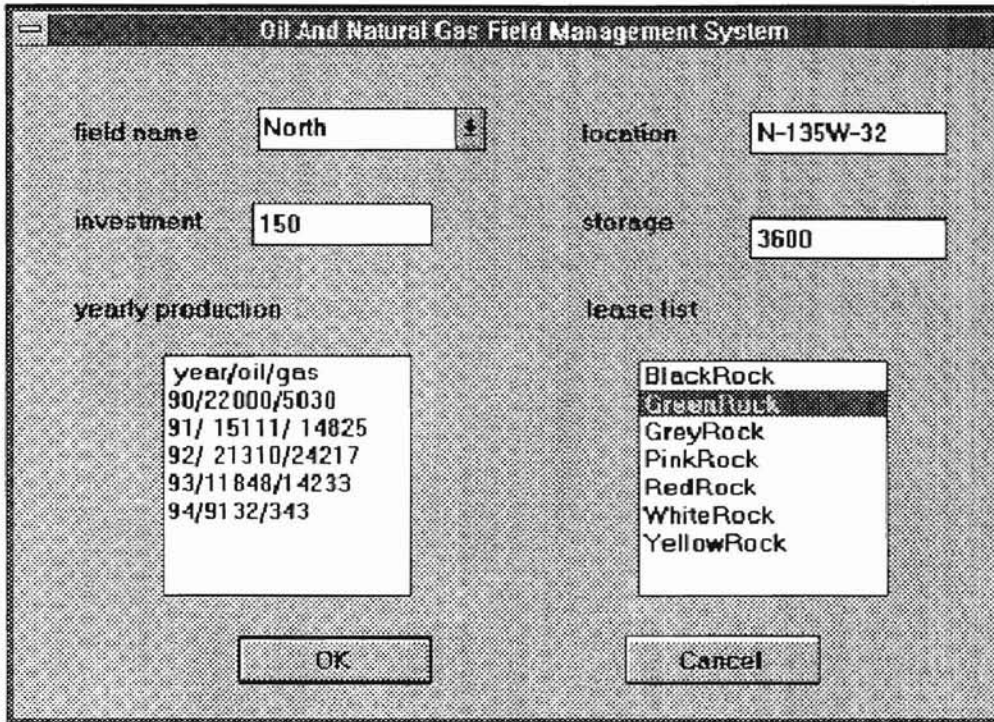


Figure-10 Oil and natural gas field management system-field level.

Selecting a field name and clicking on 'OK' button, will display the information of the selected field. Selecting a lease name and double clicking on it will open another window titled 'Oil And Natural Gas Field Management System(lease level )', with an edit box and a scrolled list of wells within this lease. This is shown in figure 11. Clicking on 'Cancel' button will return to previous window.

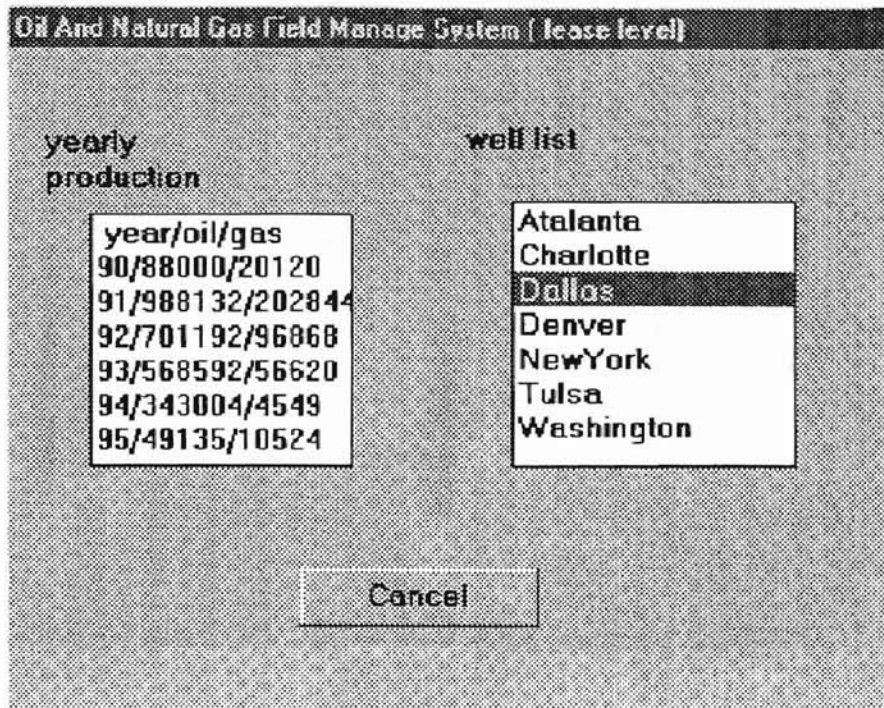


Figure-11 Oil and natural gas field management system-lease level.

