CREATING A WEB-BASED TUTORIAL FOR

IBM VisualAge FOR COBOL

By

MEI WANG

Bachelor of Science
Hunan Agricultural University
Changsha, P. R. China
1985

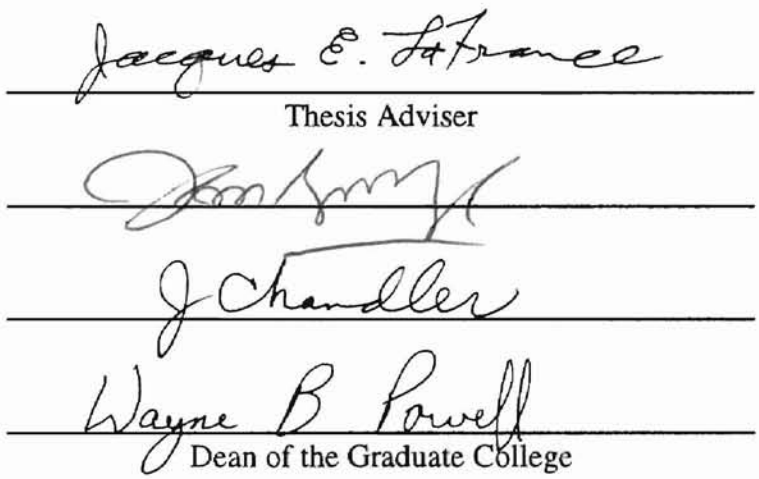
Master of Science
Southwest Agricultural University
Chongqing, P. R. China
1988

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, 1998

CREATING A WEB-BASED TUTORIAL FOR

IBM VisualAge FOR COBOL

Thesis Approved:



Thesis Adviser

Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my advisor Dr. Jacques LaFrance for his guidance, encouragement, patience and kindness throughout my graduate study. I am deeply grateful to Dr. John P. Chandler and Dr. K. M. George for their serving on my graduate committee and providing me invaluable advices and assistance. Same gratitude goes to Dr. G. E. Hedrick.

During my graduate study at Oklahoma State University, I benefitted much from the faculty and staff of the Computer Science Department. I would like to thank all the professors for their initiative and intuitive teaching. I also appreciate the opportunity that was granted to me as a teaching assistant, which facilitated my study academically as well as financially.

I am greatly indebted to my parents for their unending love and care throughout my life. Special thanks go to my husband, Liping Jiang, for his love, understanding, encouragement, and support. Without his love and support, I would not have completed my study today.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

COBOL and Object-Oriented COBOL

COBOL is a very important and pervasive programming language in the history of commercial computing. It has been the most commonly used computer language worldwide for more than thirty years [Chapin, 1997]. Since it was developed in 1959, COBOL vendors have provided proprietary extensions to the language for their customers. Standard COBOL has been revised and standardized periodically to keep pace with the development of technology and meet the rising demands of business. Until now, three revisions have been approved as standard COBOL; they are COBOL 68, COBOL 74, and COBOL 85.

As a high-level language, COBOL is relatively simple and highly readable. It was created to be usable even by a relatively less experienced programmer. A wide assortment of applications in engineering, mathematical, and scientific area can be implemented successfully using COBOL, showing the power and versatility of COBOL. As the 'BOL' in the name COBOL suggests, COBOL is a Business-Oriented Language. Its simple but capable table-handling facilities and input and output facilities make it the

best fit for business application purposes. Also, it is hardware-independent, thus resulting in a high degree of application portability. Very importantly, COBOL's readability and understandability give several times the productivity of using other languages for developing and maintaining applications [Chapin, 1997].

COBOL's ability to manipulate data makes it a good language for dealing with the vast amounts of data which businesses process [Longhurst, 1989]. It is uniquely suited for business application development, maintenance, and production support [Sayles]. Although there are so many new languages and data processing tools today, COBOL is still running on more than seventy percent of mainframe and traditional systems [IBM-ITSO, 1996]. Even when the front end of a system is converted to PC-based graphical user interface, the ultimate destination of the data is often an old system written in COBOL. COBOL lies at the heart of financial systems and probably controls more administration of our daily life than other technical tools [Levey, 1996]. There are approximately seventy billion lines of COBOL code in use and over five million active COBOL programmers around the world [IBM-ITSO, 1996]. Gartner Group August 1995 reported that COBOL was still used in over sixty-five percent of new application development [Sayles]. Since industry has a huge investment in the installed base of COBOL code and in the people who code and maintain the COBOL system, COBOL will remain the tool of choice for most business applications [Levey, 1996].

During past years, we have enjoyed many dramatic improvements in computer industry. The performance improvements and price-dropping of computer hardware have inversed the cost in developing and maintaining systems. Hardware is no longer a

constraint whereas personnel and software account for a majority part gradually [Chapin, 1997]. Today, maintaining existing COBOL systems occupies a very large percentage of data processing budgets, and also making maintenance changes becomes harder and harder over time because each change increases program complexity. As complexity grows, COBOL systems appear unchangeable. To regain control, re-engineering of those systems should be integrated with object-oriented techniques [Levey, 1996].

Object orientation has emerged as an important approach to software engineering in recent years. Object-oriented programming adopts the idea of objects in the real world and models data in terms of real-world objects. With object-oriented programming, application developers may decompose a problem into related subgroup. Each subgroup becomes a self-contained object that includes its data and its code that manipulates the data, called methods. In this way, complexity is reduced and programmers can manage larger problems. Usually, the data of an object can only be accessed through its methods, which is called encapsulation. The encapsulation mechanism keeps data and methods safe from outside interference and misuse, thus giving object-oriented programming its security and robustness. As in the real world, objects may be similar in many respects. An object may have a general set of properties inherited from its parent, as well as some specific properties that distinguish itself from others. With this inheritance mechanism, one object may be defined as the same as others but with its different data and/or methods. Inheritance allows a hierarchy of classes to be built, moving from the most general to the most specific. This property encourages the definition of new classes without extensive duplication of code, thus increasing the productivity of programmers

and simplifying the system maintenance. Another important property of object-oriented programming is polymorphism, which allows the same method to do different things in a general class of actions on different objects. The specific action selected is determined by the exact nature of the situation. This helps handle problems with greater complexity by allowing the creation of standard interfaces to related activity. Due to its promising security, reliability, reusability, robustness, and productivity, object orientation is uniquely capable of supporting complex and changing business applications. It is believed that the direction of the computer industry will be object-oriented programming [Hares, 1993; IBM-ITSO, 1996].

Although COBOL is not an object-oriented language, it is possible to build object-oriented systems using the COBOL language as it exists today. By creating hierarchical records and associated lists of executable programs and changing the way that those programs are called, unnecessary control flow can removed from programs and much of the excessive decision-making logic can be eliminated from systems [Levey, 1996]. To provide COBOL with object-oriented capability, a group, called X3J4, including a number of COBOL and object-oriented interested parties, was set up in 1989 to work on combining the two technologies. In 1993, they presented their first proposal, the object-oriented COBOL version X3J4.1 OOCOBOL, to ANSI. The new ANSI standard COBOL is expected to come soon [IBM-ITSO, 1996].

The Object-Oriented COBOL retains nearly all capabilities of COBOL-85 and adds many new features, especially object-oriented extensions, providing an opportunity of "securing major improvements in the ways information systems serve organizations by

implementing systems in O-O COBOL" [Chapin, 1997]. The Object-Oriented COBOL has been considered a more powerful and capable language than C++ and Smalltalk in many ways. It not only lets a programmer define everything, including numbers, as objects, but also gives a developer the freedom to declare typed as well as untyped objects for speed and safety and to incorporate both objects and procedural constructs according to his/her needs [Arranga, 1996]. As shown in Figure 1, object-oriented COBOL will become more and more important in the information technology area [IBM-ITSO, 1996].

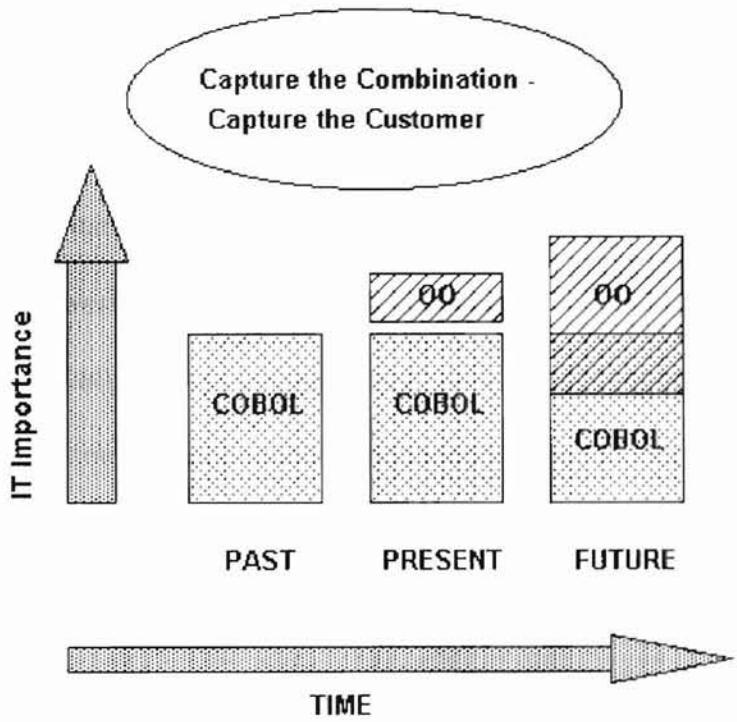


Figure 1. The Importance of OO COBOL
[IBM-ITSO, 1996].

IBM VisualAge for COBOL

To enable COBOL programmers to use object-oriented techniques to create more adaptable systems without having to learn a new language, several companies have delivered object-oriented COBOL compilers and development tools, including Micro Focus' Visual Object COBOL, Hitachi America's Object-Oriented COBOL, and IBM's VisualAge for COBOL [LaMonica, 1995; Arranga, 1996].

VisualAge for COBOL is a member of IBM's VisualAge family, which includes VisualAge for RPG, VisualAge Generator, VisualAge for Smalltalk, VisualAge for C++, VisualAge for Java, and VisualAge for Basic as well. IBM's VisualAge is a set of object-oriented application-development tools for dealing with today's complex, mission-critical, graphical user interface (GUI), client/server workstation applications. VisualAge integrates object orientation with visual programming approaches to enable programmers to take advantage of working with objects in a easier, graphical way. It uses the construction-from-parts paradigm and the Visual Modeling Technique methodology, which ease the migration to object-oriented technology [Tkach, 1996]. In addition to ready-built components that provide productivity for developers without object-oriented skills, VisualAge also offers extensibility for those who are building object-oriented skills. Among VisualAge family, VisualAge for COBOL is more attractive than other members for enterprises since it is based on object-oriented COBOL, not C++ or other difficult-to-learn object-oriented languages [Hayashi, 1996].

IBM VisualAge for COBOL is an innovative application development environment. It offers application developers the advanced features they need for today

and tomorrow [IBM, 1997]. VisualAge for COBOL supports object-oriented extensions of COBOL, which is based upon the proposed '97 Standard. The object-oriented extensions, with syntax changed slightly, implement a complete object-oriented paradigm and let developers get the productivity benefits of object-orientation. With these extensions, developers may define object classes, instantiate objects, invoke object methods, and create subclasses inheriting characteristics from other objects. Object-oriented COBOL helps application developers develop applications fast by dramatically simplifying code reuse. The COBOL objects created today can serve as the parts for building other applications tomorrow. Reusing tested and proved objects contributes to improving application quality. Since each part of an object-oriented application is highly encapsulated as an object, maintaining the application will be much easier [IBM-ITSO, 1996].

VisualAge for COBOL provides a set of visual development tools. With its Visual Builder, developers may design and build a graphical user interface by dragging and dropping the provided GUI parts, such as entry fields, buttons, and list boxes. Once the GUI has been designed, each part of the GUI may be associated with the appropriate COBOL program code, and the built-in GUI designer Code Assistant generates event code for the GUI parts to help developers connect the GUI with the COBOL code supporting it. With the Code Assistant's even-driven approach, the code associated with each GUI part automatically allows end users to do what they want. Using the Visual Builder, developers benefit from the object-oriented programming without being retrained for a new language. Visual Builder supports the Construction-from-parts

diagram. The application may be built part by part, either visual or non-visual, and finally parts are connected together through their defined interface. By encapsulating the data of a part, the way that a part works is changed without affecting its external interface. The application development cycle time can be shortened considerably with the created reusable parts [IBM, 1997].

VisualAge for COBOL also provides several visual tools for editing, debugging, and analyzing programs. The COBOL Editor offers language-sensitive capabilities that display various COBOL components in different colors, thus enabling developers to find spelling errors and syntax errors easily. It not only has standard editing functions, but also includes a Data Assistant and a Transaction Assistant. These assistants eliminate a great deal of tedious, error-prone development work. The Data Assistant furnishes access to relational databases, including the entire family of DB2 products. It displays a visual representation of a database's schema and shows tables with their entries. Developers may select the columns and desired actions by clicking with the mouse, then the Data Assistant generates the appropriate SQL queries for them. It may even do a join for developers from different database tables. Briefly, the Data Assistant simplifies the process of constructing embedded SQL statement. The Transaction Assistant eliminates the need to hand-code ECI calls in OLTP application development. It accepts specifications via an on-screen form, and then generates a CICS ECI call and a parameter list for invoking CICS programs [IBM, 1997].

