USING MULTIMEDIA AND WWW TO TEACH COBOL AND IBM O-O COBOL EXTENSION

By

MINZHE XU

Bachelor of Science

Nanjing Institute of Meteorology

Nanjing, P. R. China

1985

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1998

USING MULTIMEDIA AND WWW TO TEACH
COBOL AND IBM O-O COBOL EXTENSION

Thesis Approved:

Jacques E. LaFrance
Thesis Adviser

J Chandler

[illegible]

Wayne B. Powell
Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my grateful appreciation to my advisor, Dr. Jacques LaFrance for his advice, encouragement, patience and kindness throughout my graduate study. A special thanks to Dr. John P. Chandler for his invaluable suggestions and comments and his serving as my graduate research assistantship instructor and supporter. I would also like to express my appreciation to Dr. K. M. George for his guidance and suggestions.

I would also like to express my deep thanks to all my family, especially to my parents for their love and support. My thanks also go to all my friends for their encouragement.

I would like to thank all professors and staff in the Computer Science Department for helping me complete my graduate study at Oklahoma State University.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

COBOL is a high-level business-oriented programming language. During the last several decades, COBOL has been widely used in banking, insurance, and other business applications. COBOL is still running on more than seventy percent of the world's top organizations' computers. In addition, there are an estimated seventy billion lines of code and over five million active COBOL programmers [1]. COBOL has had several revisions. The currently common used version is the latest version COBOL 85. In order to keep the COBOL language itself and the large systems (mostly mainframe) which run the COBOL programs efficient and effective in the next century, IBM has presented the new generation COBOL, IBM VisualAge for COBOL for OS/2 and for Windows.

IBM VisualAge for COBOL has some advantages: VisualAge for COBOL supports local and remote access to a variety of databases and other file systems. All the IBM VisualAge COBOL family solutions support the ANSI 85 COBOL functions, so applications can be ported across supported platforms. VisualAge for COBOL supports object-oriented extensions, enabling the use of modern object-oriented methods with COBOL. VisualAge for COBOL provides a complete development environment. The environment includes an editor, debugger, GUI designer (OS/2 only), and performance analyzer, all integrated with WorkFrame [2].

Understanding and making applications using IBM VisualAge for COBOL for OS/2 and for Windows will greatly improve performance of the COBOL related programs and systems. At this point, it is meaningful to make a tutorial for the IBM VisualAge for COBOL for OS/2 and for Windows with multimedia and provide it on World Wide Web (Web or WWW for short). The whole project of producing the above tutorial is divided into four parts. The thesis proposed herein is to implement one of the four parts, which focuses on the tutorial for COBOL and IBM's object-oriented extensions to COBOL.

The main tool used to produce the COBOL and O-O COBOL tutorial software is Toolbook II Instructor. Toolbook II Instructor is an efficient visual interfaced multimedia courseware development system. Tutorial author can add text, sound, motion picture, and other multimedia information in one product called a book.

By using the embedded tools of the Toolbook II Instructor, we can convert the tutorial from Toolbook file to an HTML file. Therefore, eventually we can place the tutorial on the World Wide Web.

CHAPTER II

REVIEW OF LITERATURE

COBOL History and Characteristics

COBOL is a high level programming language. Its development began in 1959 by CODASYL (the Conference on Data System Language), a voluntary organization that included representatives of users and manufacturers of computers [3]. The purpose of CODASYL was to create a common programming language for use in business applications. In 1960, COBOL was newly established by CODASYL. The first standard version COBOL (USA Standard COBOL 1968) was approved in 1968. By revising and updating the 1968 standard, a new standard (American National Standard COBOL 1974) was created in 1974 [4]. After that another new version, COBOL-85, or Third Standard COBOL, was accepted by the American National Standard Institute (ANSI) in 1985. In COBOL-85, a great number of modifications to the previous standard were incorporated [5].

For the past three decades, COBOL has been the worldwide leader as the most commonly used computer language. Some of reasons for that are listed below:

1. COBOL is a relatively simple English-like high-level computer language. In 1959, when the team created COBOL, they deliberately created it to be simple. The

purpose of the team was to make COBOL even more usable by inexperienced programmers. In the 1960s, the earlier version of COBOL met the goal. The later versions have added complexity to meet rising demands, but COBOL has remained a comparatively simple language.

2. The "CO" part of name of COBOL is for Common. The team that created COBOL deliberately tried to make the language common across different platforms. The purpose was to make the language hardware-independent and software-independent.

3. The BOL part of name of COBOL suggests that COBOL is a Business-Oriented Language. It is naturally suited for commercial data processing [6].

4. As a high-level language COBOL has provided simple, yet capable table handling facilities. Many basic business applications can be viewed as processing tables. These include general ledger, production planning, payroll, and human resources. This makes COBOL better used in business applications than other programming languages [15].

Because standard COBOL has been revised and restandardized periodically to help meet changing technology and expectations, COBOL retains its popularity and leadership in business applications. Although Object-Oriented COBOL (O-O COBOL) is not approved by ANSI as a new standard yet, it has been used in practical programming.

Object-Oriented COBOL

With the computer hardware technology having advanced dramatically in recent years, the computer software has become more and more complicated and software size has become bigger and bigger. When software becomes too complicated and too big, traditional programming methodologies have encountered one problem which is hard to overcome: the difficulty of grasping the totality of solution. Object-oriented programming methodology can be used to solve this problem [26]. This is one reason that the object-oriented method is widely accepted.

There are three OOP (Object-Oriented Programming) principles that make the object-oriented method different from others. The first, encapsulation, is a mechanism that combines code (called method in OOP) and data together into an abstract data type called class. From an outside view, the class is a black box. Any manipulation of data of the class will be prohibited unless it triggers a predefined class method. How the class method operates on the class data is transparent to the outside caller. Therefore, the data will be kept safe inside the class [23]. The second, inheritance, is a process whereby one object can acquire the properties of another object. The concept of inheritance supports the hierarchical classification of objects and avoids the duplication of code. Thus the program can be much simpler. The third, polymorphism, is a property that allows one interface to be used for a general class of actions. Which action is to be used is determined by the type of the object to which it is attached. Since a group of objects all use a similar interface for a similar action, the action calling code becomes much simpler. Thus, a complicated procedure can be simplified [22].

COBOL faces challenges to update its programming methodology. Several influences and demands on COBOL contributed to the generation of Object-Oriented COBOL (briefly written as O-O COBOL). The major influences have been hardware improvement, cost inversion, software engineering, management focus shift, and software tools. Because COBOL is a popular language, there have been a lot of demands for change to the language to make it more accommodating. The remainder of this section reviews most demands that helped to make O-O COBOL.

1. Real-time Applications

An important area of computer use has been real-time applications, as in process control and operations control. In a real-time application, the computer processes multiple flows of data in time to use the output to select or specify how an event is to take place or how a process is to operate. In the implementation of its main target application, COBOL originally had a delayed-time, batch-oriented property. It had no specific provision for handling of prioritized interrupts. Over years, with the improving processing speed of computers, COBOL became feasible for real-time applications. Some medium-sized true real-time applications have been implemented with COBOL, although COBOL rarely has been used for large real-time applications. O-O COBOL, however, provides the facilities for prioritizing and handling interrupts in real-time.

2. Online Transaction Processing (OLTP)

OLTP is an important kind of business application, because transactions (such as buying and selling) are vital parts of business. People often expect such systems to be implemented with COBOL. Because hardware speeds are much faster than they used to be, this is now technically possible. The improved facilities of O-O COBOL enable more extensive use of COBOL for OLTP.

3. Re-entrant Code

In some applications, it is convenient and efficient to repeat some part of the implementing software. For example: Suppose with OTLP, five people might want to get a certain product report at the same time. Processing such transactions, all at the same time, would make a heavy demand on both the hardware and software of the system. COBOL lacks the necessary resources needed to support re-entrance. So, COBOL was restricted from some applications, such as dynamic production scheduling. For O-O COBOL, that restriction is now eased. Re-entrance is supported by O-O COBOL.

4. Interface Enrichment

Enriching the system interface has been a major trend in recent decades. In applications where people interact frequently with the system, providing a comfortable interface now may amount to as much as 75 percent of the source code [7]. COBOL lacked verbs for producing and manipulating multimedia, such as components of pie charts, visual images, and sounds. O-O COBOL greatly strengthens COBOL's ability to meet these interface demands.

5. Graphical User Interface (GUI)

Another demand on COBOL is the development of a graphical user interface (GUI) and the substitute of VDU (Visual Display Unit) displays for printed reports. The GUI typically uses a color VDU or monitor as part of the hardware for the human interface with an application system. The early version of COBOL did not incorporate verbs or sections for GUI creation and modification. O-O COBOL includes a SCREEN section and provides the opportunity for creating a comprehensive GUI interface.

6. System Rigidities

Since most people in organizations work in and with systems implemented with computer software, they may often see the rigidities in the existing system and find it hard to fulfill what they want to do. That causes the demand to change the system quickly to fit the user's current need. Because O-O COBOL is compatible with the previous versions of COBOL and has added new efficient features, O-O COBOL makes software more maintainable and less rigid.

7. Data Sharing

The demands data sharing within an organization and between organizations have greatly increased in the recent years. The scope for sharing may be a local-area network (LAN), a wide-area network (WAN), the Internet, an ordinary dial-up telephone line, a microwave link, and so on. COBOL has had strength in providing controlled data sharing. But it also has had weakness, such as in communication methods. As a new version, O-O COBOL, includes some general strengthening of data sharing, especially in file sharing and the recognition and handling of exception conditions [7].