Selecting a well name, and clicking on 'OK' button, will display a window titled 'Well Level Management'. It has a menu bar with three items: Information, Daily Data, and Analysis. This is shown in figure 12. To return to the previous window, the user need to click on the 'Cancel' button

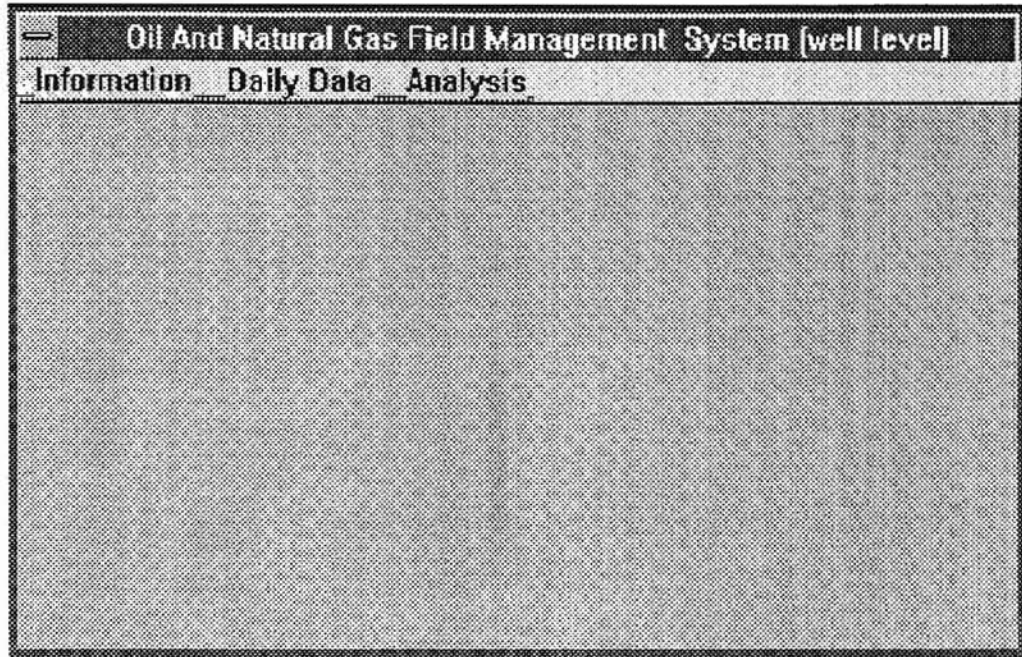


Figure-12 Oil and natural gas field management system-well level.

If a user clicks on Information, it brings a pull down menu with the options of General Information, and Clear All. This is shown in figure 13.