The interactive VisualAge Debugger helps developers detect and diagnose errors in code. It provides a set of functions that let developers quickly and efficiently control

execution and analyze data. With the available functions of the debugger, developers may step through a program in various ways, such as executing one line at a time, pausing at breakpoints, and so on. By setting breakpoints in a program, developers control how a program executes. During debugging, the source code can be displayed as its COBOL code or as assembler instructions; the contents of variables, registers in a processor, and the call stack can be viewed and modified [IBM, 1997].

In addition, VisualAge for COBOL furnishes execution trace analysis and performance tuning through its Performance Analyzer. The Performance Analyzer monitors execution of a program and generates a graphical function-by-function trace file, which includes the entry points to system calls and dynamic link libraries (DLLs), as well as procedures in the executable. It offers several diagrams to display the trace file in detail, thus helping developers understand program's flow and improve the performance of an application thoroughly [IBM, 1997].

Moreover, VisualAge for COBOL provides a Remote Edit/Compile/Debug (Remote E/C/D) function that allows developers to work with OS/390 and MVS applications residing on the host directly at their workstations, thus helping save critical development time. With Remote E/C/D, remote data access, and cross-platform development, developers are not burdened with the need to duplicate and manage host environments any more [IBM, 1997].

The professional edition of VisualAge for COBOL also provides the Redeveloper tools to help reduce the complexity of and simplify the maintenance of legacy applications. The Redeveloper features include Application Understanding to help

identify applications needing redevelopment, Program Conversion to help convert applications into more current levels of ANSI 85 standard applications, and the Year 2000 Analysis Tool to help locate occurrences of two-digit date-related fields in complex programs [IBM VisualAge].

All those characteristics described above make VisualAge for COBOL very attractive to COBOL developers. Some companies have chosen it to implement their new applications since it is able to link into their existing applications and systems and leverage existing behavior as a part of the new application. They think that VisualAge for COBOL provides them the ability to see how the physical implementation is handled and makes them better understand the object-oriented concepts [The Great COBOL Debate].

The Objectives of the Thesis

Since VisualAge for COBOL is such a powerful application development tool and consists of many new features, it is necessary to provide a comprehensive tutorial to help people understand it and master it. The objective of this thesis is to design and implement a Web-based tutorial for IBM VisualAge for COBOL, which is a part of the project — creating a curriculum for IBM certified VisualAge for COBOL object-oriented associate developer. The whole curriculum consists of four parts: Part I is about Object-Oriented COBOL; Part II is on Visual Modeling Technique; Part III is about Visual Builder; and Part IV is on VisualAge for COBOL. The objective of this curriculum is providing users a comprehensive knowledge for object-oriented COBOL programming and for using VisualAge for COBOL to develop object-oriented applications, thus enabling users to keep pace with rapid development of technologies and products.

This thesis is to design and implement a Web-based tutorial for VisualAge for COBOL. The tutorial mainly introduces users the basic features of VisualAge for COBOL development environment and its components, and instructs users to create object-oriented applications using VisualAge for COBOL. It takes advantage of the capability of the World Wide Web in delivering multimedia materials so that many images of VisualAge for COBOL graphical user interfaces can be included along with textual instructions to enrich the power of illustrations. It also takes advantage of the hypertext/hypermedia capability of the Web so that users can learn a topic in their own cognition flow, not restricting by a fixed procedure. Moreover, it exploits the reaching potential of the Web to make it available to a large amount of COBOL programmers via

the Internet.

The Organization of the Thesis

This thesis consists of four chapters, explaining the role of VisualAge for COBOL in object-oriented COBOL programming and the necessity of creating a tutorial for using VisualAge for COBOL to develop object-oriented applications and describing the design and implementation of the Web-based tutorial for VisualAge for COBOL.

Chapter I presents the importance of Object-Oriented COBOL in current business systems development and maintenance and the role of VisualAge for COBOL in object-oriented COBOL programming, which are the motivation to create a tutorial for VisualAge for COBOL and this thesis. It also explains the objectives of the thesis and briefly introduces the contents of the tutorial and the thesis.

Chapter II reviews recent researches and practices on distance education with the Internet and the World Wide Web, providing the possibilities and the basis for creating a Web-based tutorial for VisualAge for COBOL and this thesis.

Chapter III emphasizes on a detailed description about the design and implementation of the Web-based tutorial for VisualAge for COBOL, including the contents of the tutorial, the organization of the tutorial, the development systems and tools, the interface design, and the procedure of implementation.

Chapter IV concerns the results and conclusion. In it, the completed tutorial and future work on the tutorial have been presented.

CHAPTER II

LITERATURE REVIEW

Computers and Distance Education

For many years, distance education has provided opportunities to those who are seeking advanced education and training at home, on the job, or in the military whose multiple responsibilities or physical circumstances prevent them from attending a traditional institution [Bates, 1995]. Distance education does offer additional possibilities for educating and training more people than that can be easily and efficiently accommodated in more traditional settings, such as public schools and universities. It provides a wide range of courses to meet the various needs of people and makes lifelong learning possible and attractive to more people. Learners who participate in distance education programs may learn independently at their own pace. They may choose what they need from a greater variety of subjects, from a greater variety of institutions, and learn at a convenient time and location [Porter, 1997]. Researches comparing distance education to traditional face-to-face instruction indicated that teaching and studying at a distance could be as effective as traditional instruction [Moore, 1990; Verduin, 1991]. Due to the convergence of communication and computing technologies, the need for

information age workers to acquire new skills without interrupting their working for extended time, and the need to reduce education cost, distance education has received extensive attention in recent years [Sherron, 1997]. The International Council for Distance Education has estimated that more than 10 million students are currently taking degree courses at a distance in the world whereas the number of people using distance education methods for other areas and levels of study (such as continuing education, technical education, vocational and professional training) must be comparable if not greater [Van den Brande, 1993].

Initially, distance education meant teaching and learning through correspondence. As new technologies developed, distance instruction was delivered through such media as audiotape, videotape, radio and television broadcasting, and satellite transmission. In recent years, the rapid development of computer networks, dramatic improvements in the processing power of personal computer, and striking advances in magnetic storage technology have made the computer a dynamic force in distance education, providing a new and interactive means of overcoming time and distance to reach learners. Disks, CDs, and e-mails have been considered as distance instruction delivery media [Porter, 1997].

Computers have been considered as a powerful instructional medium for many years. Computer-assisted instruction and computer-based instruction have played an important role in facilitate learning at all levels of education and training [O'Neil, 1981; White, 1988; Steinberg, 1991]. They became popular more than a decade ago, as more companies and educational institutions installed computer equipments [Porter, 1997].

The advantages of computer-based instruction are predominantly reduced cost and increased effectiveness. Computer-based instruction reduces training time and reliance on instructors and provides rapid update of instructional materials. It makes consistent high-quality instruction available on large scale, provides high-quality training at remote site, and permits individualization of instruction [O'Neil, 1981; White, 1988; Steinberg, 1991].

Computers have many advantages in distance education. They can facilitate self-paced learning, and give immediate reinforcement and feedback. With integrated graphic, print, audio, and video capabilities, computers can effectively link various technologies providing learners a multimedia learning environment. In addition, computer networks link resources and individuals, wherever they might be, thus increasing access possibilities [Distance education]. Computer-mediated communication provides people opportunities of exchanging thoughts, ideas, and information via a computer connected to other computers. Using computer-mediated communication as the principal delivery format in distance education provides learners greater interaction with the instructor and other students, more chances for improvement of reading and writing skills, and greater opportunities for self-directed and self-managed personal and professional development [Berge, 1995].

The Internet, the Web, and Distance Education

The Internet is an international network that links one computer to another. Unlike a local area network (LAN), the Internet is a wide area network (WAN), one so large that virtually connects a computer to any other computers around the world. It began in the late 1960's as an experiment in designing a robust computer network that could withstand nuclear attack. The resulting network was a marvelous technical success, known as the ARPANET, but was limited in size and scope and had restrictions in access. Due to the enormous impact that the ARPANET was having on university research, allowing scientists to share data and collaborate on research project, the NSFNET, sponsored by the U. S. National Science Foundation, was began to build to connect to the ARPANET in 1984. In the mid 1980's, people began viewing the collection of networks as an internet, and later as the Internet. The number of networks, computers, and users connected to the ARPANET grew rapidly after TCP/IP became the only official protocol in 1983 [Tanenbaum, 1996]. In the early 1990's, many large computer networks on the mostly government-founded Internet were pressured to be opened for nearly unrestricted traffic by businesses and individuals eager for the power of global digital communications. This greatly popularized the Internet [Musciano, 1997]. By 1994, the number of networks linked to the Internet had been in excess of 45,000 with more than 5 million host computers connected to these networks [Barron, 1995]. The size is still increasing dramatically every year.

Around the same time the Internet opened up for business, an authoring language and distribution system for creating and sharing integrated multimedia electronic

documents on the Internet were released by the CERN, the European Particle Physics Laboratory. So was born HTML (Hypertext Markup Language) and the World Wide Web [Musciano, 1997]. The World Wide Web (the Web or the WWW) is a system of Internet servers that support specially formatted documents. These documents are formatted in HTML that supports links to other documents, as well as graphics, audio, and video files. A Web browser, such as Mosaic, Netscape Navigator, or Microsoft Internet Explorer, is an easy-to-use, graphically oriented, and point-and-click software for the Internet, enabling easy access to the Web. With the ability of most Web browsers to provide an interface for the other Internet protocols, the Web has become the primary user interface of the Internet and even a synonym for the Internet [December, 1994].

The Web brings a couple of very important features to the Internet [Brooks, 1997]. First, it provides access to full fonts, sizes, and styles for text, and may include images, sounds, and movies. Second, it offers true hyper-textural links between documents anywhere on the Web, enabling people to go among and across a series of documents or pages simply by selecting a link. The features of the Web enable educators to create documents containing hypertext/hypermedia links to provide learners a specific topic combined with a large amount of relevant information and references. Following these links in a sequence is often unique to the individual learner.

The Internet creates points of contact for widely dispersed people across time and distance and makes available rich resources of information of all kinds, thus becoming an important media in distance education in recent years [Parson; Alexander, 1995; McManus]. The Internet has made it incredibly easy for anyone with a modem to access

the educational materials and services and reach hundreds of fully accredited colleges and universities throughout the United States [Duffy, 1997; Porter, 1997].

Distance education on the Internet usually takes one or more of the following forms [Wulf, 1996; Porter, 1997]:

1. Electronic mail, or E-mail. It allows one person to write and send messages to an individual or a group. It is usually used to deliver course materials, send assignments, get feedback, and discuss through listserv.

2. Electronic bulletin boards and newsgroups. They link individuals to more information and other people interested in similar topics and provide a virtual place for discussion.

3. Course materials and tutorials. They can be downloaded from the Internet.

4. Interactive tutorial on the Web.

5. Intranets for distributing training for employee.

6. Informatics. Many online databases can be used to acquire information and pursue researches.

Advantages of delivering distance education on the Internet and the Web include multimedia and interactive capabilities, time and place flexibility, potential to reach a global audience, no concern about compatibility of computer equipment and operating systems, quick development time (compared to videos and CD-ROMs), easy updating of contents, as well as archival capabilities, and usually lower development and operating costs (compared to satellite broadcasting) [Bates, 1995; Eastmond, 1995; Wulf, 1995].

As more and more people have access at home or in the office to the World Wide

Web, as well as other parts of the Internet, distance educators have considered the Web as a major instruction and training delivery tool, whereas other Internet services, such as e-mail, mailing list, and bulletin board, as supplementary means [Porter, 1997]. The Web-base instruction and training on the Internet are growing rapidly. The World Lecture Hall (http://www.utexas.edu/world/lecture) lists approximately 700 courses as being delivered by higher educational institutions via the Internet and it is growing daily [Parson]. If one surfs on the Web today, he/she may find many virtual universities, global schools, and on-line campuses, such as the Globewide Network Academy (http://uu-gna.mit.edu:8001/uu-gna/), Global Campus (http://www.acs.csulb.edu/gc/), Virtual on-line University (http://www.athena.edu/vou.html), and so on, which provide programs from a single course to all kinds of degrees. "College Online" [Duffy, 1997] provides a detailed directory of more than 400 undergraduate and graduate courses available online from accredited institutions and a complete listings of Web site addresses of colleges and universities. Meanwhile, many major corporations and companies, such as General Motors Corp., AT&T, and Hewlett-Packard, have begun to deliver Web-based instruction to their employees and other business professionals [Callaway, 1996; Distance learning].

As Web-based instruction becomes more and more prevalent, there are some researches undergoing on designing and creating interactive Web-based instruction [Brooks, 1997; Porter, 1997]. One of the benefits of the Web is the use of hypertext and hypermedia to link plain documents or multimedia information. Hypertext/hypermedia has been claimed to have a number of advantages in learning. Kearsley (1988) considered that "hypertext matches human cognition, in particular the organization of

memory as a semantics network in which concepts are linked together by association." The non-linear exploration of hypermedia could lead to a better understanding of the structure of a particular knowledge domain [Jonassen, 1988]. Hypertext is also considered as an active learning strategy since it requires readers to make decision about their reading [Brooks, 1997]. However, poorly organized hypertext links could easily turn into hyperchaos, where learners have too many choices and get lost [Barron, 1993]. Another exciting feature of the Web is its multimedia capability. The Web provides information in many different formats. Although text is still a popular way to transmit information, the Web may present information in other media, such as graphics, animations, audio or videos. Multimedia has been involved in education for many years. It can be used to show students things that are difficult to illustrate in text and display simulated experiments and operations for learning by experience. Through interaction with multimedia, students can increase attention span [Adams, 1996]. Multimedia can be used to customize instruction to suit the pace, learning style, and time and place constraints of the learner. It offers distance learners "an environment more closely matched to traditional learning" [Adams, 1996]. Although the Web can deliver multimedia information, some kinds of multimedia files, such as audio, video files, usually occupying a large size of disk space, are currently transferred very slowly due to the constraint of network bandwidth.