IBM VisualAge for COBOL

IBM VisualAge for COBOL is a new release by IBM. It supports O-O COBOL with the IBM extensions. With the object-oriented extensions, some coding rules have been added to ensure object-oriented applications can be implemented. IBM VisualAge for COBOL is one member of the IBM VisualAge family. IBM VisualAge family include VisualAge for RPG, VisualAge Generator, VisualAge for Smalltalk, VisualAge for C++, VisualAge for Java, and VisualAge for Basic. IBM VisualAge for COBOL supports the ANSI COBOL-85 functions and is adapted to different platforms. VisualAge for COBOL supports local and remote access to a variety of databases and other file systems. VisualAge for COBOL provides a complete object-oriented development environment which includes an editor, debugger, GUI (graphical user interface) designer, and performance analyzer.

VisualAge for COBOL also provides an object-oriented visual programming tool: Visual Builder. By adopting Visual Modeling Technique, Visual Builder provides a comprehensive and consistent development environment; all components at every stage of development are objects [13]. With visual builder, the developer can create objects on a GUI (graphical user interface) and construct logic and active relations between objects by dragging and dropping. Each object can be coded to trigger a desired action . The built-in tool Code Assistant can generate event code for CUI parts. This automation helps the coding be simpler and clearer [27].

The editor of VisualAge for COBOL can display codes with different colors, and a ruler on top of the editor interface. It also provides a Data Assistant to access database and a Transaction Assistant to support the OLTP application development [27].

Using the VisualAge for COBOL debugger, a programmer can control the flow of the execution of the program by setting break points, facilitating the identification of coding errors. The debugger can display the contents of variables, registers and the call stack [28].

VisualAge for COBOL also provides a performance analyzer and remote edit/compile/debug functions which make the VisualAge for COBOL implementation environment more convenient and efficient.

The above advantages of VisualAge for COBOL have greatly improved the capabilities of the COBOL family language and have satisfied most of the demands of COBOL program developers.

The Purpose of This Thesis

The purpose of this thesis is to design and implement a tutorial for COBOL and IBM's object-oriented extensions to COBOL and post the tutorial on the World Wide Web (WWW or Web for short). The tutorial is one part of the project of creating a curriculum for IBM VisualAge for COBOL. The whole curriculum includes four parts: Part I covers COBOL and IBM's object-oriented extensions to COBOL; Part II is about the Visual Modeling Technique; Part III is about Visual Builder; and Part IV is about IBM VisualAge for COBOL. The goal for the curriculum is to help the user master object-oriented programming and use well the VisualAge for COBOL as a powerful tool.

The tutorial covered in this thesis consists of two sections: COBOL and O-O COBOL. The COBOL section will illustrate COBOL language essential concepts, program structure, coding rules, file systems and so on. O-O COBOL section will discuss object-oriented program concepts, abstract data type definitions, and O-O COBOL coding rules.

The tutorial will be developed with ToolBook II Instructor, a multimedia courseware development tool and converted to HTML files. Thus the tutorial can be posted on the Web. The reason we created a Web-based tutorial is that the number of Internet users has increased exponentially and the Internet has become one of the most important information resources. People who are connected to the Internet can access countless resource sites through the Web by simply using hyperlinks. The Internet is fast becoming an integral part of our personal and professional lives [30]. The Internet has also gradually become a very important tool for education [20].

The Internet, The Web and Distance Education

The Internet is a global collection of interconnected computer networks [29]. The Internet was originally a military project used only for defense purposes. It began in 1969 with the ARPANET, which was an experiment in networking to build a computer network that could withstand an enemy nuclear attack. ARPANET linked together the Department of Defense with military research contractors, including many universities engaged in military research. The ARPANET was very successful in creating reliable networking due to dynamic rerouting [8]. This revolutionary concept involved routing traffic from a disrupted link through the surviving portions of the network system. Due to the success ARPANET many universities wanted to connect to the ARPANET. After 1983 the TCP/IP became the only official protocol and the ARPANET greatly increased [12]. TCP/IP protocol ensures computers that running on different systems (UNIX, Macintosh, Windows 95/NT, and other systems) share the same connection to the Internet. In the1980s, the National Science Foundation created NSFNET and interconnected to ARPANET. Today's Internet emerged based on those networks. In early 1990, the Internet's rate of growth increased dramatically. Since the mid 1990s, the Internet has been growing exponentially [14]. Now over two hundred nations have IP connections to the Internet [17]. One reason that the Internet has increased so fast is that the World Wide Web is widely used on the Internet.

The World Wide Web (Web for short) is a vast collection of documents stored on Internet computers [30]. People who access to the Web can view the Web documents, but not everyone with access to the Internet can access to the Web. The flexibility and popularity of the Web is due to the new protocol HTTP (Hypertext Transfer Protocol) [9].

The Web documents were written in a computer language called Hypertext Markup Language (HTML) [25]. HTML is the language used to prepare Web hypertext documents and create hypertext links. There are some advantages to the Web. The Web is nonlinear: with hypertext link, people can access a series Web documents by just selecting the link. The Web is graphical: the hypertext file can contain any kind of multimedia. The Web is interactive: one can enter text, fill in forms, select options, play sounds, run programs, and even paint pictures. All these advantages attract people to access the Web for various purposes. In the last couple of years, with the rapid increase in the Internet bandwidth and the increasing cost effectiveness of Web service, the number of Web users has increased significantly. For example, Schwab's online accounts increased fifty-eight percent in six months (1.2 million at the end of 1997 to 1.9 million as of July 31, 1998) [16].

The Internet is also widely used in education. It has become a revolutionary mechanism for distance education [24]. Historically, distance education has been proved as an important and effective supplement to traditional "onsite" education. In late 1950, the emergence of closed-circuit TV was envisioned as a powerful solution for distance education especially for meeting education and professional training for people in remote areas [21].

Closed-circuit TV was great progress in creating a mechanism for distance education, but it was still a one-way "talking head" presentation. With the Web, the way of communication among instructors and students has been greatly improved. There are no time and distance restrictions any more. Students can read the course at any time and any place as long as the computer is connected to the Internet [18]. With the Web, students have many topics to choose from at different education sites. Students can also easily look up references, talk on the Internet with e-mail, discuss questions, and so on. Now hundreds of universities provide Web online courses that cover almost every major and there are various education plans for Web students [19]. With the Web, distance education has become much more convenient and efficient. Now more and more people are using the online Web to meet educational objectives [18]. It is believed that distance education with the Internet will keep growing.

CHAPTER III

METHODS AND IMPLEMENTION

This Chapter will present the tools, methods, implementation procedures, and contents of the tutorial for COBOL and Object-Oriented COBOL. There are two basic requirements for the IBM VisualAge for COBOL tutorial. First, since the tutorial is Web based, it should to be able to be converted into HTML version. Second, the tutorial consists of four parts and each part should be consistent with the other formats. In order to meet this goal, it is important to choose a proper and effective development tool.

The Multimedia Software Used to Create the Tutorial

(A) Introduction to ToolBook II Instructor

The software chosen for the development of the tutorial is ToolBook II Instructor (version 5.01), which is developed by Asymetrix Corporation. Asymetrix ToolBook II Instructor gives the courseware author the tools needed to construct and deliver powerful multimedia-based learning environments. The course created with ToolBook II Instructor can be delivered on demand to any student with access to the World Wide

Web, or it can be presented on a local-area network (LAN) for easy local access [10].

ToolBook II Instructor is very effective in creating, distributing, and managing a powerful multimedia courseware. It provides two powerful options for setting up distributed learning systems.

- For distributing courseware on LAN, one can use ToolBook II Instructor Course Management System (CMS), which is part of the ToolBook II Instructor package.
- For distributing courseware on the Internet, one can use ToolBook II Library Course Management System. One can also define a course without using the Library.

(B) The Advantage of ToolBook II Instructor in Creating Courseware

ToolBook is a courseware development system for Windows applications. A ToolBook II application has all the features of Windows applications – graphical user interface, event driven programming, and the ability to interact with other windows.

ToolBook II is an interactive environment for creating (author level) and running (user level) applications. ToolBook II's drawing tools can be used to create the visual interface of an application—its graphics, buttons, fields, and so on. One can also use ToolBook II's programming language, OpenScript, to define the behavior of elements in the application. Users of an application (product) also use ToolBook II to run it. With ToolBook II the elements on the windows are handled by clicking the mouse, typing the keyboard, and so on. ToolBook II uses the term, "book", as a paradigm for the application [10].

A ToolBook II application consists of one or more DOS files called "book". Like a printed book, a book in ToolBook II is divided into pages. One page represents the application's screen. Pages contain fields, buttons, and graphics. Pages and the items on them are called objects in ToolBook II. Different pages can share the same background.

Allowing many pages to share the same background has several important advantages: first, all object properties defined on the background remain the same on each page, so each page will be consistent with the other pages in format; second, the relative hyperlink relation in the background will remain the same on the pages which can greatly save time in creating hyperlinks on each page and reduce the possibility of a misleading hyperlink.

Toolbook II provides the tools to created hotwords. Hotwords are special area of text that allows a user to create and attach scripts to fields or record fields. One can use hotwords to show a hidden field, go to another page, or perform some other actions. Hotwords make convenient navigation controls because they provide an unobtrusive and flexible method for linking related topics.

Toolbook has a built-in tool to covert the Toolbook II book file into HTML files. Using this tool the tutorial created by Toolbook II can be easily converted to a series of HTML files in which all the hyperlinks the same. This advantage is significant when the book size is fairly large.