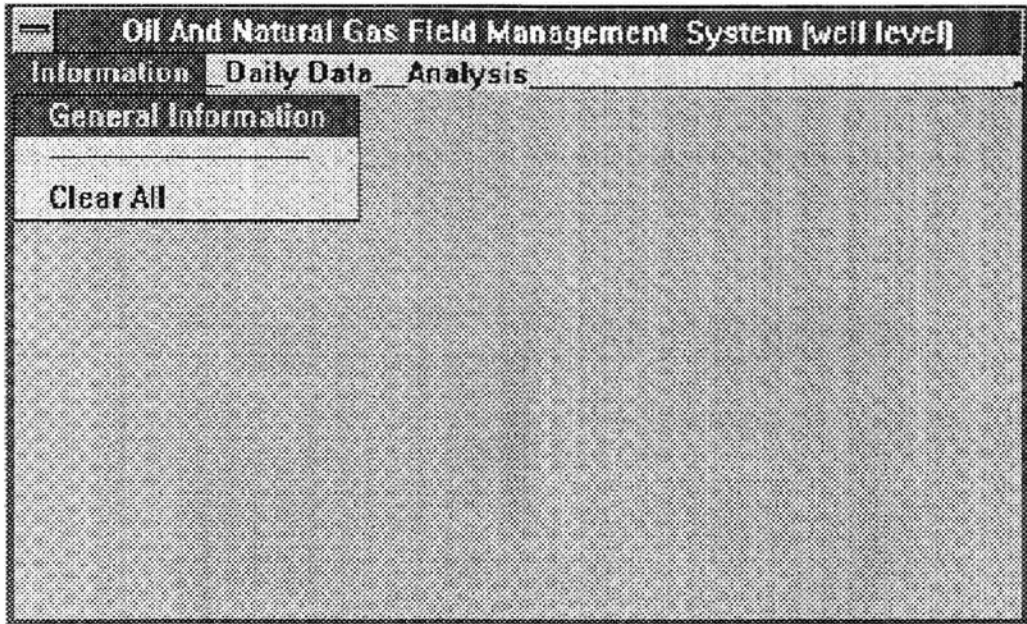
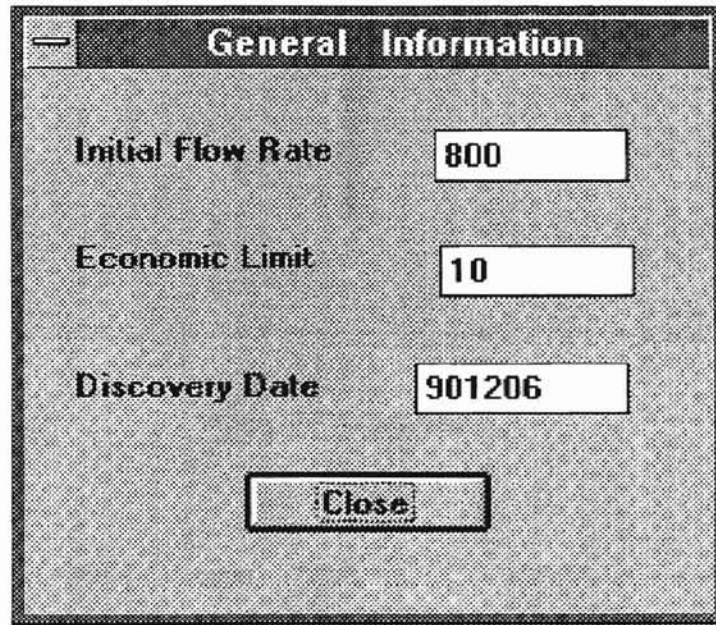


Figure-13 Information menu.

General Information is related to a window titled "General Information". It has three edit boxes, they are Initial Flow Rate, Discovery Date, and Economic Limit, as shown in figure14. Clicking on 'Close' button will return to previous window.



The image shows a window titled "General Information" with a standard window control bar (minimize, maximize, close). The window contains three input fields and a button:

Initial Flow Rate	800
Economic Limit	10
Discovery Date	901206

At the bottom center of the window is a button labeled "Close".

Figure-14 Information window.