Although there are many courses and tutorials of almost every science and field available on the Web, no one covers IBM VisualAge for COBOL. Since VisualAge for COBOL is such a new development tool for not only COBOL programmer but also other

people who are interested in it, and the Web-based instruction could reach every one via the Internet, creating a Web-based tutorial for VisualAge for COBOL is necessary. Meanwhile, the above reviewing of literatures reveals that it is possible to create an appropriate Web-based tutorial for VisualAge for COBOL.

CHAPTER III

DESIGN AND IMPLEMENTATION

This chapter will illustrate in detail the design and implementation of the Web-based tutorial for VisualAge for COBOL. The tutorial is a part of the project — creating a curriculum for IBM certified VisualAge for COBOL object-oriented associate developer. The other parts of the project are about Object-Oriented COBOL, Visual Modeling Technique, and Visual Builder. The primary purpose of this tutorial is introducing users basic features of VisualAge for COBOL development environment and its components and instructing users to do object-oriented COBOL programming and create COBOL applications using VisualAge for COBOL.

The Contents of the Tutorial

The contents are very important for a tutorial. To determine what should be included in this tutorial, the author considered its place in the whole curriculum and referred to IBM VisualAge for COBOL Users' Guide for Windows, Visual Builder Users' Guide, and VisualAge for COBOL's online help, which come with IBM

VisualAge for COBOL Version 2.0.

As mentioned above, this tutorial is a part of the curriculum for IBM certified VisualAge for COBOL object-oriented associate developer. The curriculum is composed of four parts: Part I — Object-Oriented COBOL, introducing Object-Oriented COBOL language; Part II — Visual Modeling Technique, presenting the construction-from-parts paradigm and Visual Modeling Technique as a theoretical basis for object-oriented COBOL programming; Part III — Visual Builder, describing the features and functions of Visual Builder, a graphical user interface designer in VisualAge for COBOL; Part IV — VisualAge for COBOL, i.e. this tutorial, introducing VisualAge for COBOL development environment and its components. With these four parts, the curriculum provides users a comprehensive knowledge for object-oriented COBOL programming, including programming language, object-oriented programming theory, and application development tools.

According to the purpose of the tutorial and the features of VisualAge for COBOL, this tutorial was divided into three chapters. Chapter One, Introducing VisualAge for COBOL, briefly introduces some basic features of VisualAge for COBOL development environment, such as WorkFrame, Editor, Visual Builder, Debugger, and Performance Analyzer, and some concepts concerned in the application development, such as project and project parts. Chapter Two, Creating Applications, provides step-by-step instructions for creating an application using VisualAge for COBOL, including setting up a project, creating a graphical user interface, generating source code, debugging, analyzing and tracing execution, building, and running an application. In

Chapter Two, two examples are used for illustrating the procedure of application development with VisualAge for COBOL. One example, the Hello program, is used for explaining how to create a non-GUI application with VisualAge for COBOL. This program prompts a user to enter his/her name and then shows "Hello, name" on the screen. Another example, the Greeting application, is used for illustrating how to create a GUI application with VisualAge for COBOL. This application has a graphical user interface — a small window with a text label, an entry field, a Greet Me button and an Exit button. The text label prompts a user to type his/her name into the entry field. When the user clicks the Greet Me button, a greeting "Hello, name" is displayed in the window. When the user clicks the Exit button, the application terminates. Chapter Three, Using VisualAge Tools, describes in detail various functions and usages of each VisualAge for COBOL tool, including WorkFrame, Editor, Debugger, and Performance Analyzer.

The following are the contents of the tutorial:

- Chapter One: Introducing VisualAge for COBOL
  - Project
  - WorkFrame
  - COBOL Editor
  - Visual Builder
  - Debugger
  - Performance Analyzer

- Chapter Two: Creating Applications

- Setting up a Project
  - Creating a project
  - Registering the iwp file type
  - Customizing the project
- Creating a Non-GUI Application
  - Creating COBOL source files
  - Editing source code
- Creating a GUI Application
  - Designing a graphical user interface
  - Invoking Visual Builder
  - Adding and customizing parts
  - Defining features
  - Generating features source
  - Connecting parts
  - Generating code
- Building the application
- Running the application
  - Starting the application
  - Specifying environment variables
- Debugging
  - Building the application for debugging
  - Invoking the Debugger

        - Setting breakpoints
        - Monitoring and changing data
        - Executing the program
        - Ending the Debugger
    - Analyzing and Tracing Execution
        - Preparing for performance analyzing
        - Invoking the Performance Analyzer
        - Generating a trace file
        - Viewing the trace file
        - Exiting the Performance Analyzer

- Chapter Three: Using VisualAge Tools
    - Using WorkFrame IDE
        - Opening a WorkFrame project
        - Creating a project
        - Creating a new file
        - Running a tool
        - Copying and moving a project part
        - Deleting project parts
        - Setting options for actions
        - Setting and deleting environment variables
        - Filtering parts

    - Compiling, linking, building, and running
- Using VisualAge Editor
    - Starting the Editor
    - Creating a new file
    - Opening an existing file
    - Saving a file
    - Editing a file
    - Embedding another file into the current document
    - Finding and replacing text
    - Finding a line
    - Using marks in a file
    - Multiple document views
    - Inserting SQL statements
    - Calling a CICS program
- Using VisualAge Debugger
    - Starting the Debugger
    - Preparing for debugging
    - Debugger windows
    - Running a program
    - Halting and restarting
    - Stepping through a program
    - Setting breakpoints

    - Listing and sorting breakpoints
    - Modifying breakpoints parameters
    - Disabling and enabling breakpoints
    - Deleting breakpoints
    - Displaying debugging information
- Using Performance Analyzer
    - Preparing for performance analyzing
    - Starting the Performance Analyzer
    - Creating a trace file
    - Diagrams for a trace file
    - Opening a trace file in a diagram
    - Windows of the Performance Analyzer
    - Call Nesting Diagram
    - Dynamic Call Graph
    - Executing Density Diagram
    - Statistics Diagram
    - Time Line Diagram

The Organization of the Tutorial

According to its contents and the features of hypertext, the tutorial is designed as a series of HTML pages linking together. Each page covers one topic or less. The organization of topics is based on their logic progression, as shown in the above contents section.

There is a main menu page that contains links to three chapters at the beginning of the tutorial, whereas each chapter includes a chapter heading page briefly introducing its contents and including index links to its topics and sub-topics. Meanwhile, a summary page included in each chapter to summarize its contents also contains links to its main topics. Each page is linked to its previous page and next page, as well as the first page and the last page of the tutorial. When a topic refers to other topics, hyperlinks are set enabling users to browse corresponding topics. In addition, to enable users to refer to the other three parts of the curriculum, there are links to them set in the first page of the tutorial. Figure 2 shows links among pages of the tutorial and links among the four parts of the curriculum.

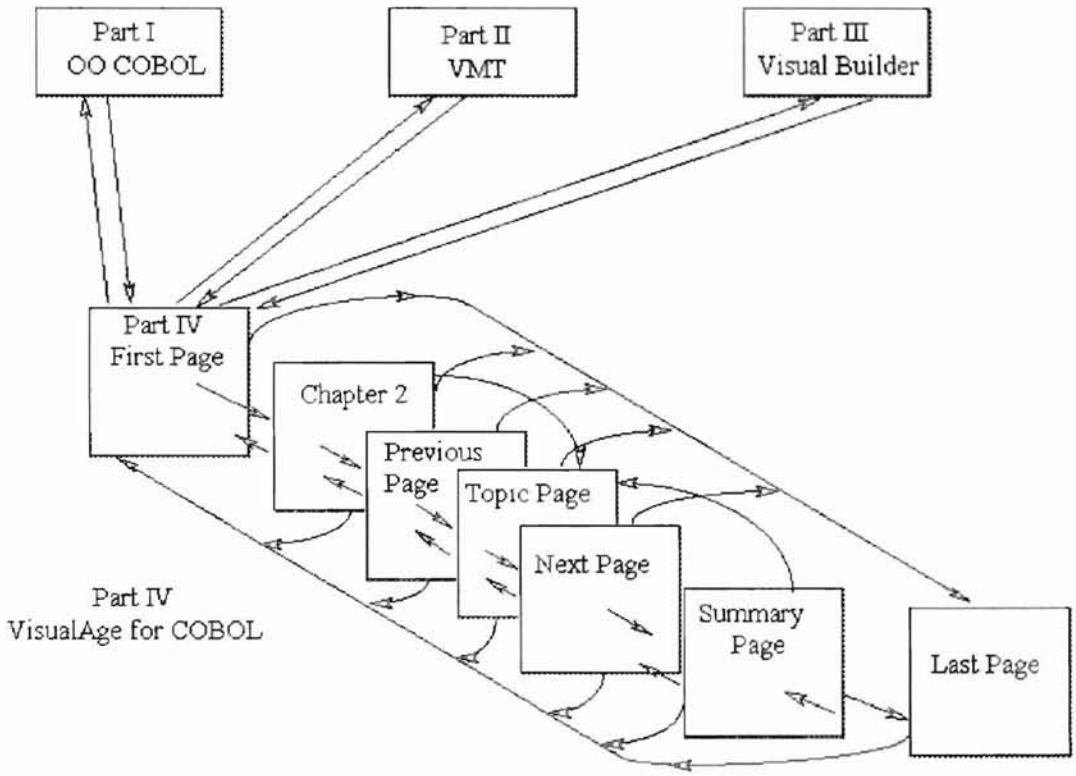


Figure 2: Links among pages in the tutorial and
links among four parts of the curriculum

Development Systems and Tools

Asymetrix ToolBook II Instructor 5.0 [Asymetrix, 1996] was used for developing the tutorial. ToolBook II Instructor is an authoring system for creating interactive courseware. It provides a sophisticated array of tools, such as widgets, templates, and Book Specialists, simplifying and automating courseware design and development. ToolBook II uses the metaphor of a book as the basis for its application. A ToolBook II application may consist of one or more books. A book in ToolBook II is divided into pages representing different screens of the application. There could be several objects, such as text fields, buttons, and graphics, on a page. If some objects are shared among several pages, they could be placed on a background, which is common to those pages. With these concepts, developing a courseware is similar to compiling a textbook. ToolBook II also provides a utility for converting a ToolBook II book into a set of HTML pages. This Internet capability eases authoring of Web-based courseware.

The tutorial was initially designed and created using ToolBook II Instructor 5.0 on a PC with Windows 95. After the book for the tutorial was built and tested in the ToolBook II system ensuring that the interactive features work properly, it was exported into pages of HTML format through the built-in HTML format converting function of ToolBook II. Then, each HTML page was modified with the Page Composer of Netscape Communicator 4.0 for adjusting the positions of tables and the sizes of pictures. The tutorial of a series of HTML pages was viewed and tested via a Web browser, such as Microsoft Internet Explorer and Netscape Navigator on a PC. To ensure every feature works well, the tutorial in HTML format was also transferred to a Web server, and

retrieved through the Web.

IBM VisualAge for COBOL version 2.0, which is installed on a PC with Windows NT 4.0, was used for developing examples used in the tutorial.

To provide users more perceptual knowledge on VisualAge for COBOL, many images of graphical user interfaces of various components of VisualAge for COBOL were included in the tutorial supporting textual instructions. HyperSnap-DX 3, a screen capture software of Hyperionics (http://www.hyperionics.com), was used to take pictures of graphical user interfaces of VisualAge for COBOL.

The Interfaces of The Tutorial

The tutorial was created as a book including a title page, a main contents page, three chapter headings, and general contents pages. To keep consistency of the book and lessen stress of users' eyes, the green marble template was chosen for the background of all pages. Several objects were set on the background: four navigation buttons enabling users to freely go to the previous page, the next page, the first page, and the last page, respectively; a text field for either the title of the tutorial or the title of a chapter; another text field beneath the title text field for the subtitle or topic, as shown in Figure 3.