The Structure of the Tutorial

The tutorial for COBOL and O-O COBOL is constructed as one ToolBook II book. The book consists of four kinds of different pages: title page, content pages, chapter heading pages and general content pages. Each kind of page is created based on certain type of background. There are four types of backgrounds:

- Title background
- Menu background
- Chapter heading background
- General content background.

All the backgrounds use the green marble template for consistency. In Toolbook II, all the applications are implemented by creating and using objects. In the tutorial, each background has four button objects to link between pages: the first page, the previous page, the next page, and the last page (see Figure 1). In Toolbook, there is a background object called record field used to specify a field location in background level and the field will be filled with contents (text, graph, video or other multimedia files) in the page level. Each background also contains several record fields to specify the location of text or graphics on each page, which based on the same background. In the menu background two extra record fields were added for tile and subtitle and also two extra navigation buttons was add to ensure the subsections can be linked to directly (see Figure 3). The chapter heading background contains four record fields. They are used for title of the chapter, subtitle of the chapter, content list of the chapter, and the section note for that chapter (see Figure 5). The general content background has three record fields used for title of the chapter, sub title of the chapter, and section note for that chapter (see Figure 7).

All pages with the same background share a common object provided by the backgrounds. In addition, each page has added extra fields or other objects to meet the needs for presenting the tutorial context individually. In the title page, one text field was added in the center of the page to illustrate the topics covered in this tutorial (see Figure 2). The menu page was added one text field on the left side of the page to show how to use the navigation buttons to read each part of the tutorial (see Figure 4). No extra text field needed to be added to chapter heading pages. Because every necessary field was specified in the background, what needed to be done was just to fill contents into each field. The content field was used for listing contents discussed in the chapter. The contents were created as hotwords, so users could randomly browse any topic they desired without having to follow the tutorial sequence. The part note field was used for indicating which section this chapter belongs to (either ANSI COBOL or O-O COBOL) (see Figure 6). The tutorial main body was presented in each general content page. On each general content page, one extra text field was added to contain the detail of each topic in the tutorial. Since most texts of each topic were too long for the primary interface to hold, the text field used a vertical scroll bar to read the contents (see Figure 8).

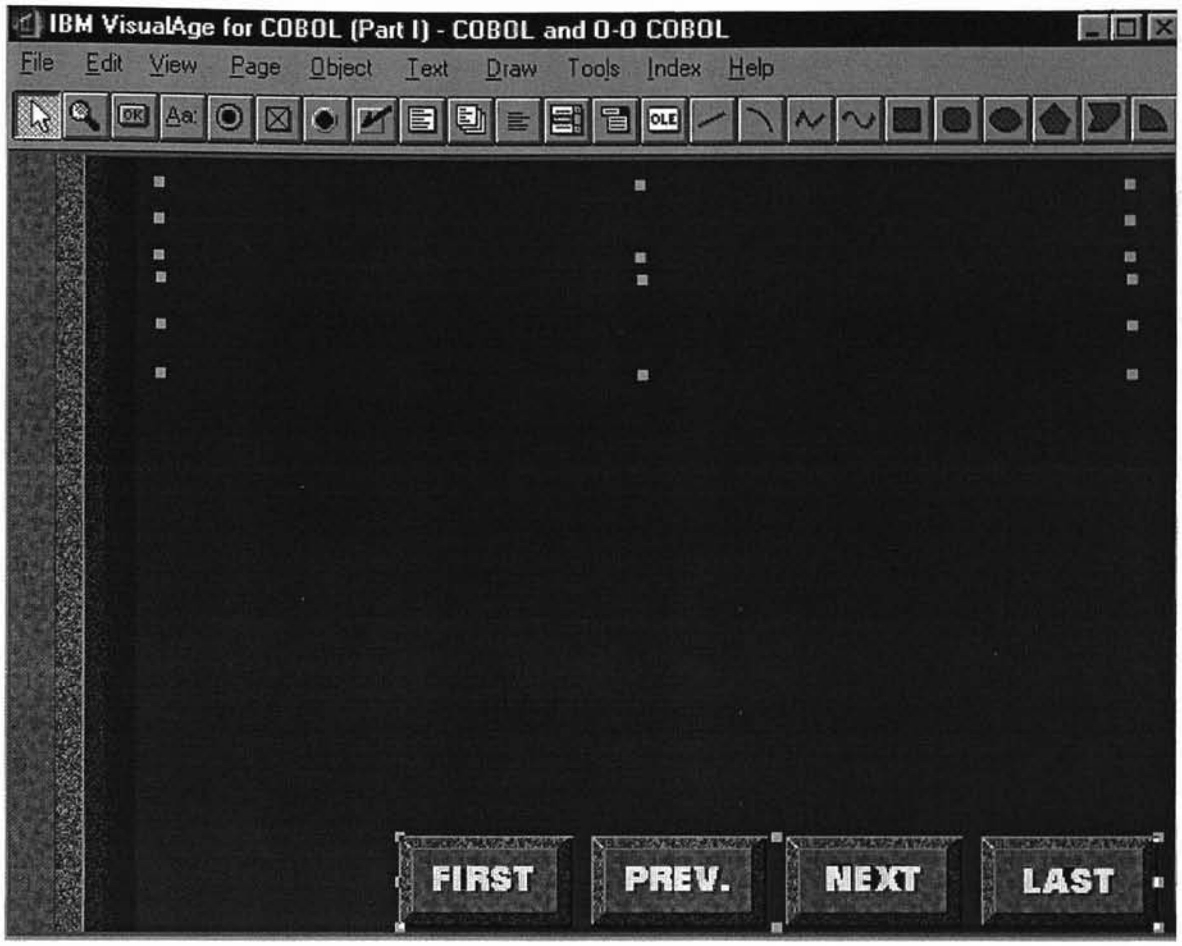
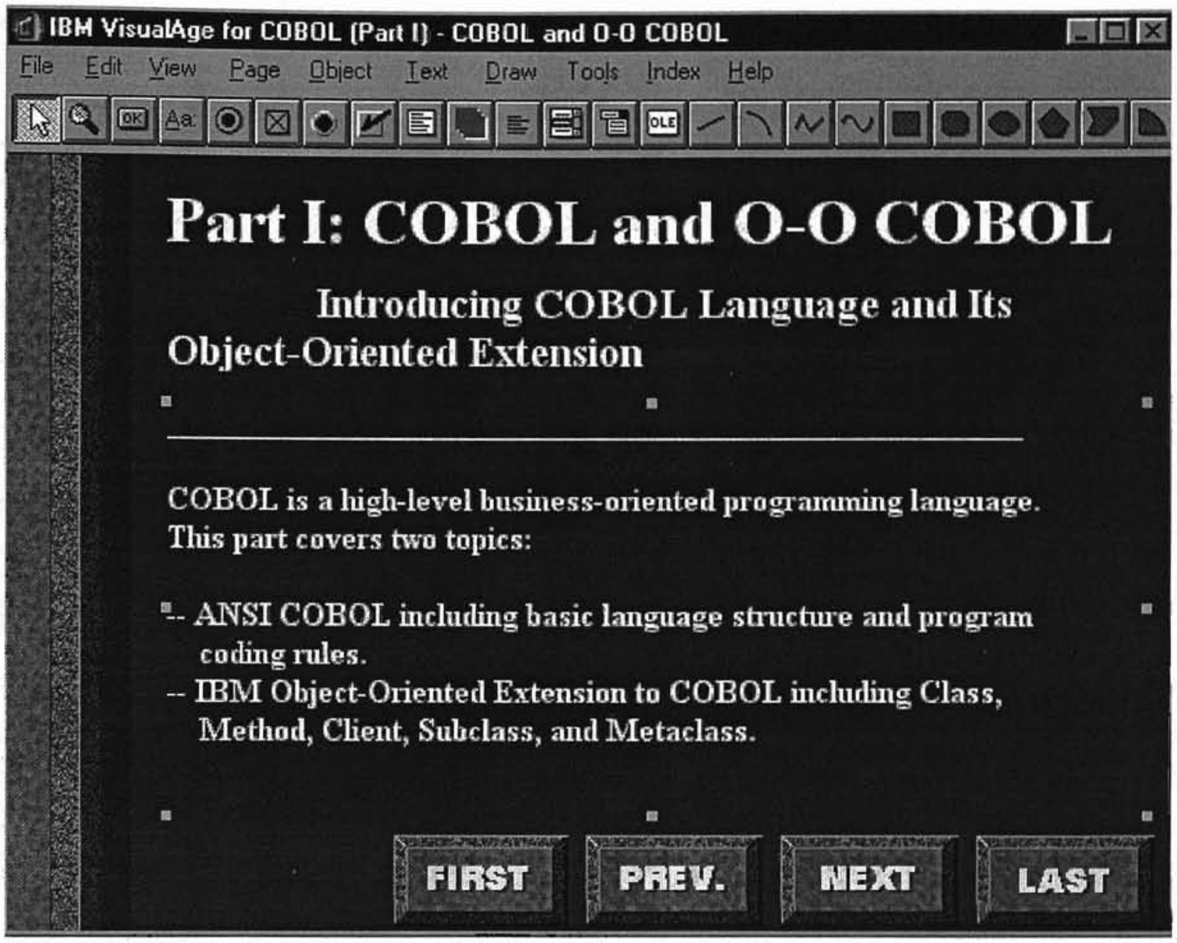

IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL
File Edit View Page Object Text Draw Tools Index Help
FIRST
PREV.
NEXT
LAST

Figure 1. The background of title page


IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL
File Edit View Page Object Text Draw Tools Index Help
FIRST
PREV.
NEXT
LAST

Part I: COBOL and O-O COBOL

Introducing COBOL Language and Its Object-Oriented Extension

COBOL is a high-level business-oriented programming language. This part covers two topics:

-- ANSI COBOL including basic language structure and program coding rules.