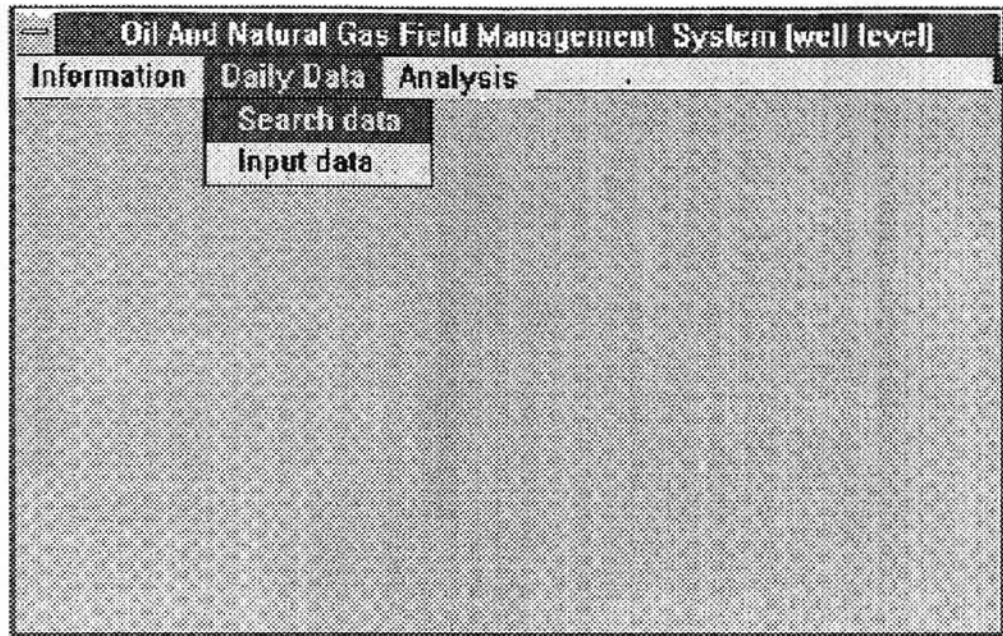


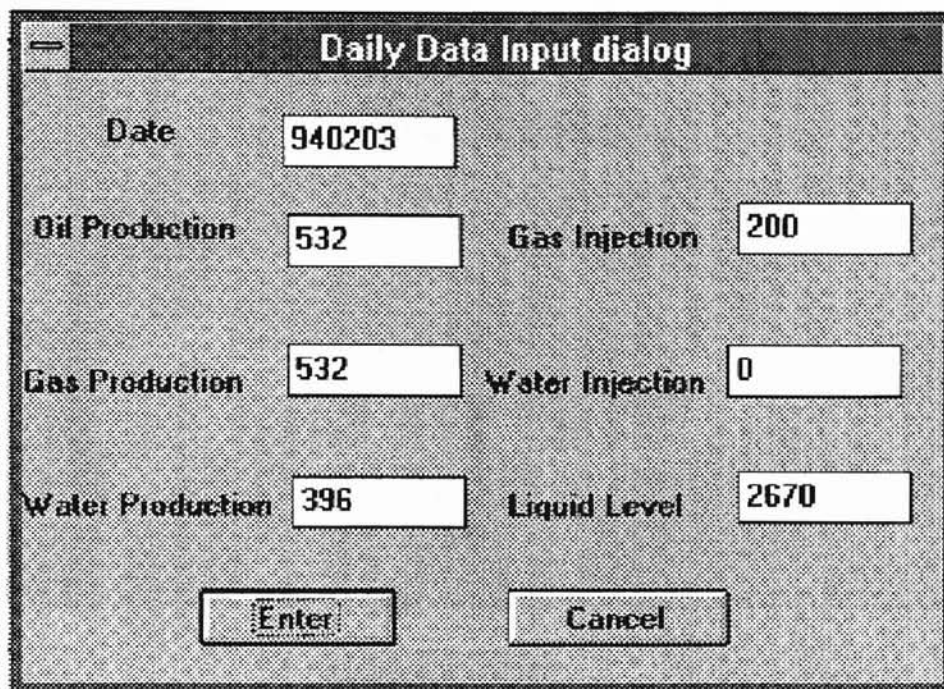
Figure-15 Daily data menu.

Clicking on Daily Data box brings up a pull down menu with the options of Search and Input. This is shown in figure 15. Clicking on 'Search' brings up a window titled "Search Data". It has seven edit boxes: search date, oil production, water production, gas production, gas injection, water injection and liquid level. User should type in the date and then click on 'Search' button to search daily data, as shown in figure 16.

Daily Data Search dialog			
Search Date	920426	Search	
Oil Production	515	Gas Injection	0
Gas Production	80	Water Injection	0
Water Production	225	Liquid Level	1604
Cancel			

Figure-16 Search dialog box.

Clicking on 'Input' brings up a window titled "Input Data". It has seven edit boxes. they are search date, oil production, water production, gas production, gas injection, water injection and liquid level. User should type in the date and daily data as shown in figure 17, then click on 'Enter' button. If the data is successfully entered in database, it will bring a successful dialog box. This is shown in figure 18. Clicking on 'Cancel' button will return to previous window.



The image shows a dialog box titled "Daily Data Input dialog". It contains several input fields for data entry. The fields are arranged in a grid-like fashion. At the bottom, there are two buttons: "Enter" and "Cancel".

Field Name	Value
Date	940203
Oil Production	532
Gas Injection	200
Gas Production	532
Water Injection	0
Water Production	396
Liquid Level	2670

Figure-17 Input data dialog box.



The image shows a dialog box titled "Success". It contains a single line of text: "The daily data of '940203' has been entered into database!". At the bottom center, there is an "OK" button.