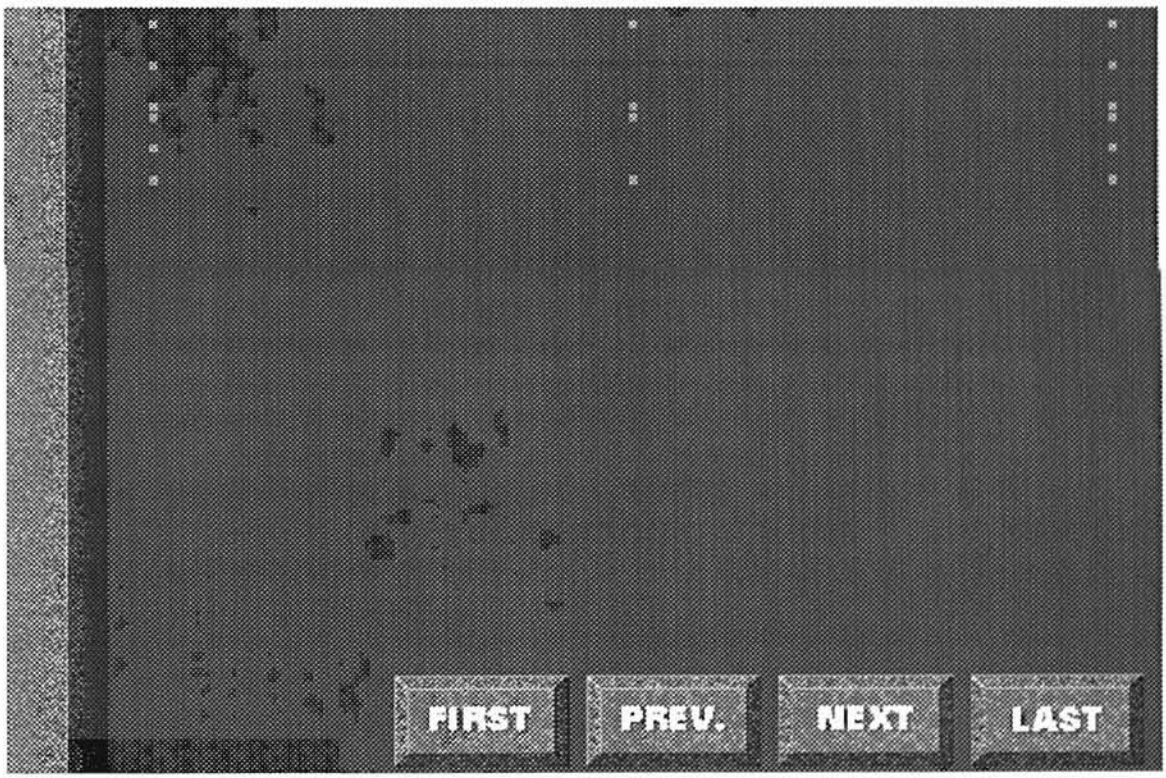


Figure 3. The background of the tutorial

In addition to the objects on the common background, each page applied one or more objects that cooperate for the illustration. The title page used one more text field for the information of what the tutorial is about and how to use it (see Figure 4). The main contents page contained three buttons for users going to the corresponding chapter through a simple point-and-click and a text field for a brief explanation (see Figure 5). Each chapter heading page had a text field displaying topics and sub-topics of the chapter. The words or phrases for the topics and sub-topics were set as hot words linking to the corresponding pages, with which users could choose what they want to look at first or later (see Figure 6).

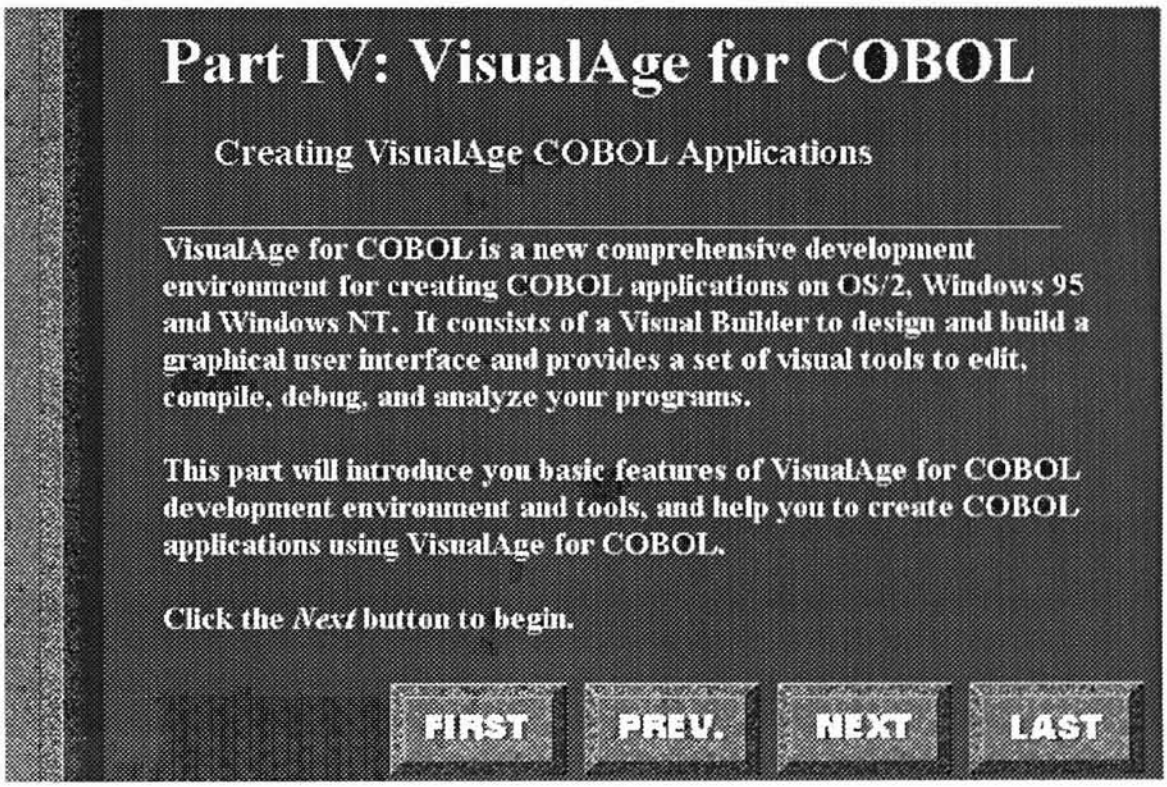


Figure 4. The title page of the tutorial

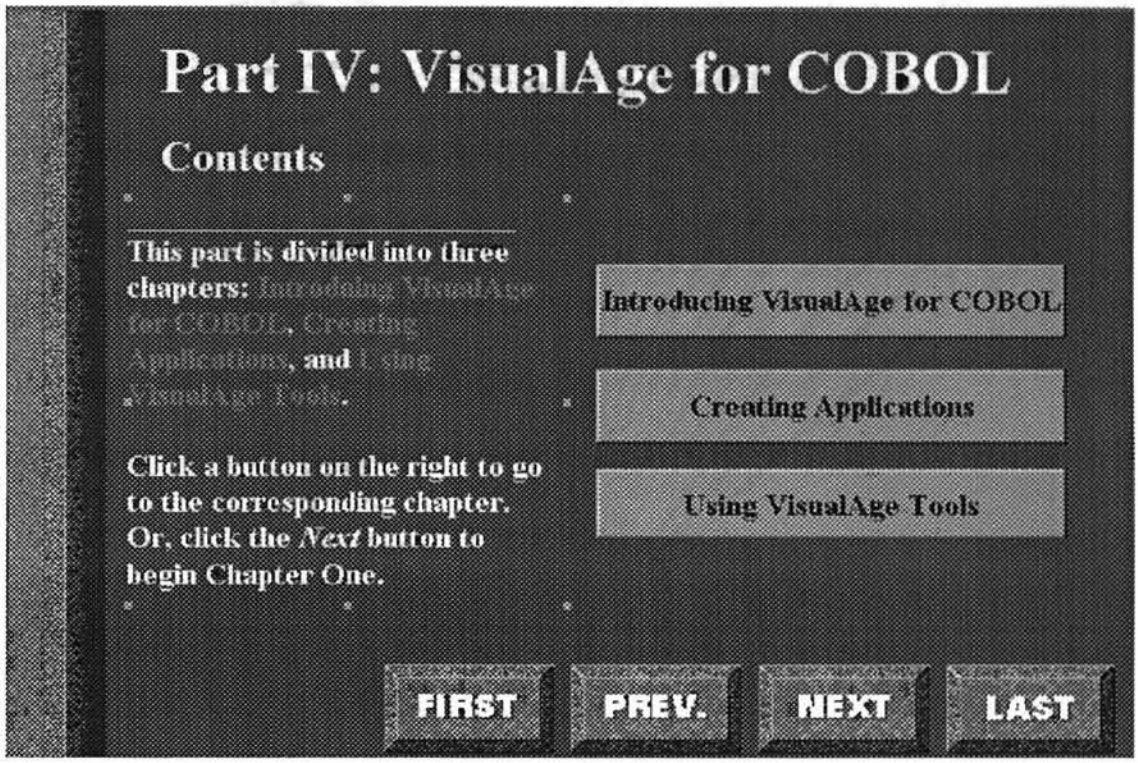


Figure 5. The main contents page of the tutorial

Each general contents page mainly contained one or more text fields and a graphic place holder for textual illustrations and a relevant image, respectively (see Figure 7). Hot words (hyperlink) were set in some pages when other concepts and usages were referred to enabling users to learn by their own cognition flow. The last page of each chapter was reserved as a summary page that summarizes the contents of the chapter.