-- IBM Object-Oriented Extension to COBOL including Class, Method, Client, Subclass, and Metaclass.

Figure 2. The title page of the tutorial

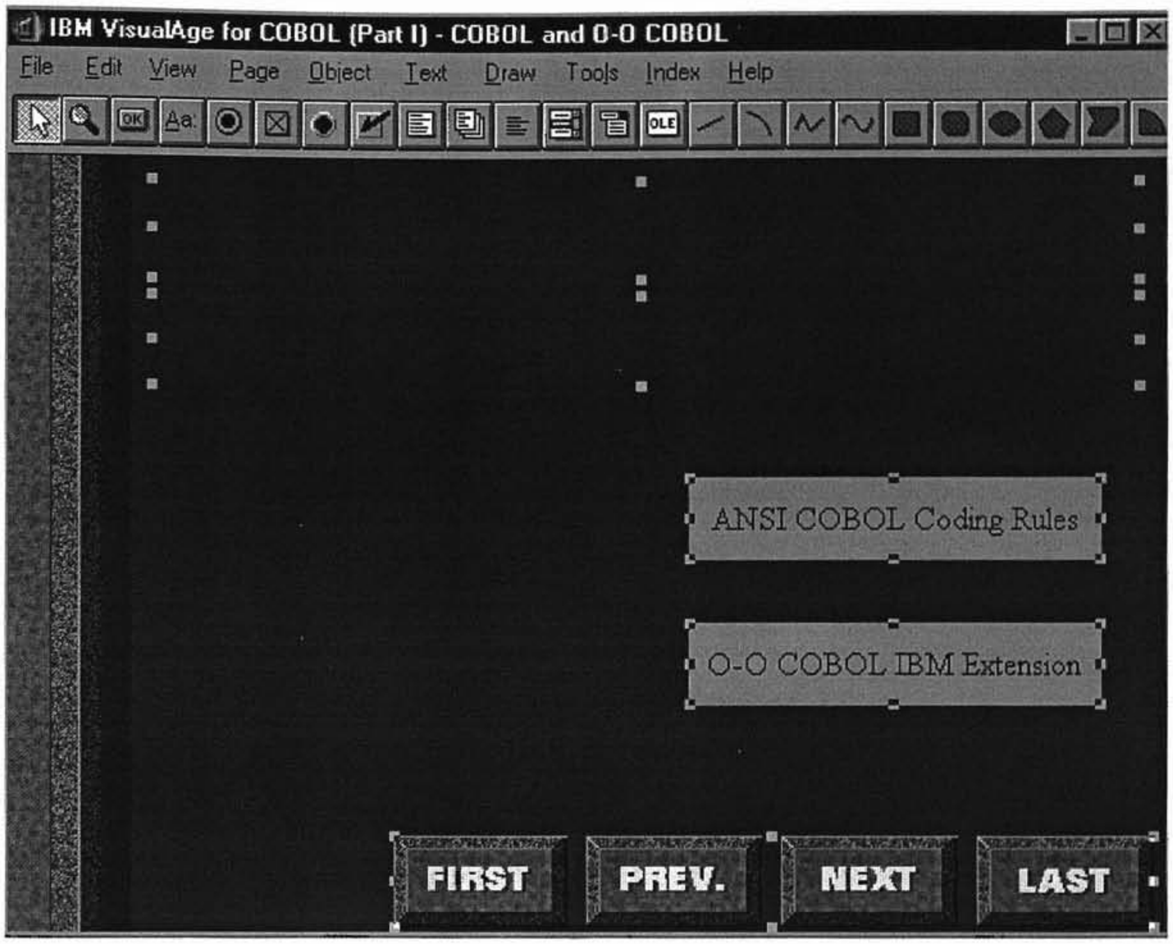
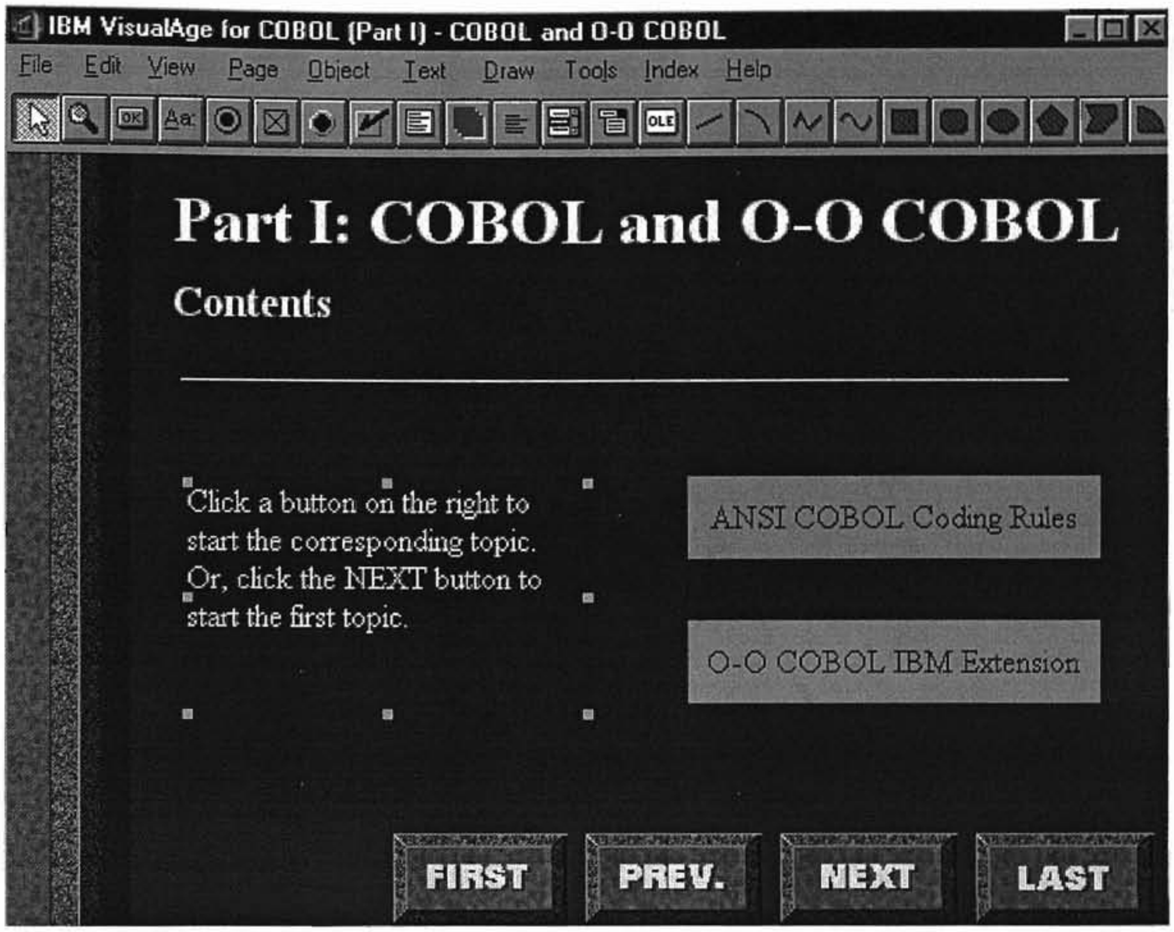

IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL
File Edit View Page Object Text Draw Tools Index Help
ANSI COBOL Coding Rules
O-O COBOL IBM Extension
FIRST
PREV.
NEXT
LAST

Figure 3. The background of menu page



IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL
File Edit View Page Object Text Draw Tools Index Help
FIRST
PREV.
NEXT
LAST

Part I: COBOL and O-O COBOL

Contents

Click a button on the right to start the corresponding topic. Or, click the NEXT button to start the first topic.