Figure-18 Successful dialog box.

Clicking on Analysis brings a pull down menu with the options of Life, Rate, and History Graph. This is shown in figure 19.

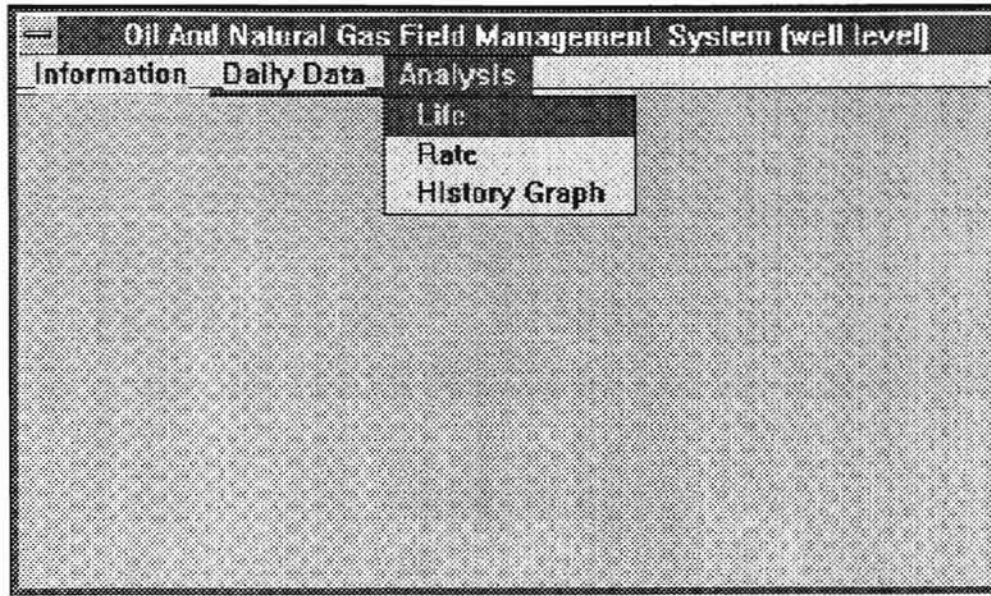


Figure-19 Analysis menu.

When user clicks on 'Rate', it will bring up the corresponding rate dialog box. This is shown in figure 20. If user clicks on 'OK' button the program will return to the previous window.



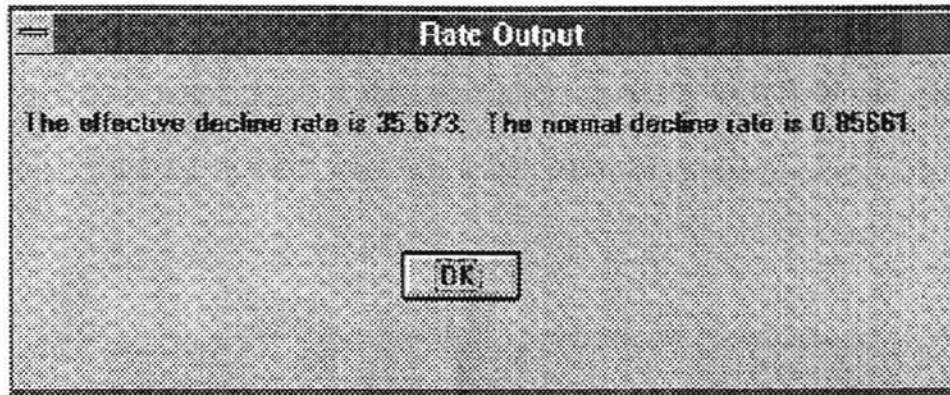


Figure-20 Rate dialog box.

If user clicks on 'Life', it will bring up corresponding life dialog box. This is shown in figure 21. If user clicks on the 'OK' button the program will return to previous window.