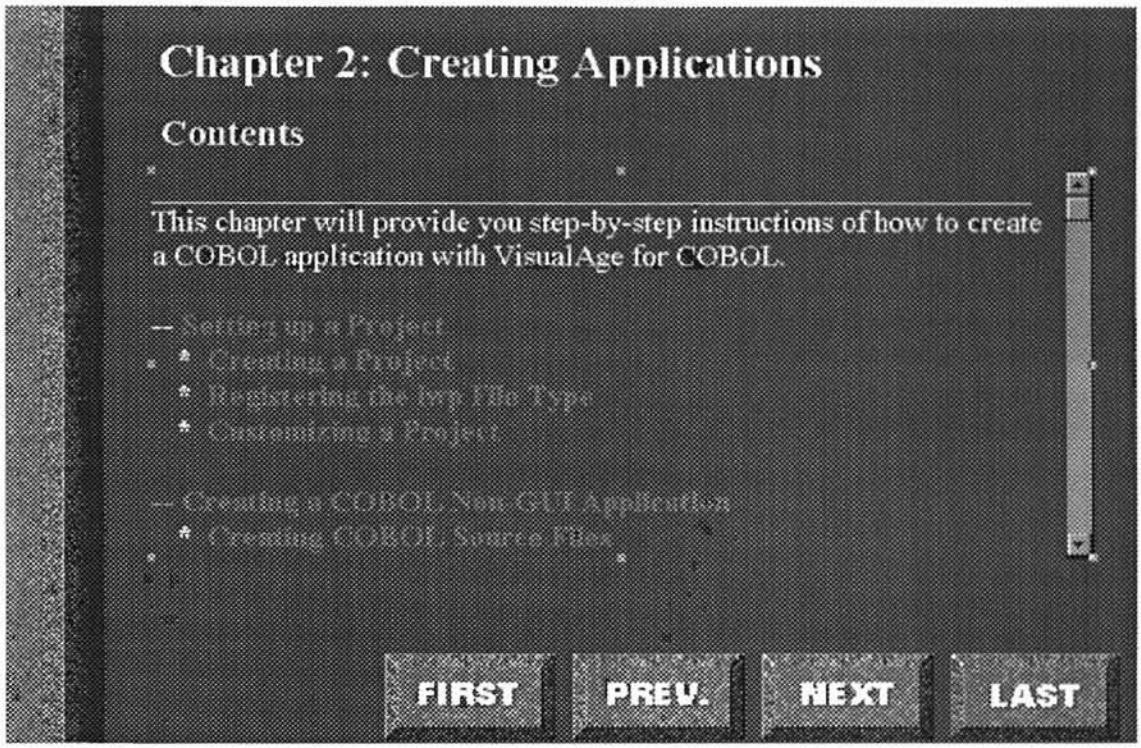


Figure 6. A chapter heading of the tutorial

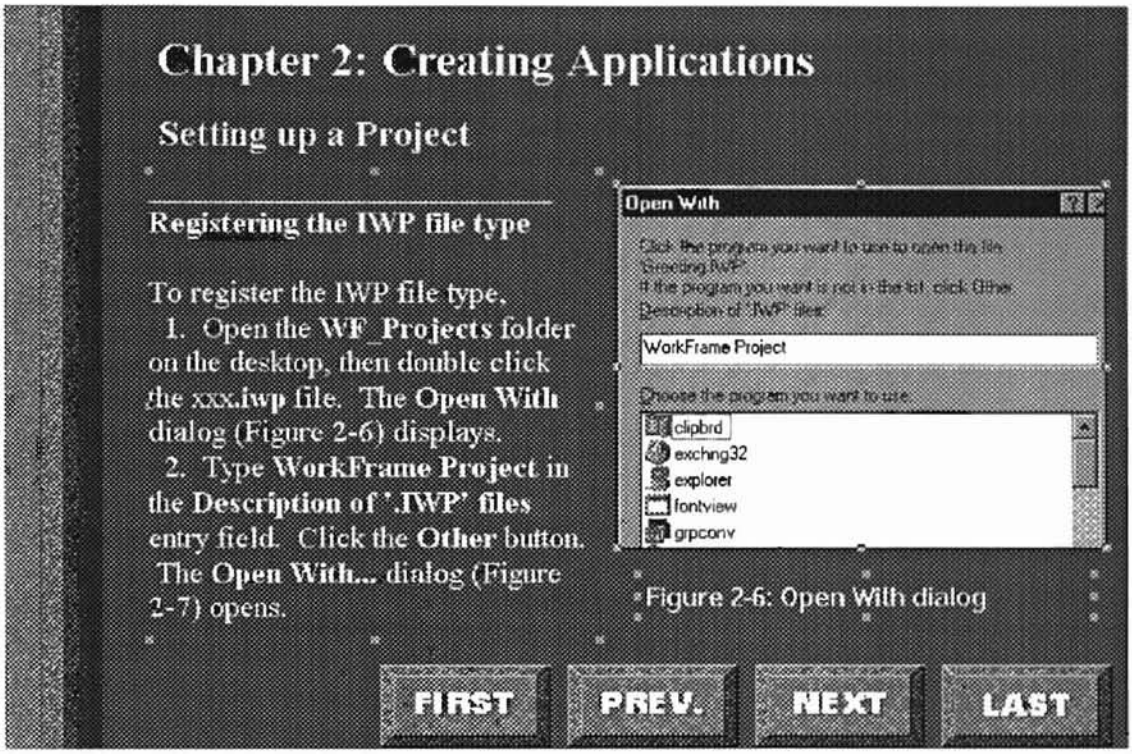


Figure 7. A general page of the tutorial

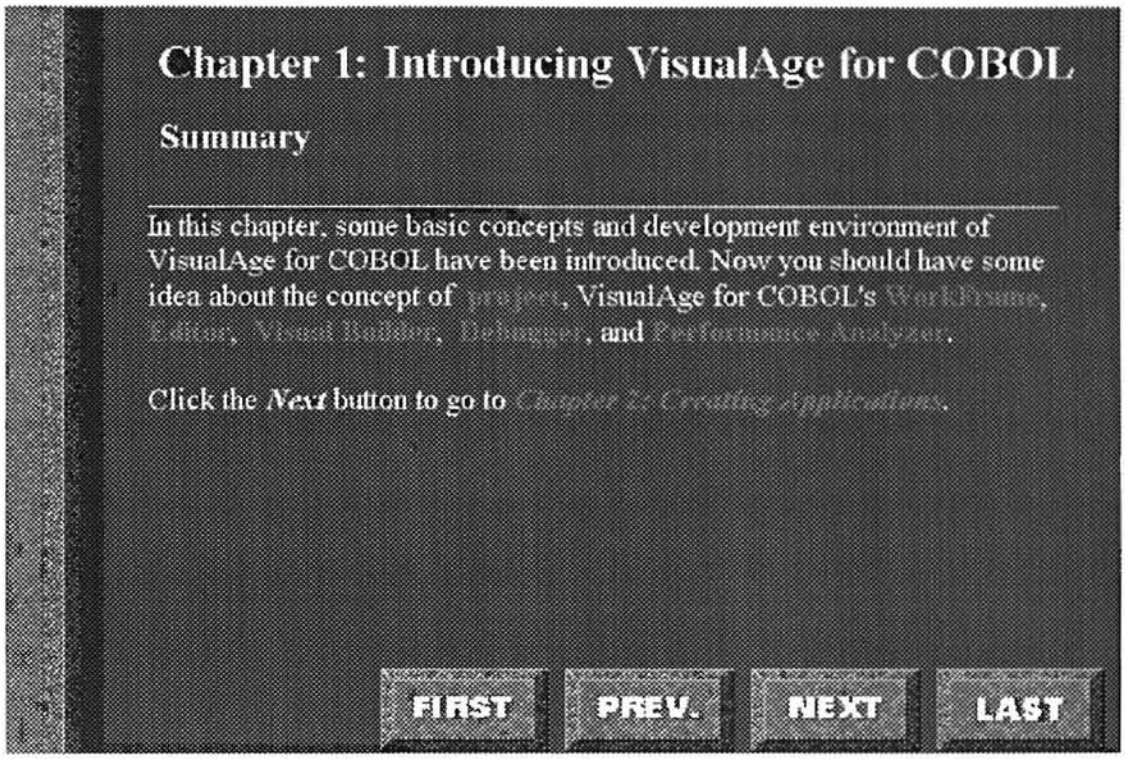


Figure 8. A summary page of the tutorial

Implementation

The tutorial was initially created as a book with ToolBook II Instructor. The first step was specifying the structure for the book using the Internet Content Book Specialist of ToolBook II Instructor. During this step, the caption shown by Internet browser was set as IBM VisualAge for COBOL Part IV; the green marble template was chosen as the layout of the book providing a consistent graphical look for the book and providing simple options for page labeling and navigation; the page size was defined in six by four inches; the initial book structure was specified to include a title page, a main menu page, chapter headings and general content pages; and the book was set to have three chapters and twenty content pages for each chapter.

After the book was created with the Book Specialist, it consisted of a series of pages, on which several standard objects, such as navigation buttons and record fields, had been set. With this foundation, the contents of the tutorial could be inserted into pages. The next step was compiling the book. During this step, the title, subtitle and textual illustrations were added into the corresponding record fields of each page; more objects such as record fields for graphic labeling and graphics were placed onto a page according to the requirement of the content; the positions and sizes of the objects were adjusted; more pages were added into Chapter Two and Chapter Three whereas some blank pages were deleted from Chapter One; and hot words (hyperlinks) were set, as required. Due to the limit of the page size, there was usually one record field for the text illustration with no more than two graphics for a topic per page. When a topic required more graphics, several pages were used. Since the difference among the contents of three

chapters, eight content pages were used for Chapter One whereas sixty-three and fifty-seven content pages for Chapter Two and Chapter Three, respectively. The completed title page, main contents page, chapter heading page, and general content page are shown in Figure 4 to 7.

When the book for the tutorial was completed in ToolBook II's author level, it was tested at ToolBook II's reader level to ensure that all features work properly. Then all pages of the book were converted to HTML pages with the built-in HTML transfer utility of ToolBook II and viewed in a Web browser. Microsoft Internet Explorer 2.0 and 3.0, as well as Netscape Navigator 2.0, 3.0 and 4.0 were used to view and test those converted HTML pages. In converted HTML pages, tables were used to posit the objects such as text, graphics, and navigation buttons. Due to the table implementation problem of the Netscape Navigator, some pages were shown in the browser with displaced objects. Figure 9 shows a page in ToolBook II environment whereas Figure 10 shows the same page displayed in Netscape Navigator 4.0 with displaced text, graphic and navigation buttons. To solve this problem, most pages were revised with the Composer of Netscape Communicator 4.0 for setting the column width of the table to equal (Figure 11). Moreover, since a page in ToolBook II environment is not big enough to display both text and a whole picture, most pictures were shrunk to fit into a small frame whereas the text was displayed in a text field with a scroll bar. After a page was transferred to a HTML page, the picture on the page was still displayed as the same size as that on the original ToolBook II page. A shrunk picture is distorted. With the Composer of Netscape Communicator 4.0, the size of an image could be adjusted and restored to its original

size. So the pages with graphics were also modified with the Composer of Netscape Communicator 4.0 for adjusting the sizes of the graphics to their original sizes.

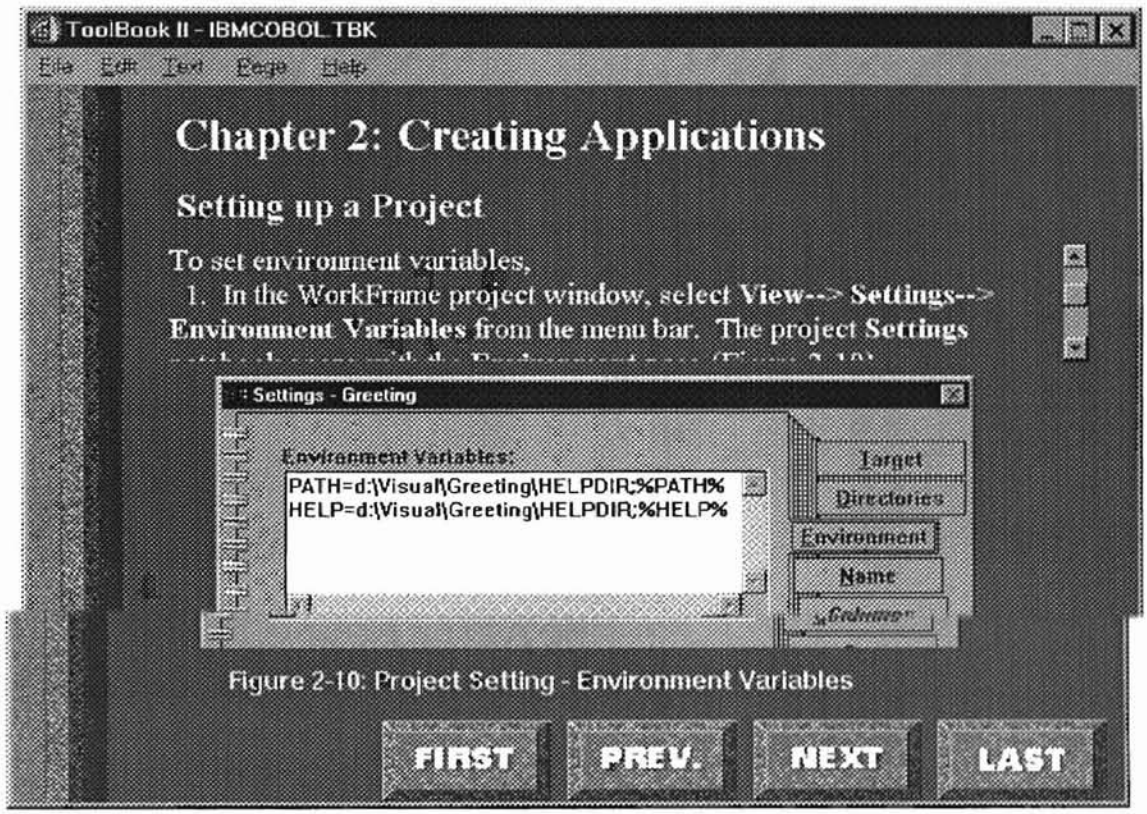


Figure 9. A page in ToolBook II environment

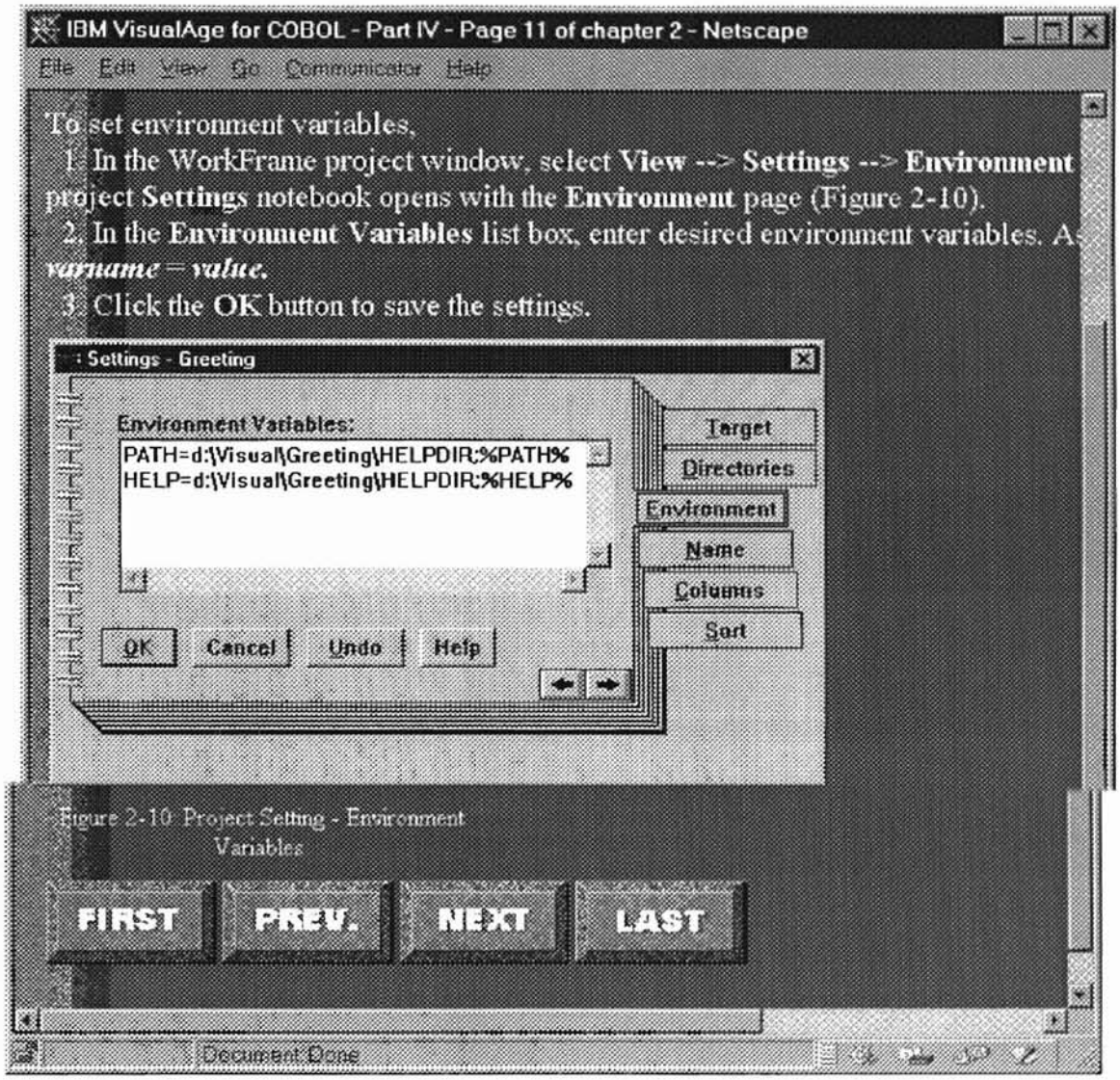


Figure 10. The same page shown in Netscape Navigator with displaced text, graphic, and navigation buttons