ANSI COBOL Coding Rules

O-O COBOL IBM Extension

Figure 4. The menu page of the tutorial

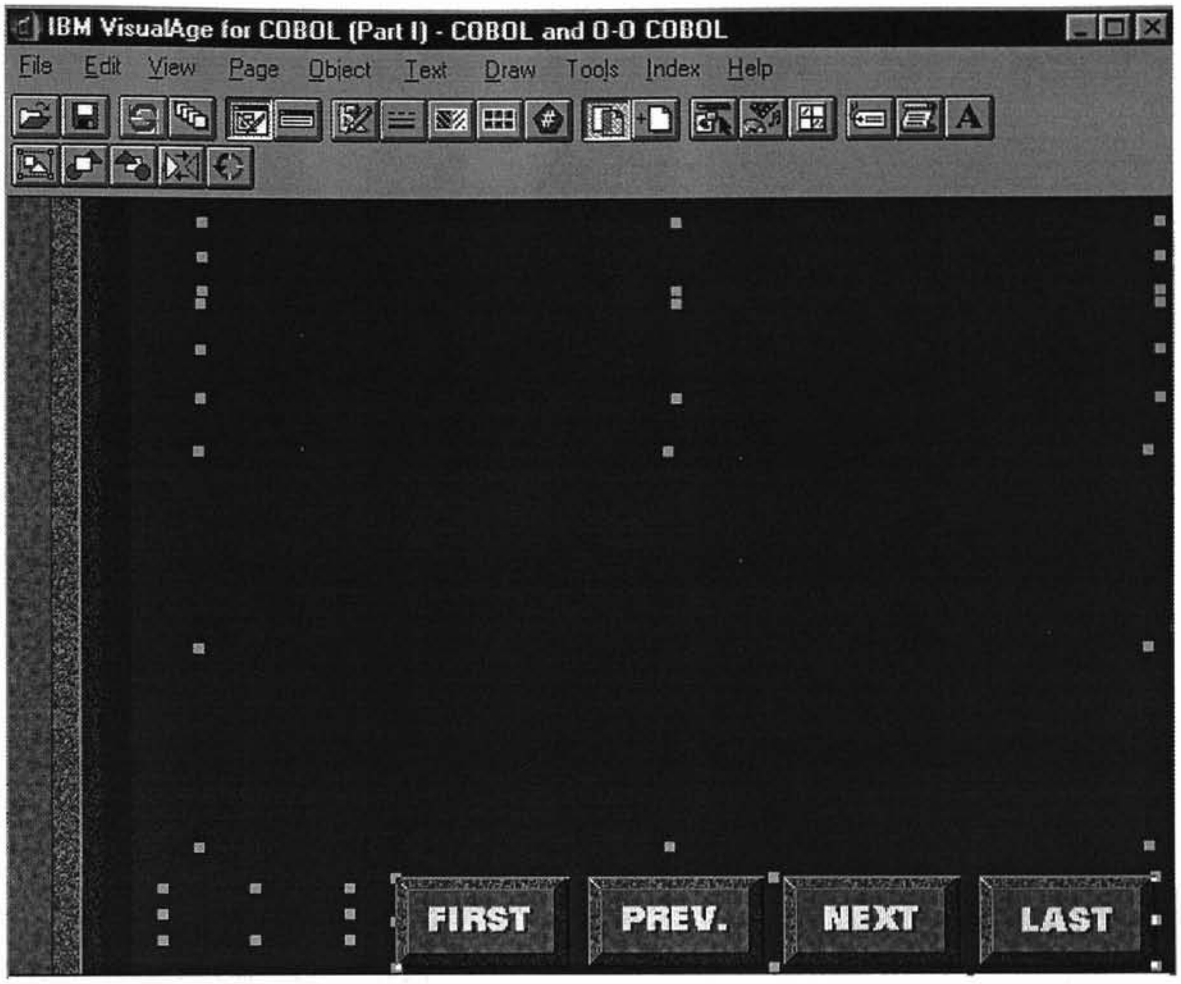
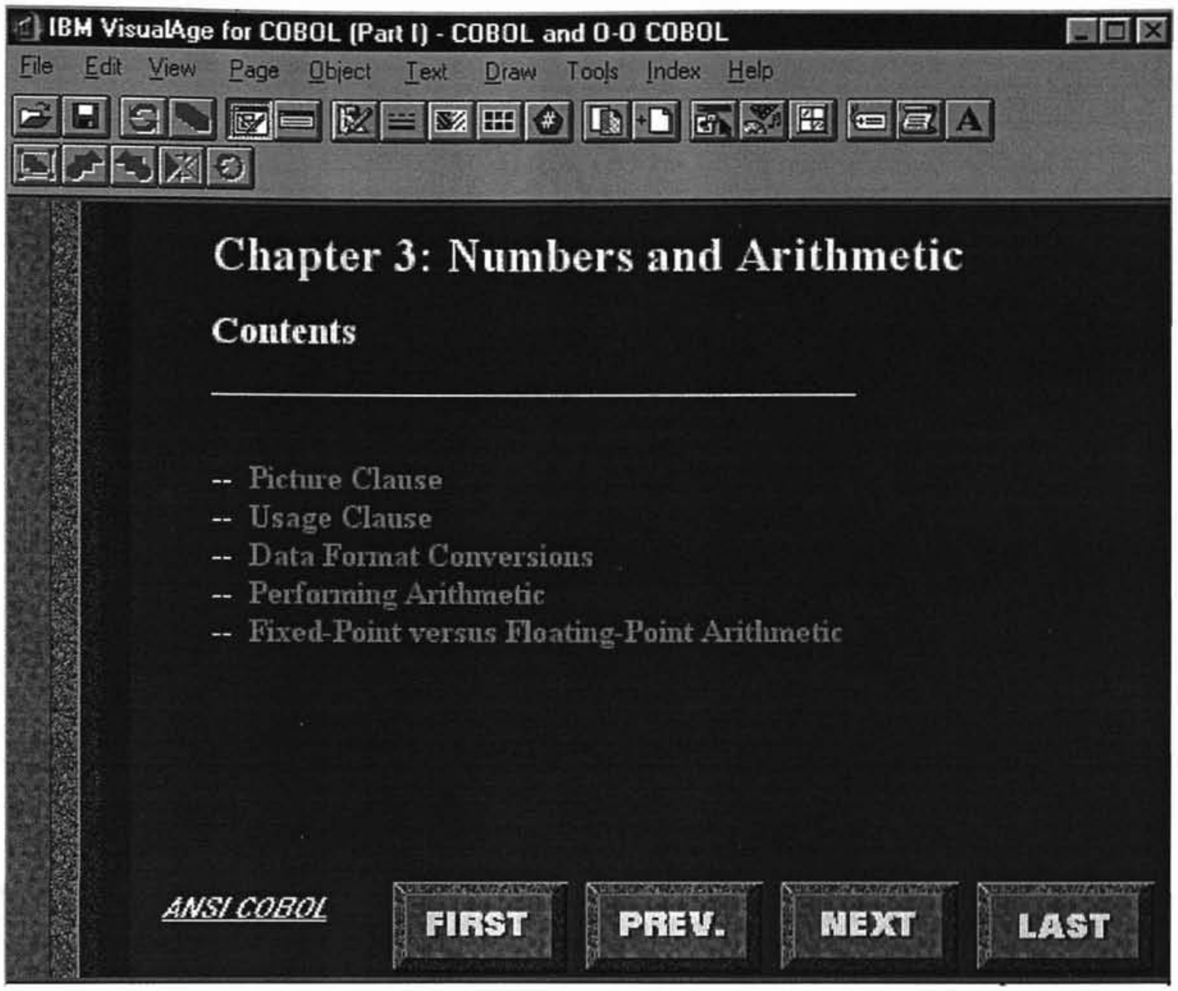

IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL
File Edit View Page Object Text Draw Tools Index Help
FIRST
PREV.
NEXT
LAST

Figure 5. The background of chapter heading page


IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL
File Edit View Page Object Text Draw Tools Index Help
ANSI COBOL
FIRST
PREV.
NEXT
LAST

Chapter 3: Numbers and Arithmetic

Contents

-- Picture Clause
-- Usage Clause
-- Data Format Conversions
-- Performing Arithmetic
-- Fixed-Point versus Floating-Point Arithmetic