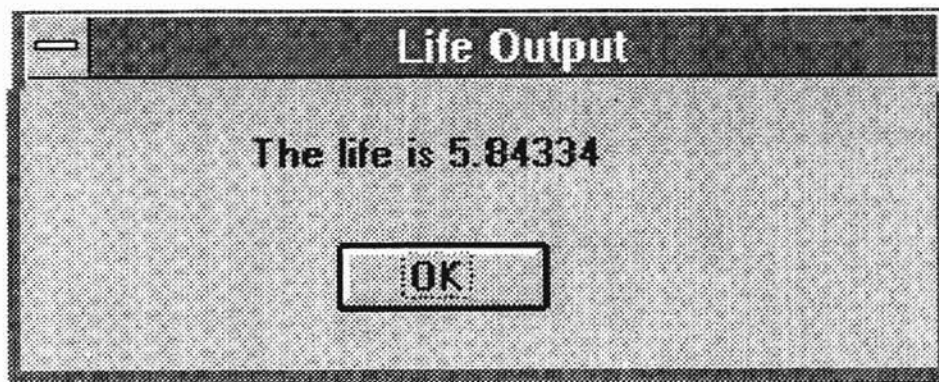


Figure-21 Life dialog box.

Clicking on 'History Graph' brings up the graph dialog box. This is shown in figure 22. In this figure, it gives the information about the well flow rate change with time change. Clicking on the 'Close' button will return to the previous window.

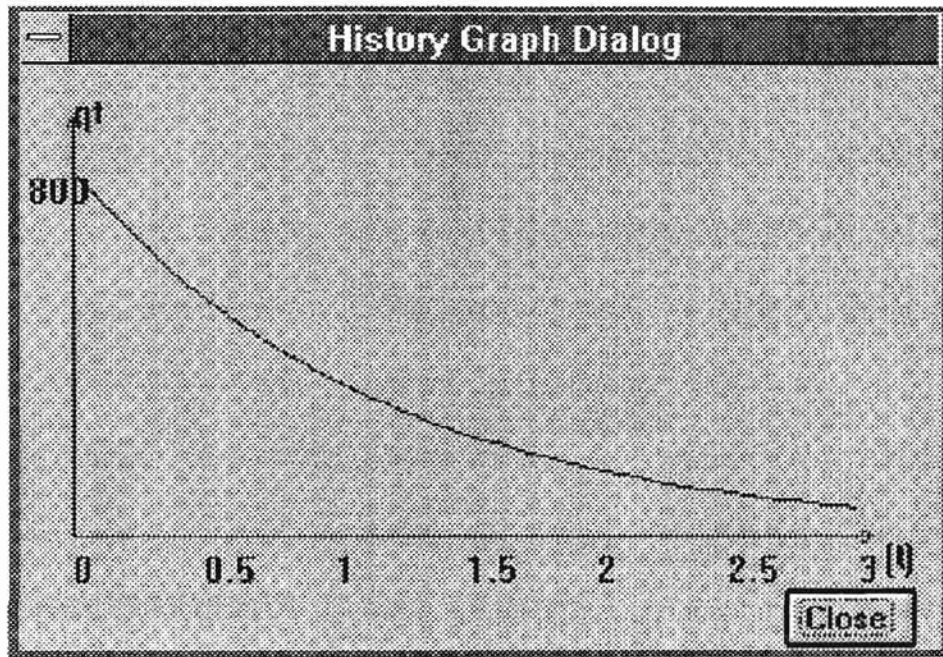


Figure-22 History graph dialog box.



## CHAPTER V

### SUMMARY AND CONCLUSION

In this research project, the author designs and implements a window-based oil and natural gas field database management system. Microsoft Visual C++ is used to develop graphic user interface. The system presents a method to analyze data automatically. People without computer programming experience and petroleum knowledge can use it easily. This program has been tested by using real field data. Future improvement and research can be directed at enlarging the oil and gas fields, displaying more information of those fields, such as maturation, generation, and migration. This improvement is expected to be useful to petroleum companies and business persons.

## REFERENCES

- [1]. Ananthaswamy Anil, Data communications using object-oriented design and C++ McGraw-Hill, New York, 1995.
- [2]. Andrews, Mark, Visual C++ object-oriented programming, SAMS Pub., Carmel, Ind., 1993.
- [3]. Arps, J.J., "Analysis of decline curves", Petroleum technology, September 1994.
- [4]. Burnett, Margaret, M., Visual objected-oriented programming: concepts and environments, Manning, Greenwich, CT, 1995.
- [5]. Dawson, John A., Computing for geographers, Crane, Russak, Newton Abbot: David and Charles, NewYork. 1976.
- [6] Dean Keiswetter & Brett Bennett. "A Windows application for generating input models for numerical modeling", Computer & Geosciences, Vol. 22, No. 3. April, 1996. pp355-357.
- [7]. Florentin, J.J.. Object-oriented-programming systems, Chapman & Hall. London. 1991.
- [8]. Halladay Steve. Object-oriented software engineering, R&D publications. Lawrence, KS. 1993.