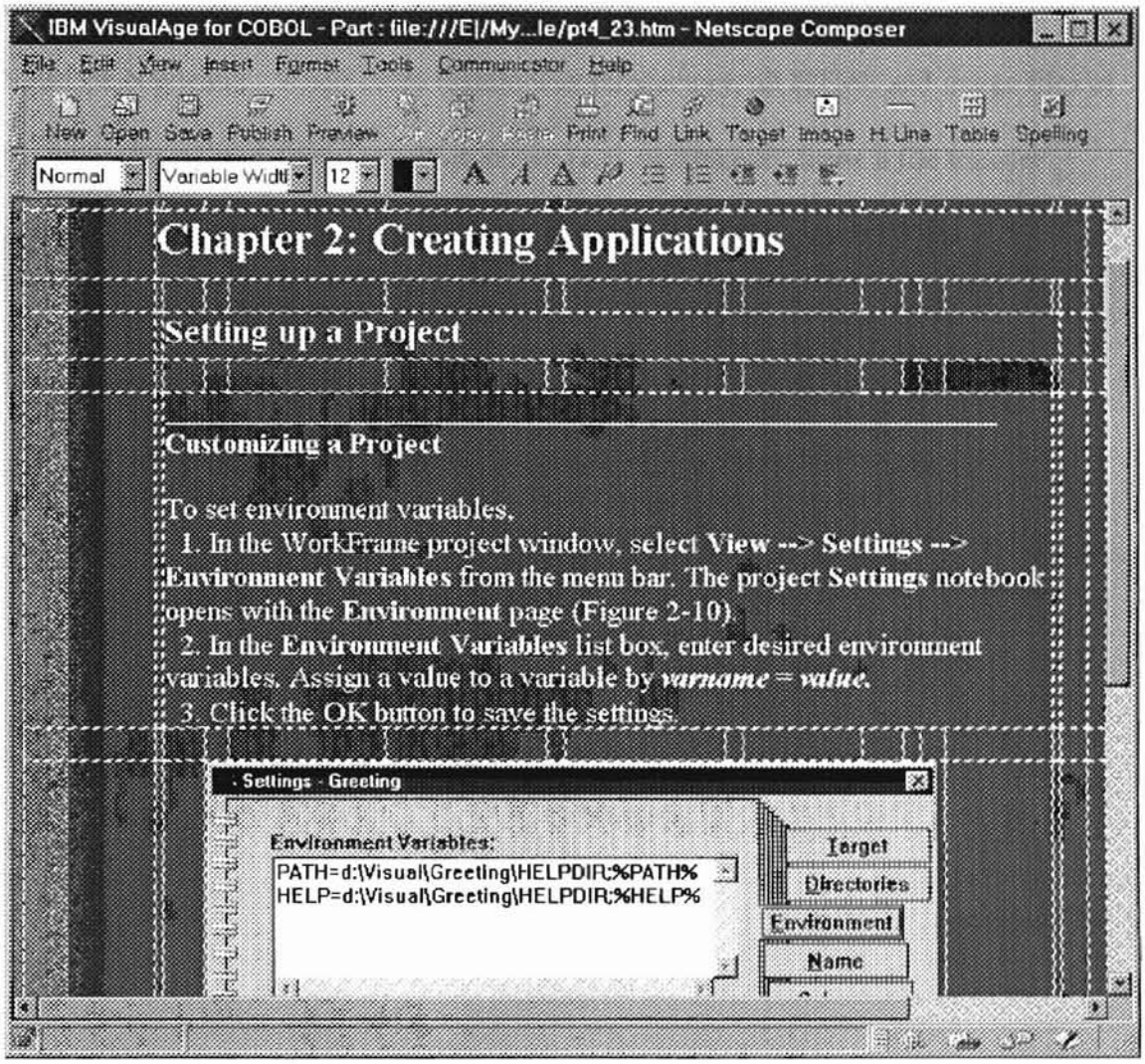


Figure 11. The page modified with the Composer of Netscape Communicator 4.0 for setting the column width of the table and restoring the image size

After HTML pages of the tutorial were modified with the Composer of Netscape Communicator 4.0, they were viewed and tested again in Microsoft Internet Explorer and Netscape Navigator. When every feature was satisfactory, all HTML and image files of the tutorial were transferred to a Web server (a.cs.okstate.edu) using a file transfer protocol (FTP) utility, then the tutorial could be retrieved through the Web.

CHAPTER IV

RESULTS AND CONCLUSION

Results and Discussions

The completed tutorial is composed of one hundred thirty-three separate yet inter-linked HTML documents (Web pages) and ninety-four image files used in the Web pages. All files for the tutorial are temporarily put on a Web server computer of Computer Science Department. The temporary address for the Web pages of the tutorial is http://a.cs.okstate.edu/~wmei. Figure 12 to Figure 18 display some selected Web pages of the tutorial. To get the best result, users should choose Microsoft Internet Explorer 3.0 or above, or Netscape Communicator 4.0 to browse the Web pages of the tutorial. Although most HTML pages were modified with the Composer of Netscape Communicator 4.0 for adjusting table settings, some pages are still shown with displaced objects in Netscape Navigator 3.0.