Figure 6. The chapter heading page of the tutorial

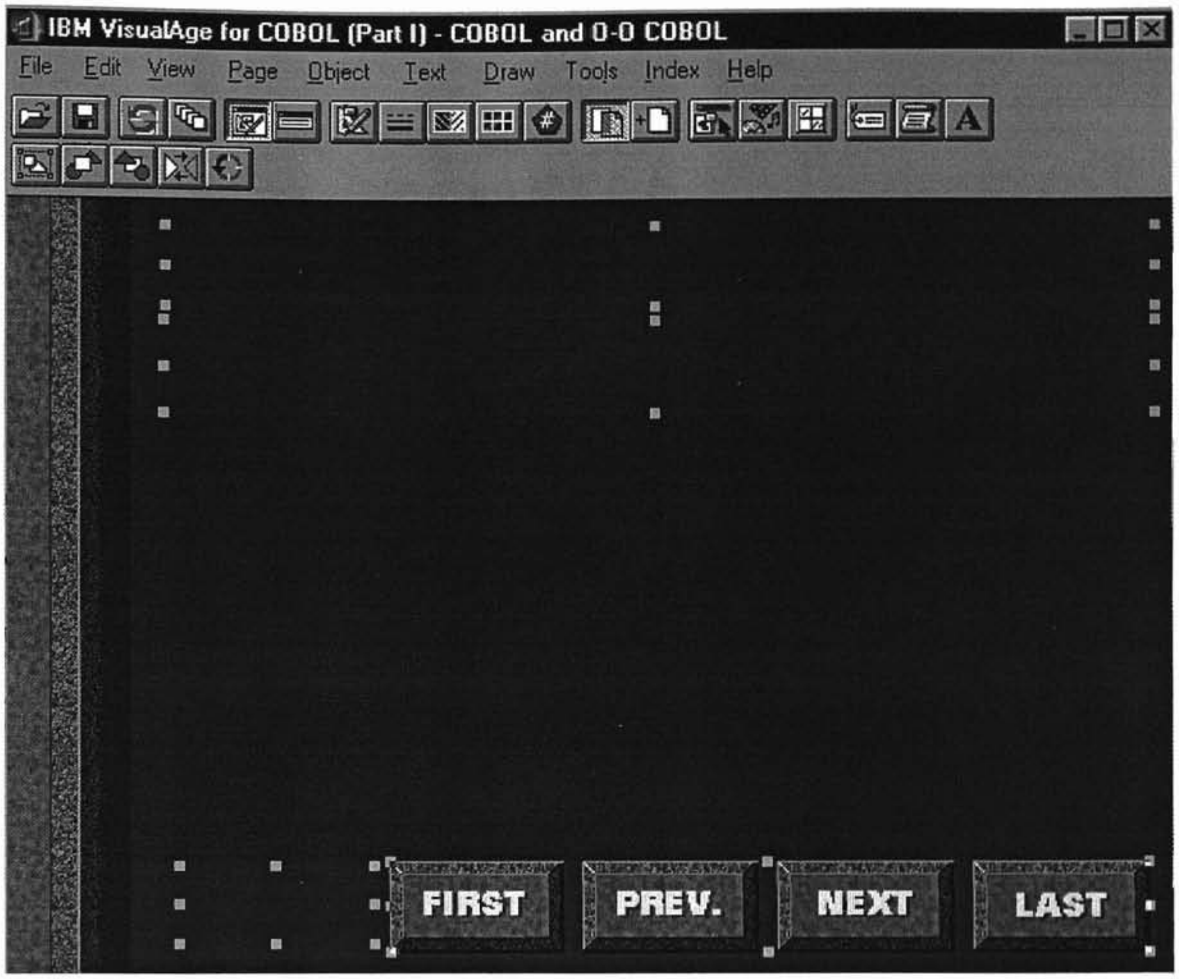
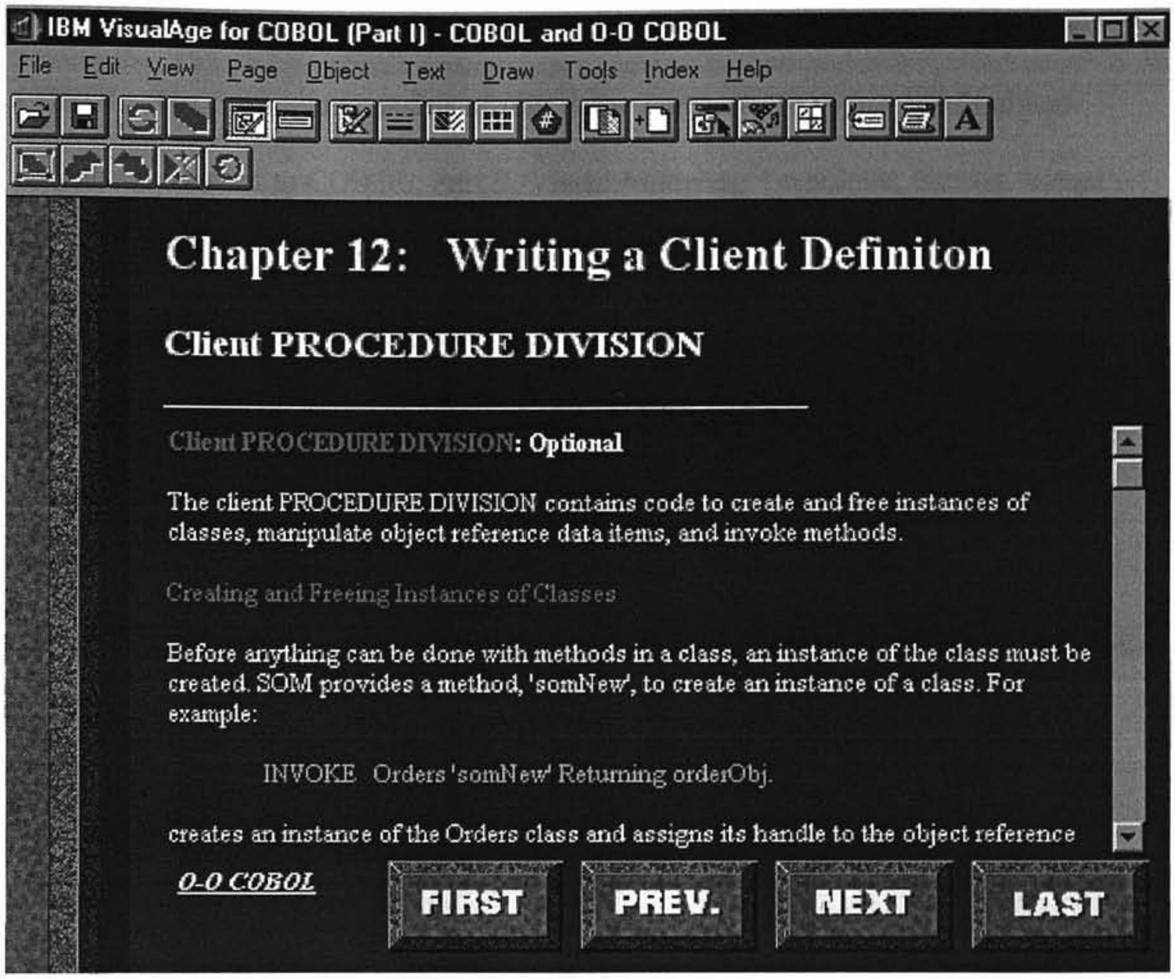

IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL
File Edit View Page Object Text Draw Tools Index Help
FIRST
PREV.
NEXT
LAST

Figure 7. The background of general content page


IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL
File Edit View Page Object Text Draw Tools Index Help

Chapter 12: Writing a Client Definiton

Client PROCEDURE DIVISION

Client PROCEDURE DIVISION: **Optional**

The client PROCEDURE DIVISION contains code to create and free instances of classes, manipulate object reference data items, and invoke methods.

Creating and Freeing Instances of Classes

Before anything can be done with methods in a class, an instance of the class must be created. SOM provides a method, 'somNew', to create an instance of a class. For example:

```
INVOKE  Orders 'somNew' Returning orderObj.
```

creates an instance of the Orders class and assigns its handle to the object reference

Figure 8. The general content page of the tutorial

The Content Arrangement of the Tutorial

This tutorial is the first part of the whole the curriculum for IBM VisualAge for COBOL. The curriculum consists of four parts: Part I, COBOL and IBM's object-oriented extensions to COBOL; Part II, Visual Modeling Technique; Part III, Visual Builder; and Part IV, IBM VisualAge for COBOL. This thesis is about the tutorial of Part I. During the design of this tutorial, the contents were carefully selected to ensure cover all the essential topics COBOL and O-O COBOL language.

The tutorial was composed of two sections: COBOL and O-O COBOL. The COBOL section covered: COBOL essential concepts; program structure; number and arithmetic; handling tables; selection and iteration; string handling; processing files; and error handling. The O-O COBOL section covered: concepts of object-oriented programming; the definitions and coding rules of class, method, client, subclass, metaclass [11]. Each of the above topics was illustrated in one chapter. Each chapter covered different number of sub topics. Many simple program codes and figures were given in the tutorial in order to help user easy to understand the contents. This tutorial was made up of thirteen chapters. The following are the list of contents of this tutorial:

- ANSI COBOL Coding Rules
  - Chapter 1. Definition of COBOL Terms
    - COBOL Syntax
    - Coding Rules
    - Variables, Literals, and Constants
  - Chapter 2. Program Structure

- IDENTIFICATION DIVISION
    - ENVIRONMENT DIVISION
    - DATA DIVISION
    - PROCEDURE DIVISION
- Chapter 3. Numbers and Arithmetic
    - Picture Clause
    - Usage Clause
    - Data Format Conversions
    - Performing Arithmetic
    - Fixed-Point versus Floating-Point Arithmetic
- Chapter 4. Tables Handling
    - Defining a Table (OCCURS Clause)
    - Referring to an Item in a Table
    - Putting Values into a Table
    - Creating Variable-Length Tables (DEPENDING ON Clause)
    - Searching a Table (SEARCH Statement)
    - Processing Table Items (Intrinsic Functions)
- Chapter 5. Selection and Iteration
    - Selection (IF and EVALUATE Statements)
    - Iterative Loop (PERFORM Statement)
- Chapter 6. String Handling
    - Join Data Items (STRING Statement)

    - Splitting Data Items (UNSTRING Statement)
    - Referencing Substrings of Data Items (Reference Modifiers)
    - Tallying and Replacing Data Items (INSPECT Statement)
    - Converting Data Items (Intrinsic Functions)
    - Evaluating Data Items (Intrinsic Functions)
  - Chapter 7. Processing Files
    - General Introduction
    - File Input/output Overview
    - COBOL Coding for Files
    - File Sorting and Merging
  - Chapter 8. Error Handling
    - STRING and UNSTRING Operations
    - Arithmetic Operations
    - Input/Output Error Handling Techniques
    - Call Statements

- O-O COBOL IBM Extension
  - Chapter 9. Writing Object-Oriented Programs
    - O-O Concept
    - Object
    - Method
    - Class

- Instance

- Chapter 10. Writing a Class Definition
    - Class IDENTIFICATION DIVISION
    - Class ENVIRONMENT DIVISION
    - Class DATA DIVISION
    - Class PROCEDURE DIVISION
    - Complete Class Example
- Chapter 11. Writing a Method Definition
    - Method IDENTIFICATION DIVISION
    - Method ENVIRONMENT DIVISION
    - Method DATA DIVISION
    - Method PROCEDURE DIVISION
    - Complete Method Example
- Chapter 12. Writing a Client Definition
    - Client IDENTIFICATION DIVISION
    - Client ENVIRONMENT DIVISION
    - Client DATA DIVISION
    - Client PROCEDURE DIVISION
    - Complete Client Example
- Chapter 13. Writing a Subclass Definition
    - Subclass IDENTIFICATION DIVISION
    - Subclass ENVIRONMENT DIVISION

- Subclass DATA DIVISION
- Subclass PROCEDURE DIVISION

- Chapter 14. Writing a Metaclass Definition
  - Metaclass IDENTIFICATION DIVISION
  - Metaclass ENVIRONMENT DIVISION
  - Metaclass DATA DIVISION
  - Metaclass PROCEDURE DIVISION

Implementation

This tutorial was created with Toolbook II Instructor. In order to keep the interface consistent among the four parts of the whole curriculum and later converted to HTML files, the marble green template was selected from the Internet Content Book Specialist of Toolbook II Instructor.