- [9]. Henry F. Korth & Abraham Siberschatz, Database system concepts, McGraw-Hill, NewYork, 1991.
- [10]. Henry Lieberman, "Intelligent graphics", Communications of the ACM, Vol. 39, No.8 August 1996, pp38-49.
- [11]. <http://www.cs.princeton.edu/courses/cs111/modules/programming/java/seqvsevent>.
- [12]. <http://www.sb.com.au/wayne/tdcpppl/dcp11p7.htm>
- [13]. Hu, David, Object-oriented environment in C++, MIS Press, Portland, OR, 1990.
- [14]. Huang Jinyu, Statwin: "An objected-oriented approach to estimate the toxicity of effluents on fresh water organisms", M.S. Thesis in computer science department, Oklahoma State University, 1995.
- [15]. Jamal A. Abdalla, "An object design framework for structural engineering", Engineering with computers, Vol. 11, No. 4, 1995, pp 213-226
- [16]. Kruglinski,David. Inside visual C++, Third Ed, Microsoft Press, Redmond, WA, 1995.
- [17]. Landon, Robert C., Principles of petroleum development geology, PTR Prentice Hall, Englewood Cliffs, NJ, 1996.
- [18]. Leavens, Alex. Designing GUI applications for Windows, M&T Books, New York, 1994.
- [19]. Marcus, Aaron. The Cross-GUI handbook: for multiplatform user interface design, Addison-Wesley Pub. Co., Reading, MA, 1995.

- [20]. Michel, J.L. Robin, "Software review", Computer & Geoscience, Vol.21, No. 9, November 1995, pp1113-1117.
- [21]. Nobles, M.A., Using the computer to solve petroleum engineering problems, Gulf Pub. Co., Houston, TX, 1974
- [22]. Paul G. Teleki, Basin analysis in petroleum exploration, Kluwer Academic, Dordrecht, Boston, 1994.
- [23]. Robinson, Joseph, E., Computer applications in petroleum geology, Hutchinson Ross Pub. Co., Stroudsburg, PA, 1982.
- [24]. Rodey Bell & David Sharon, "Tools to engineer new technologies into applications", IEEE Software, Vol. 12, No 4, March 1995, pp 11-16.
- [25]. Speight, J.G., The chemistry and technology of petroleum, M.Dekker, New York, 1980.
- [26]. Stephen G. Eick, Micael C. Nelson, "Graphical analysis of computer log files", Communications of the ACM, Vol. 37, No.12, December 1994, pp 50-57.
- [27]. Takayuki Dan Kimura, "Form/Formula: A visual programming paradigm for user-definable user interfaces", Computer, Vol 28, No.3, Marchh 1995, pp 27-35.
- [28]. Timothy Budd, Object-Oriented Programming, Addison-Wesley publishing company, Reading, MA, 1991.
- [29]. Troutman, Arthur, The DEEP Edward's trend in south Texas, Oil Frontiers Pub. Co., Austin, TX, 1958.

- [30]. United States, Bureau of Land Management, Oil & gas leasing analysis : final environment impact statement, US Dept. of Agriculture, Forest Service, Washington, DC, 1993.
- [31]. Voisard A. "Towards a toolbox for Geographic user interfaces", Advances in spatial database, 2nd Symposium, SSD 1991, Zurich, Switzerland, August 1991, proceedings, pp 75-97.
- [32]. Xia Chen, A CAI system for learning natural resource knowledge ( NRKS), Thesis, Computer Science Department, Oklahoma State University, 1994
- [33]. Yourdon, Edward, Object-oriented system design, E Press, Englewood Cliffs, N.J,1994.

2

VITA

Xiyuan Chen

Candidate for the Degree of

Master of Science

Thesis: A SOFTWARE ENVIRONMENT FOR OIL AND NATURAL GAS  
FIELD MANAGEMENT

Major Field: Computer Science

Biographical :

Education: Graduated from Nankai University, Tianjin, China in 1988; received Bachelor of Science Degree in Applied Chemistry. Complete the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in May 1997.

Experience: Chemical Engineer, Synthetic Material Research Institute, China, July 1988, to June 1993.