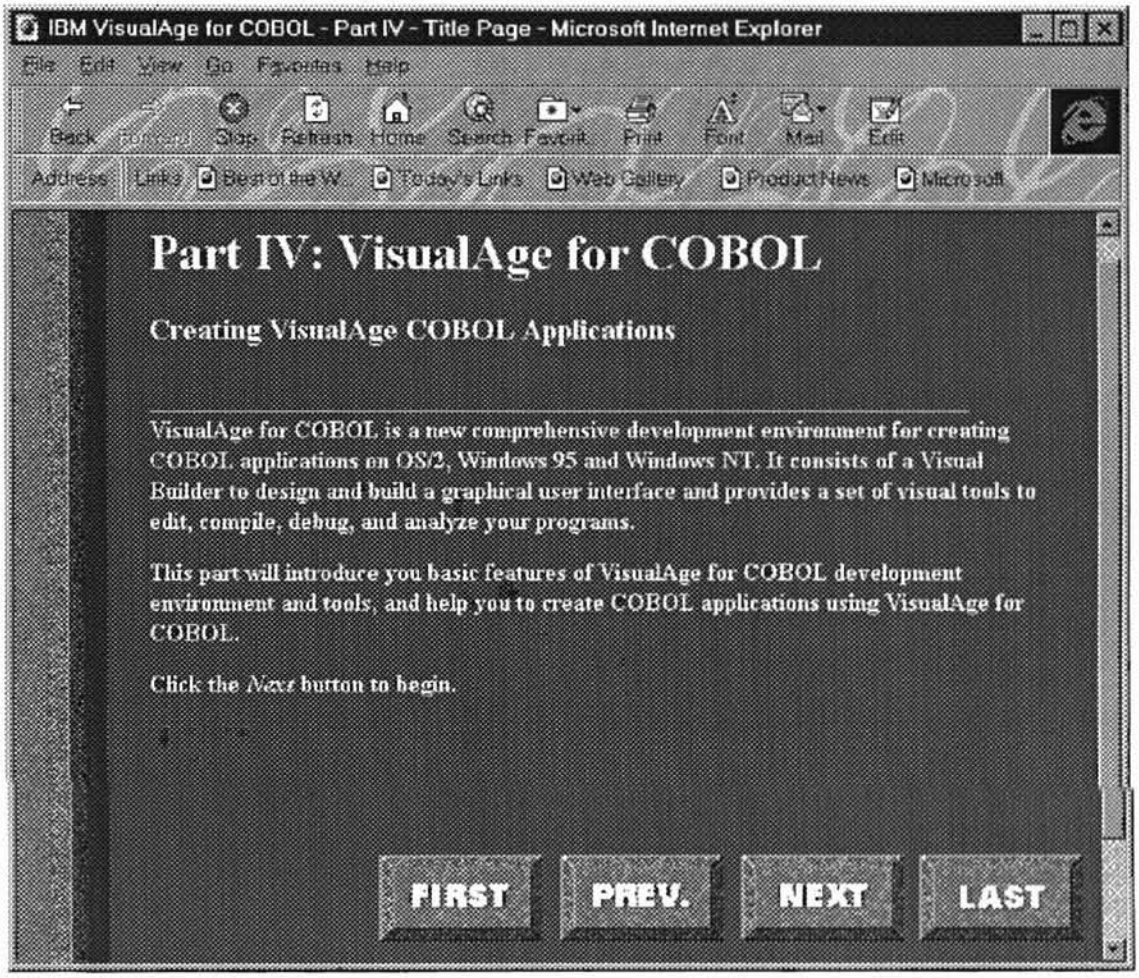


Figure 12. A selected Web page of the tutorial
— the title page

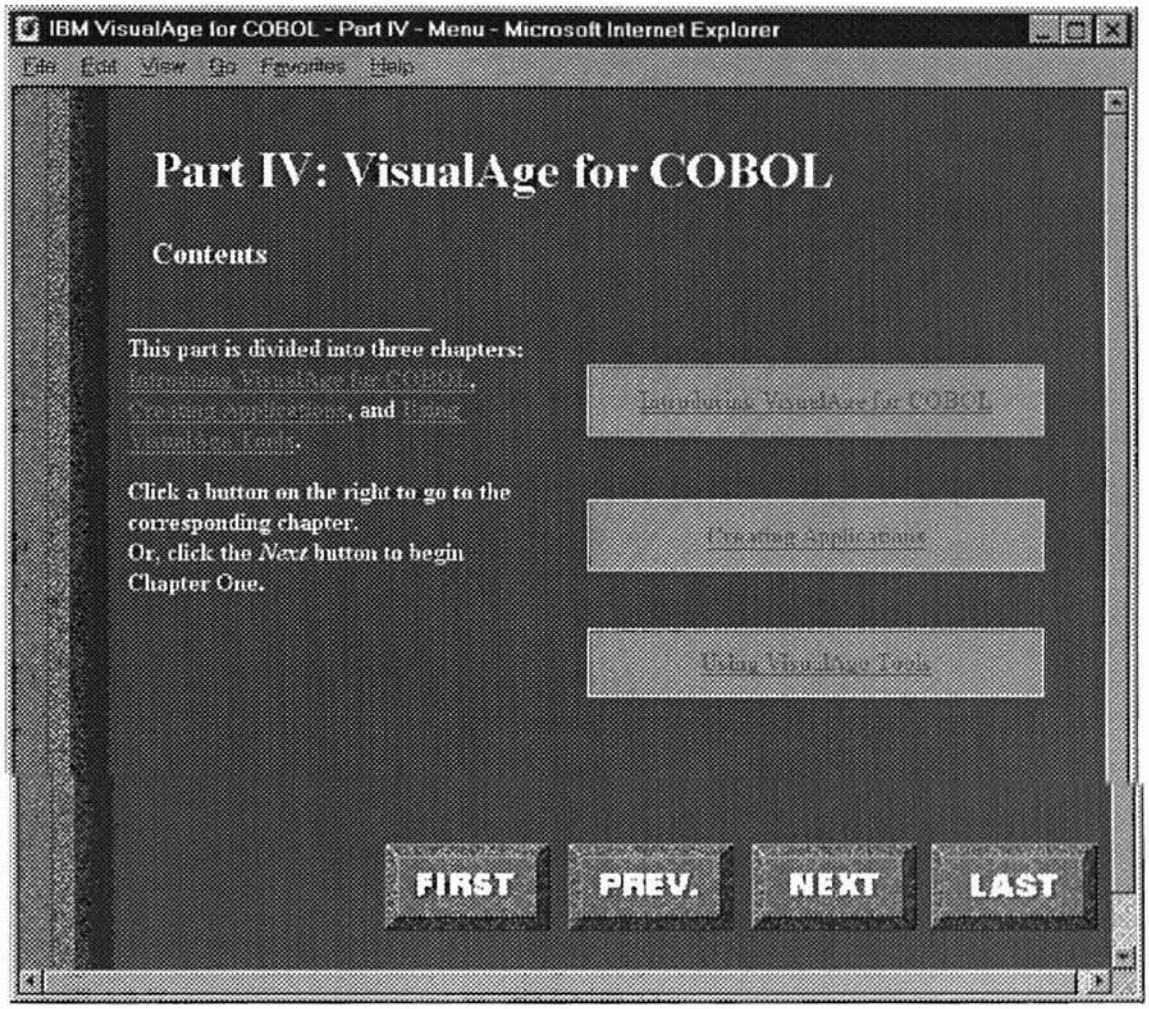


Figure 13. A selected Web page of the tutorial
— the main menu page

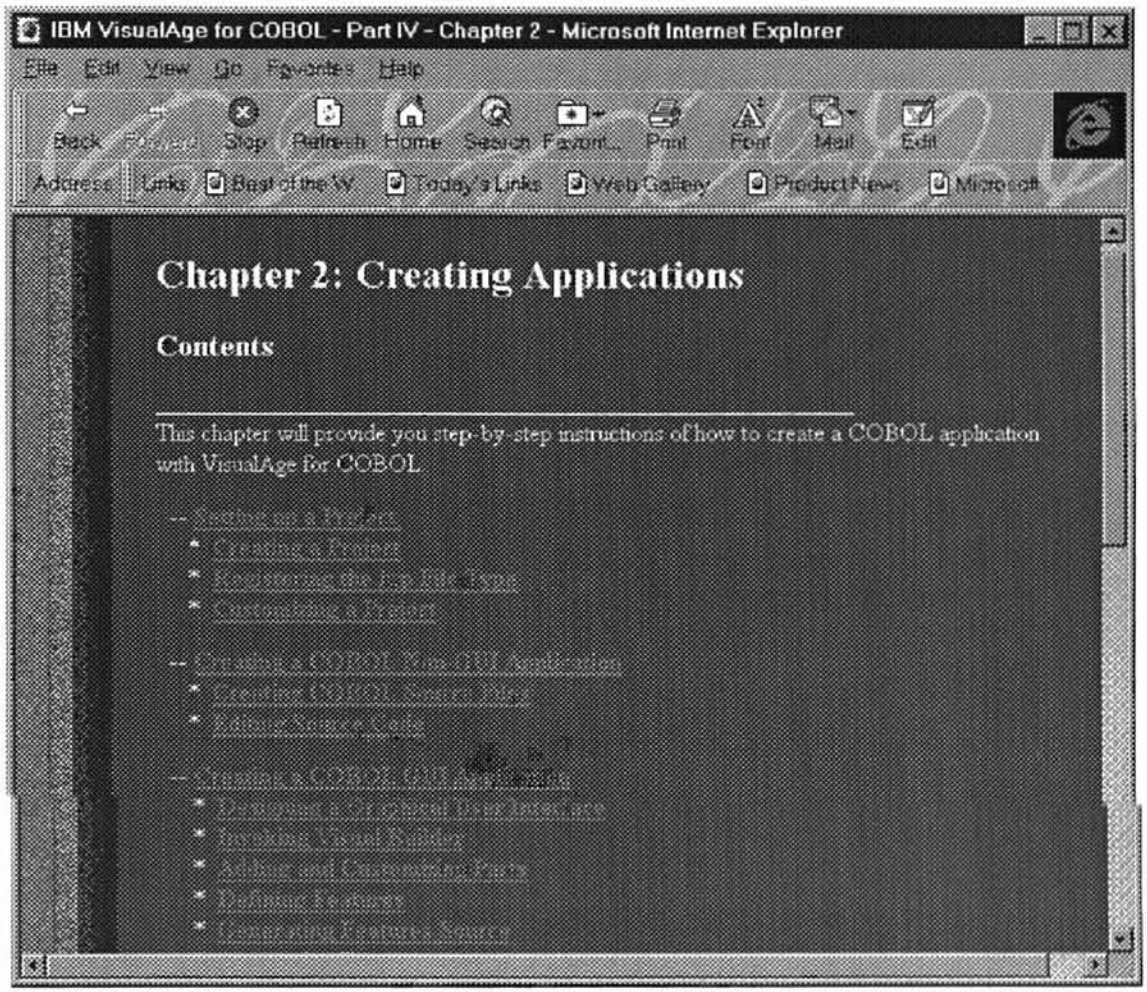


Figure 14. A selected Web page of the tutorial
— a chapter heading page

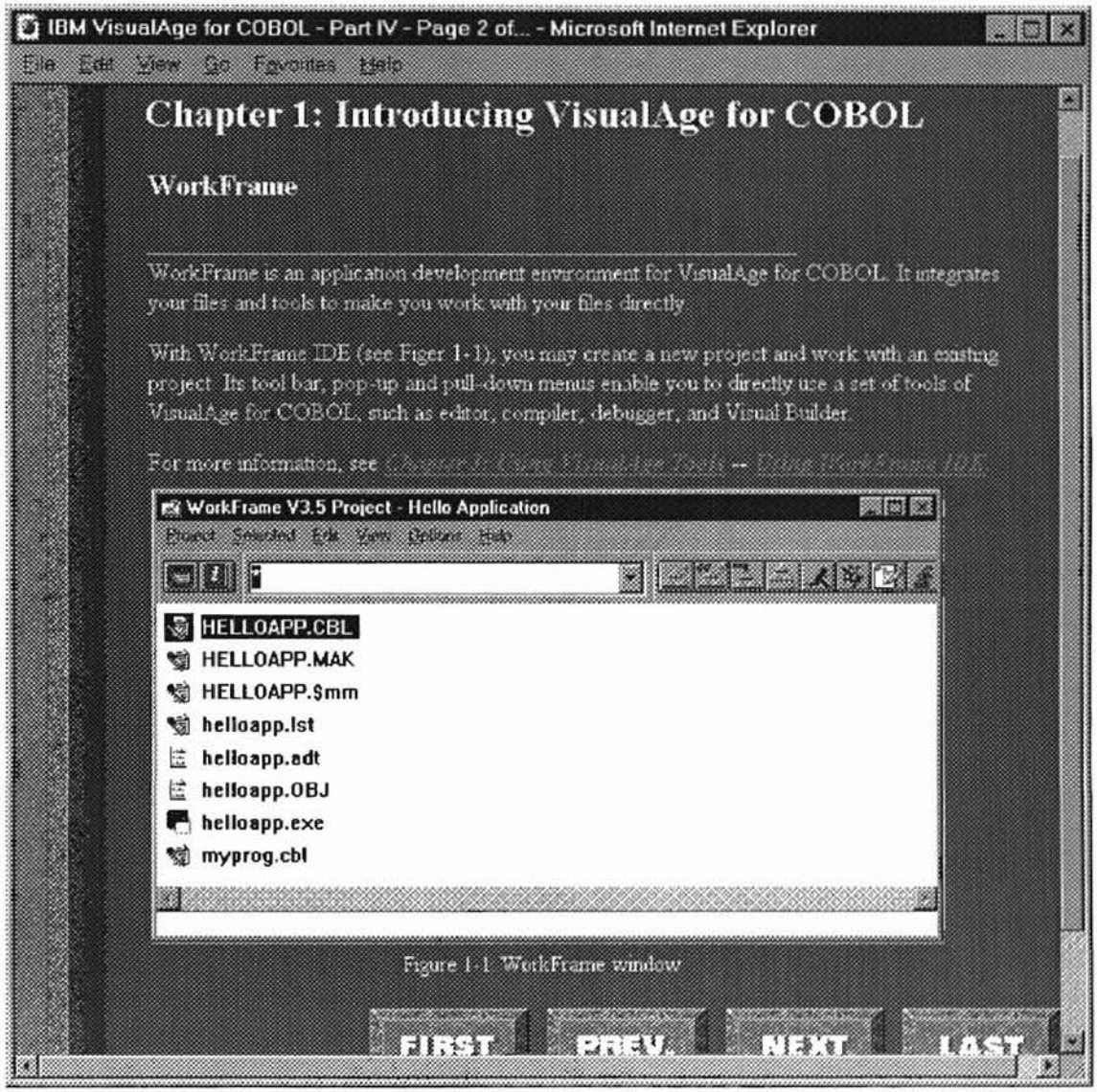


Figure 15. A selected Web page of the tutorial
— a general content page in Chapter One

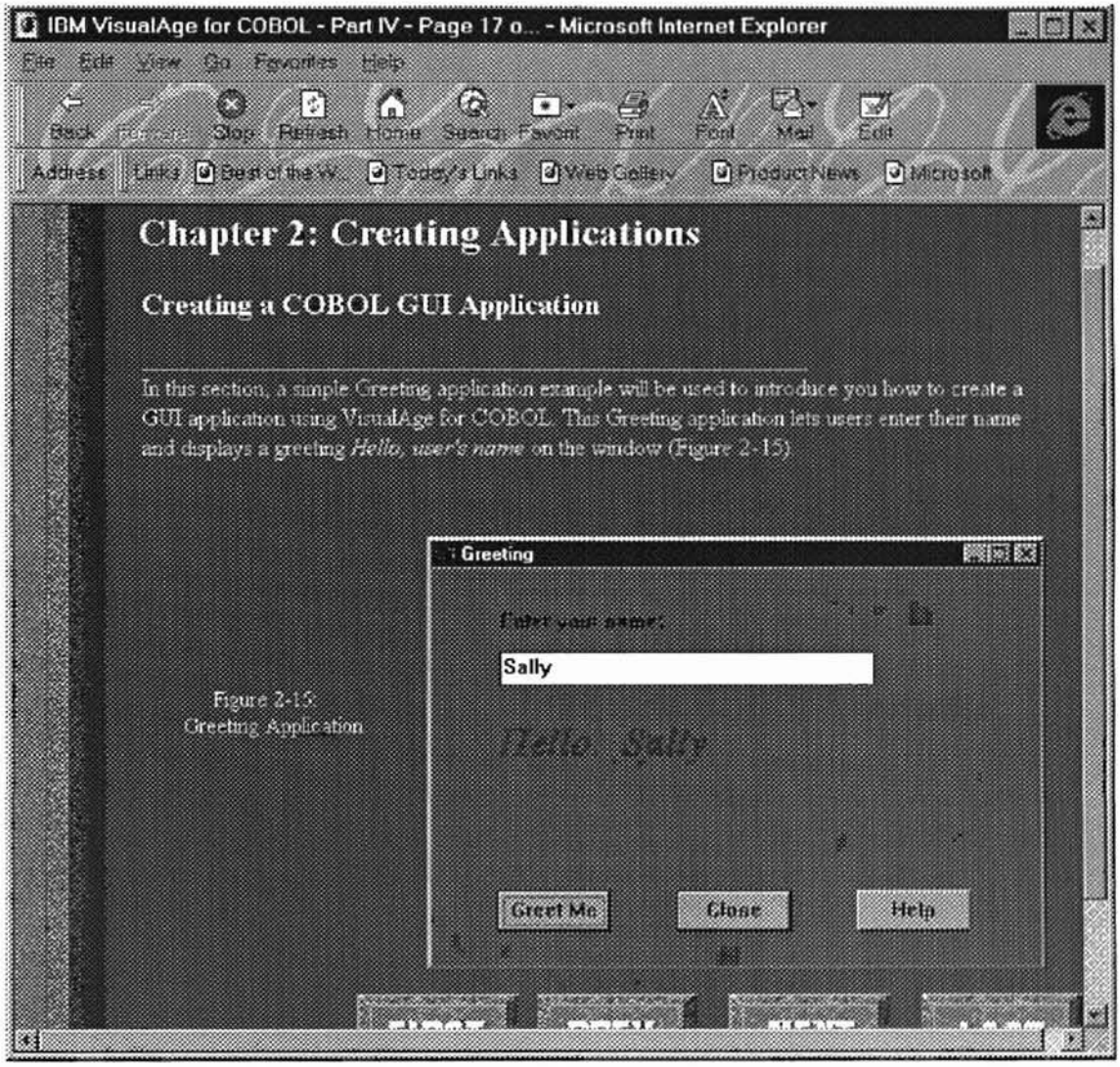


Figure 16. A selected Web page of the tutorial
— a general content page in Chapter Two

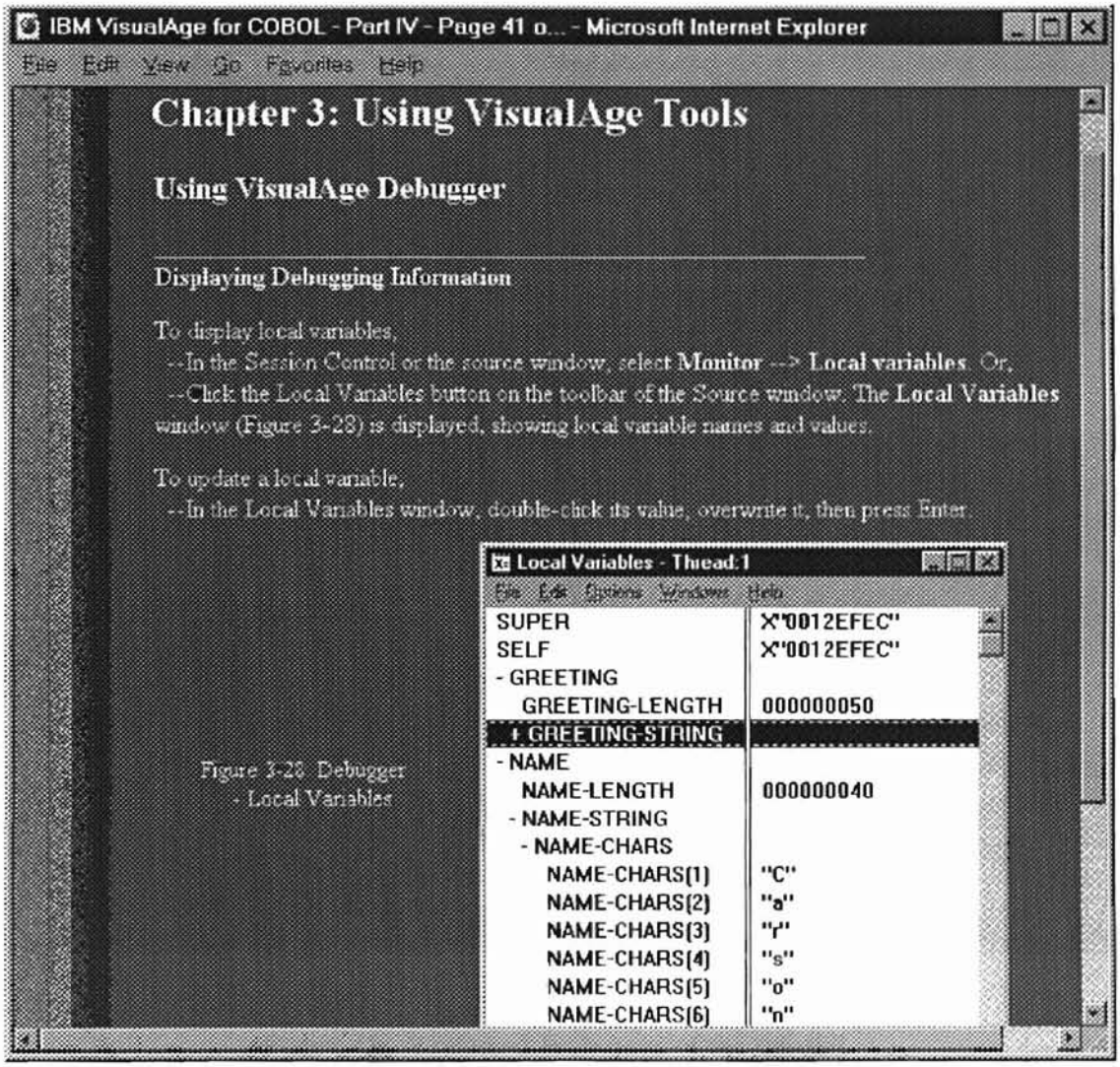


Figure 17. A selected Web page of the tutorial
— a general content page in Chapter Three