The tutorial is also called a "book" in Toolbook II Instructor terminology. Toolbook II Instructor contains two different application levels: author and user level. The book was created at author level. Tutorial book was constructed based on the original empty book. The book contained four different page backgrounds: tile background, menu background, chapter heading background, and general content background. These page backgrounds had some common objects on them: four navigating buttons for linking to other pages; some record fields which were used for chapter title or sub title. Some objects were added to the page background as needed. For example, a record field was added at the left bottom area on both chapter heading and general content backgrounds, so later footnote could be added to classify COBOL or O-O COBOL section on page level.

After the backgrounds were set up, the page construction steps began. If necessary some text fields should be added on the page in order to write text into it. All general content pages added text fields in order to write text. Since the size of Toolbook II page interface is fixed, a vertical scroll bar was used in the page text field in order to write more text without restrained by the page interface frame. In the left button area it wrote either ANSI COBOL or O-O COBOL to specify the topic belonged to what section.

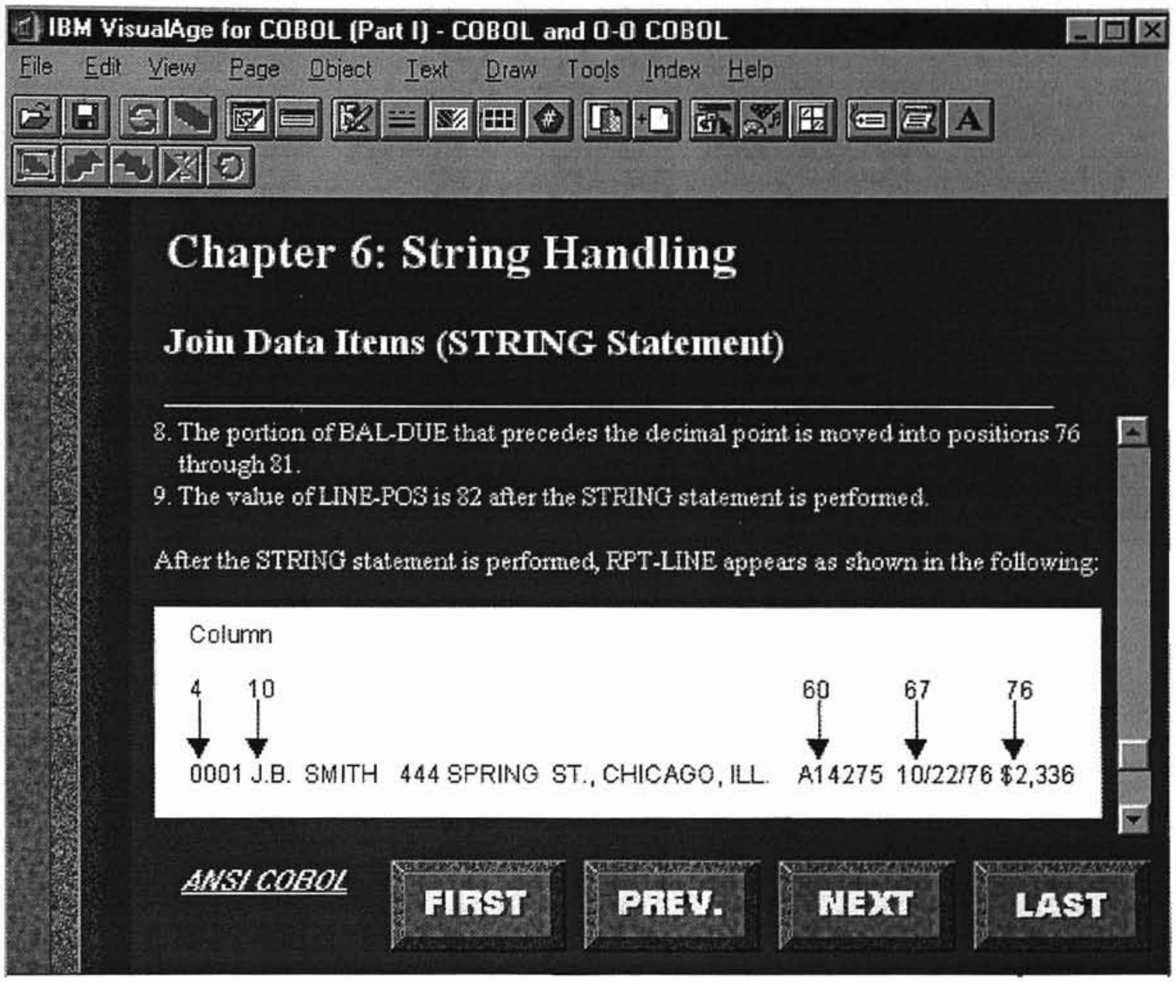
When the contents were written, the topics were highlighted with a certain color and the font was bold. The subtopic also highlighted with the same color, but the font was not bold. In the tutorial, many program code examples were presented (see Figure 8). They were all highlighted with a different color. Many figures were also given in the tutorial to assist the reader in understanding the contents better (see Figure 9).

Hotwords is a special word used in Toolbook II which means a certain highlighted text area that contains hyperlink. With hotwords the reader can reference related topics without having to following the page sequences. Hotwords ensure the reader can actively read any topic he or she needs. In this tutorial, many hotwords were used in the context and also all the topics listed in the chapter heading page were created as hotwords. So the reader can easily select a topic and read details in each page (see Figure 6).

After the book was completed in Toolbook II Instructor's author level, it was tested in Toolbook II's reader level. The performance in reader level exactly met the author' design requirement. The book was then considered completed.

The next step was to convert the book to HTML files with the Toolbook II built-in HTML transfer tool. Then Web browser was used to test the HTML files to see if the HTML files were properly converted. The browsers used to test the HTML files were Netscape Navigator 3.0, 4.0 and Microsoft Internet Explorer 3.0 and 4.0.

During conversion of the book to HTML files, Toolbook II put every object in the page or background into the cells of tables in the HTML files. It caused a problem that some objects in Toolbook were misplaced in HTML. One common problem on every converted HTML files is that all indents in the Toolbook text were disappeared in HTML files (see Figure 10). Netscape Communicator 4.0 was used to add indents and arrange some figures and graphs in HTML files (see Figure 11). But in some cases Netscape Communicator 4.0 could not solve the problem (see Figure 12). To solve this kind problem, The HTML code had to be adjusted. Figure 13 shows the adjusted HTML file Web interface. Appendix A shows parts of the adjusted HTML codes.


IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL
File Edit View Page Object Text Draw Tools Index Help
ANSI COBOL
FIRST
PREV.
NEXT
LAST

Chapter 6: String Handling

Join Data Items (STRING Statement)

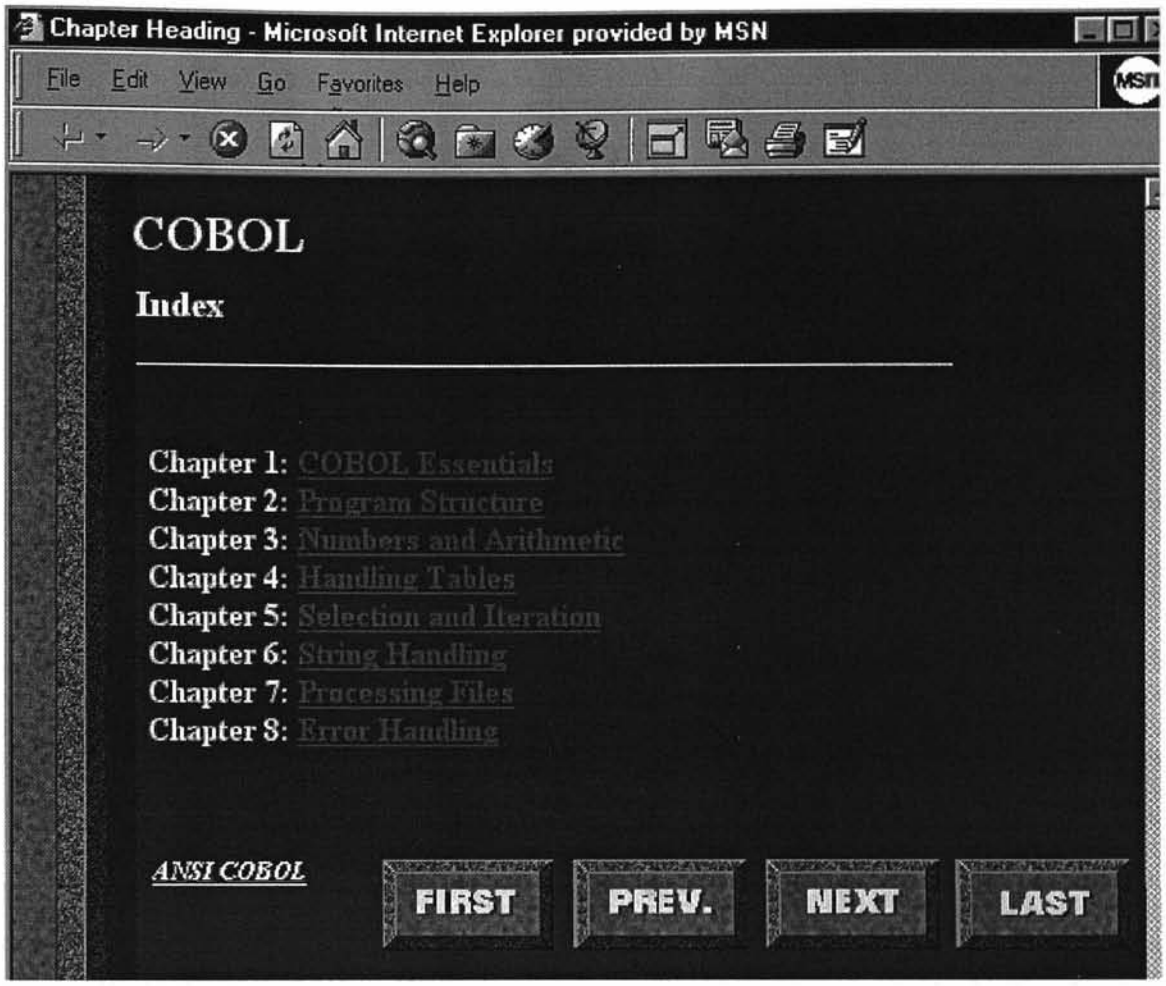
8. The portion of BAL-DUE that precedes the decimal point is moved into positions 76 through 81.
9. The value of LINE-POS is 82 after the STRING statement is performed.