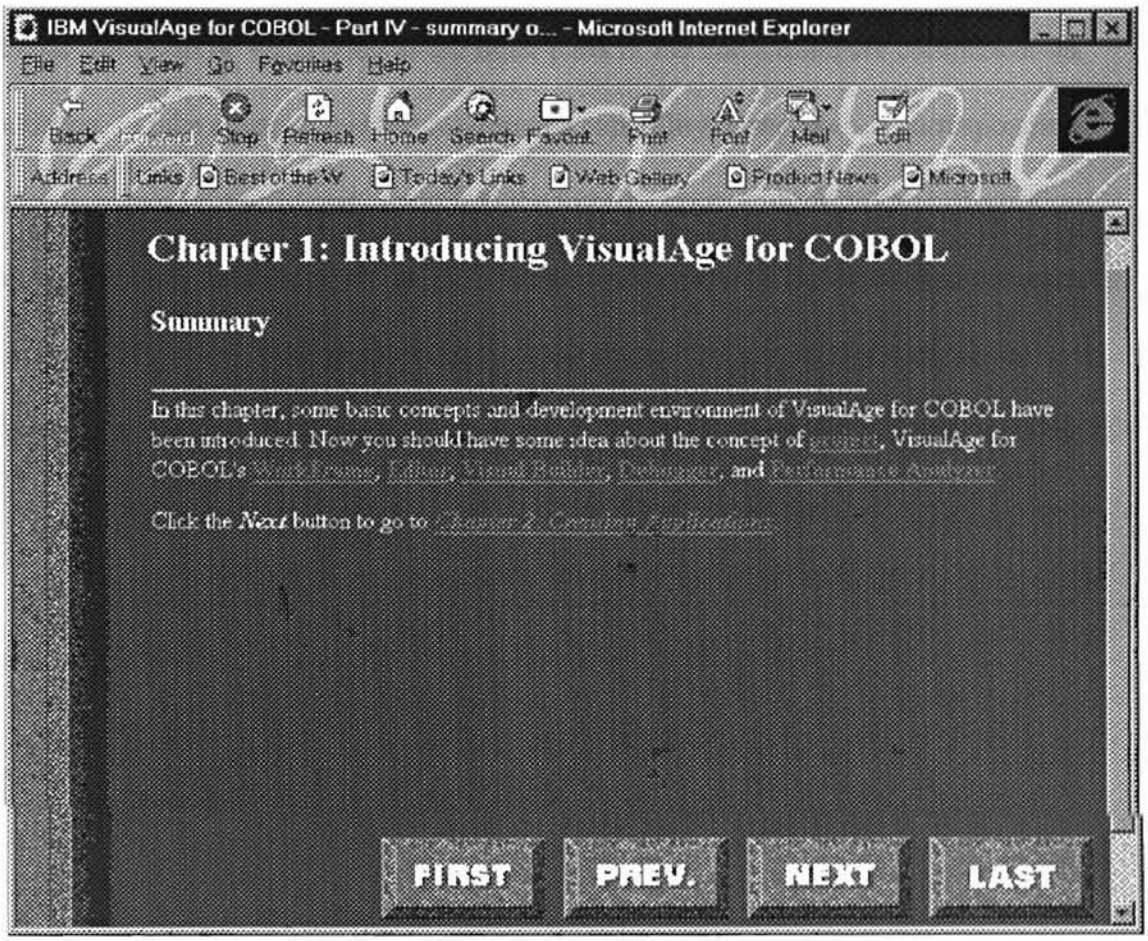


Figure 18. A selected Web page of the tutorial
— a summary page

Since the other three parts of the curriculum have not been completed when this thesis is written, the links in this tutorial that link to the other parts have not been set yet. Those links will be set as soon as the other parts of the curriculum are completed. In addition, to take advantage of the multimedia capability of the Web, a corresponding narration may be considered to add into the tutorial along with the textual illustrations and graphics to hold the users' attention and involve more of their senses in the learning process. Moreover, a management system for controlling access to the curriculum and handling security should be established on the Web server computer for the curriculum.

Conclusion

In this thesis, a Web-based tutorial for VisualAge for COBOL was developed to introduce the basic features of VisualAge for COBOL development environment and provide step-by-step instructions for creating object-oriented COBOL applications using VisualAge for COBOL. The tutorial is composed of a series of HTML pages that could be posted on the Web, and thus could be browsed by a large amount of learners via the Internet. In this tutorial, many pictures of the graphical user interfaces of VisualAge for COBOL were used along with the textual instructions to enrich the power of illustration; hyperlinks were set to provide users a flexibility during studying.

This tutorial was initially created with ToolBook II Instructor 5.0 and then converted to HTML files using the built-in HTML converting utility of ToolBook II. This method is very effective for authoring a Web-based courseware since authors do not need to write tedious HTML tags.

BIBLIOGRAPHY

Adams, E., Carswell, L., Ellis, A., Hall, P., Kumar, A., Meyer, J. & Motil J. (1996). Interactive multimedia pedagogies: Report of the working group on interactive multimedia pedagogy. *Integrating Technology into Computer Science Education, ACM SIGCSE, SIGCSE Bulletin*, Vol. 28, Special Issue, pp. 182-191.

Alexander, S. (1995). Teaching and learning on the World Wide Web. http://www.scu.edu.au/sponsored/ausweb/ausweb95/papers/education2/alexander.

Arranga, E. C. (1996, March 11). COBOL gets with objects. *Informationweek*, No. 570, pp. 70-78.

Arranga, E. C. (1996, March 11). The tower of COBOL. *Informationweek*, No. 570, p. 71.

Asymetrix Corp. (1996). *A Guide to Creating Interactive Courses: Asymetrix ToolBook II Instructor.*

Barron, A. E. & Orwig, G. W. (1993). *New Technologies for Education: A Beginner's Guide*. Englewood, CO: Libraries Unlimited, Inc.

Barron, B., Ellsworth, J. H., & Savetz, K. M. (Ed's). (1995). *The Internet Unleashed 1996.* Indianapolis, IN: Sams.net Publishing.

Bates, A. W. (1995). *Technology, Open Learning and Distance Education*. London: Routledge.

Berge, Z. L. & Collins, M. P. (Eds.) (1995). *Computer Mediated Communication and the Online Classroom. Volume III: Distance Learning.* Cresskill, NJ: Hampton Press, Inc.

Brooks, D. W. (1997). *Web-Teaching: A Guide to Designing Interactive Teaching for the World Wide Web*. New York, NY: Plenum Press.

Callaway, E. (1996, December 12). The learning Web. *PC Week* (online), http://www.zdnet.com/pcweek/builder/1209/09learn.html.

Chapin, N. (1997). *Standard Object-Oriented COBOL.* New York, NY: John Wiley & Sons, Inc.

Chorafas, D. N. (1997). *Visual Programming Technology.* New York, NY: McGraw-Hill.

December, J. (1994). *The World Wide Web Unleashed.* Indianapolis, IN: Sams Publishing.

Distance education at a glance. http://www.uidaho.edu/evo/disl.html.

Distance learning fact sheet. http://www.usdla.org/dl.html.

Duffy, J. P. (1997). *College Online: How to Take College Courses without Leaving Home.* New York, NY: John Wiley & Sons.

Dwyer, D., Barbieri, K., & Doerr, H. M. Creating a virtual classroom for interactive education on the Web. http://www.igd.fhg.de/www/www95/papers/62/ctc.virtual.class/ctc.virtual.class.html.

Eastmond, D. V. (1995). *Alone but Together: Adult Distance Study through Computer Conferencing.* Cresskill, NJ: Hampton Press.

Hares, J. S. & Smart, J. D. (1993). *Object Orientation Technology, Techniques, Management and Migration,* Chichester, UK: John Wiley & Sons.

Hayashi, A. M. (1996, July). When will VisualAge come of age? *Datamation,* pp. 58-63.

IBM. (1997). *IBM VisualAge for COBOL: Building Parts for Fun and Profit. Version 2.0.* GC26-9038-00.

IBM. (1997). *IBM VisualAge for COBOL: User's Guide for Windows. Version 2.0.* SC26-9037-00.

IBM. (1997). *IBM VisualAge for COBOL: Visual Builder User's Guide. Version 2.0.* SC26-9053-00.

IBM-ITSO. (1996). *IBM VisualAge for COBOL for OS/2 Object-Oriented Programming.* SG24-46-6-00.

IBM VisualAge for COBOL version 2.1 technical brochure. http://www.software.ibm.com/ad/cobol/tb9_97.html.

Jonassen, D. H. (1988). Designing structured hypertext, and structuring access to hypertext. *Educational Technology*, 28(11), 13-16.

Kearsley, G. (1988). Authoring considerations for hypertext. *Educational Technology*, 28(11), 21-24.

LaMonica, M. (1995, October 16). IBM delivers object-oriented repository. *Infoworld*, 17(42).

LaMonica, M. (1995, August 14). Micro Focus leverages COBOL object COBOL adapts code for Win95 app development. *Infoworld*, 17(23).

Levey, R. (1996). *Reengineering COBOL with Objects*. New York, NY: McGraw-Hill.

Longhurst, J. & Longhurst, A. (1989). *COBOL*. Englewood Cliffs, NJ: Prentice Hall.

McManus, T. F. Delivering instruction on the World Wide Web. http://www.ccwf.cc.utexas.edu/~mcmanus/wbi.html.

Moore, M.G. & Thompson, M.M. (1990). The effects of distance learning: a summary of the literature. *Research Monographs*, No. 2, American Center for the Study of Distance Education, College of Education, Pennsylvania State University, University Park, PA.

Musciano, C. & Kennedy, B. (1997). *HTML The Definitive Guide*. Cambridge, UK: O'Reilly & Associates, Inc.

O'Neil, H. F. Jr. (1981). *Computer-Based Instruction*. New York, NY: Academic Press.

Parson, R. An investigation into instruction available on the World Wide Web. http://www.oise.on.ca/~rparson/definitn.html.

Porter, L. R. (1997). *Creating the Virtual Classroom: Distance Learning with the Internet*. New York, NY: John Wiley & Sons, Inc.

Sayles, J. COBOL and the business programming paradigm. http://www.tiac.net/user/tangaroa/jss.html.

Sherron, G. T. & Boettcher, J. V. (1997). *Distance Learning: The Shift to Interactivity*. CAUSE Professional Paper Series, #17. Boulder, CO: CAUSE.

Shu, N. C. (1989). *Visual Programming*. New York, NY: Van Nostrand Reinhold.

Steinberg, E. R. (1991). *Teaching Computer to Teach.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Tanenbaum, A. S. (1996). *Computer Networks (3rd Ed.).* Upper Saddle River, NJ: Prentice Hall.

The Great COBOL Debate. http://www.cobol.org/welcome2.htm.

Tkach, D., Fang, W., & So, A. (1996). *Visual Modeling Technique.* Menlo Park, CA: Addison-Wesley.

Van den Brande, L. (1993). *Flexible and Distance Learning.* Chichester, UK: John Wiley & Sons.

Verduin, J. R. & Clark, T. A. (1991). *Distance Education: The Foundations of Effective Practice.* San Francisco, California: Jossey-Bass.

White, C. S. & Hubbard, G. (1988). *Computers and Education.* New York, NY: Macmillan Publishing Company.

Wulf, K. (1996). Training via the Internet: where are we. *Training and Development,* 50(5), 50-55.

VITA

Mei Wang

Candidate for the Degree of

Master of Science

Thesis: CREATING A WEB-BASED TUTORIAL FOR IBM VISUALAGE FOR COBOL

Major Field: Computer Science

Biographical:

Personal Data: Born in Changsha, Hunan, China, the daughter of Qingqi Wang and Shaofang Xiao. Married to Liping Jiang.

Education: Received Bachelor of Science degree in Agriculture from Hunan Agricultural University, Changsha, Hunan, China in July 1985; received Master of Science degree in Agriculture from Southwest Agricultural University, Chongqing, China in June 1988. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in May 1998.

Experience: Employed by Shanghai Agricultural College, Shanghai, China as a lecturer from July 1988 to December 1995; employed by the Department of Computer Science, Oklahoma State University, as a graduate teaching assistant from January 1997 to May 1998.