After the STRING statement is performed, RPT-LINE appears as shown in the following:

```
Column

4  10                                                60     67       76
0001 J.B. SMITH  444 SPRING ST., CHICAGO, ILL.   A14275 10/22/76 $2,336
```

Figure 9. A general content page with a figure



Chapter Heading - Microsoft Internet Explorer provided by MSN
File Edit View Go Favorites Help

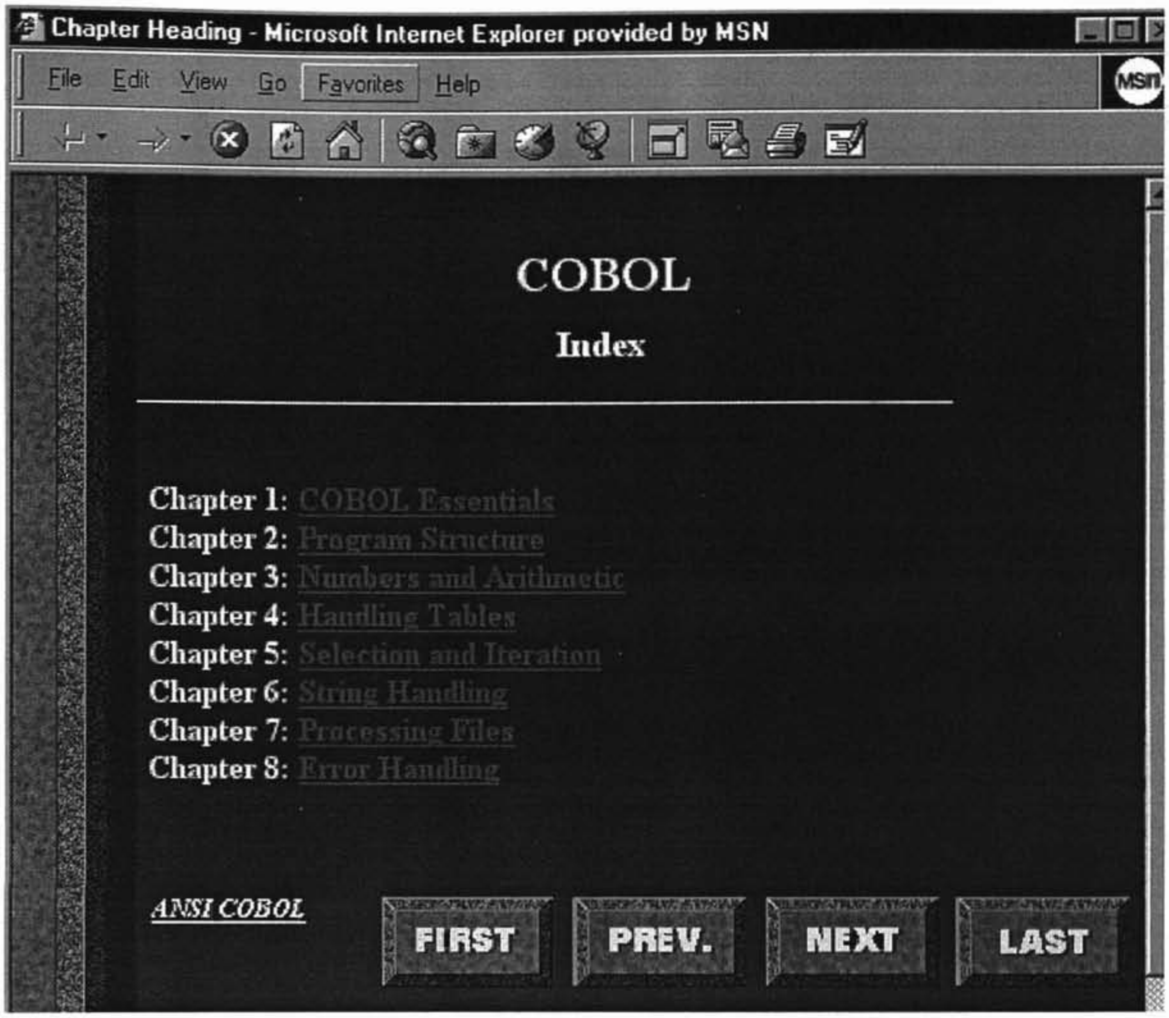
COBOL

Index

Chapter 1: COBOL Essentials
Chapter 2: Program Structure
Chapter 3: Numbers and Arithmetic
Chapter 4: Handling Tables
Chapter 5: Selection and Iteration
Chapter 6: String Handling
Chapter 7: Processing Files
Chapter 8: Error Handling

ANSI COBOL
FIRST
PREV.
NEXT
LAST

Figure 10. An original HTML file with displaced contents

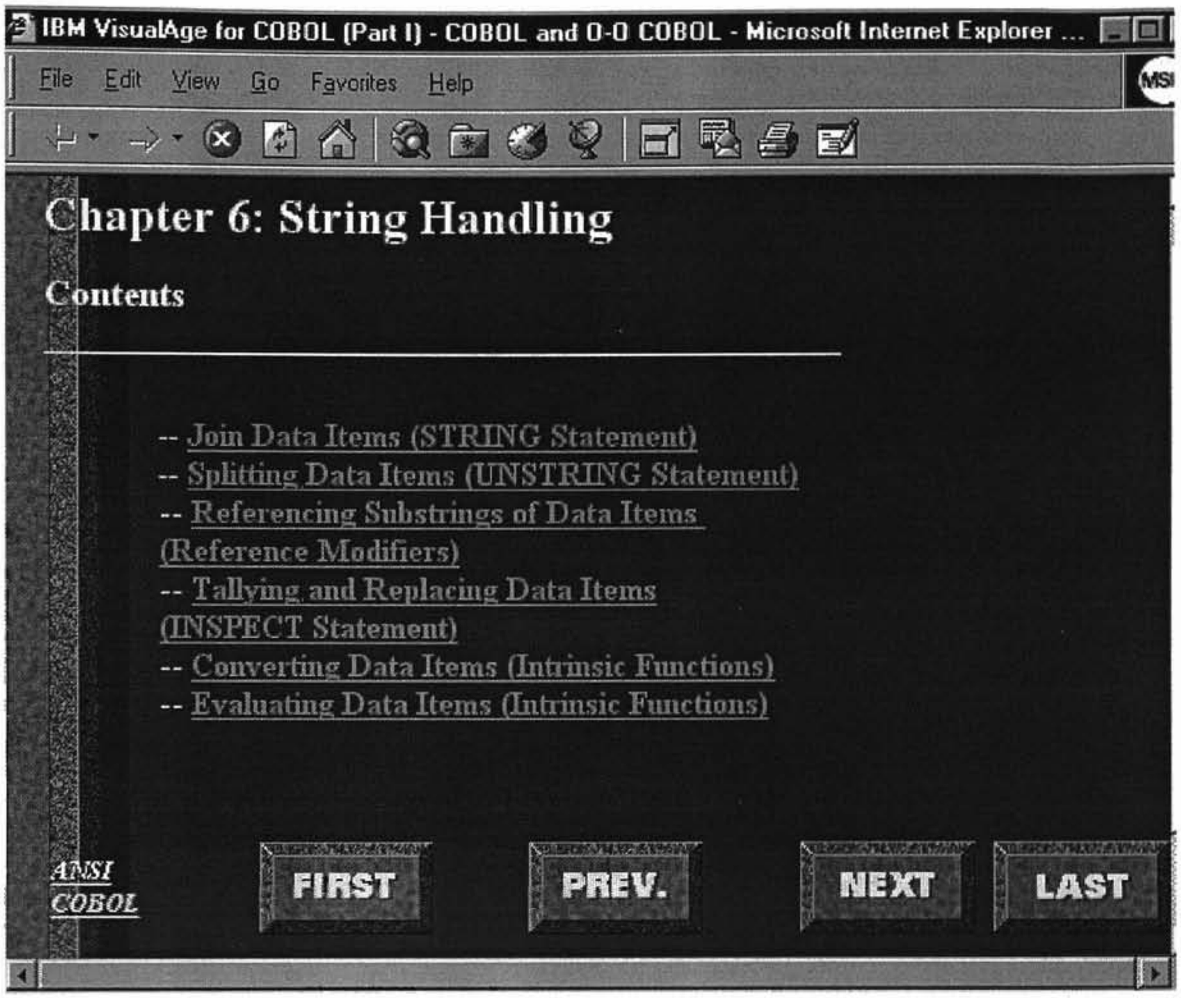

Chapter Heading - Microsoft Internet Explorer provided by MSN
File Edit View Go Favorites Help

COBOL

Index

ANSI COBOL
FIRST
PREV.
NEXT
LAST

Figure 11. A corrected HTML file

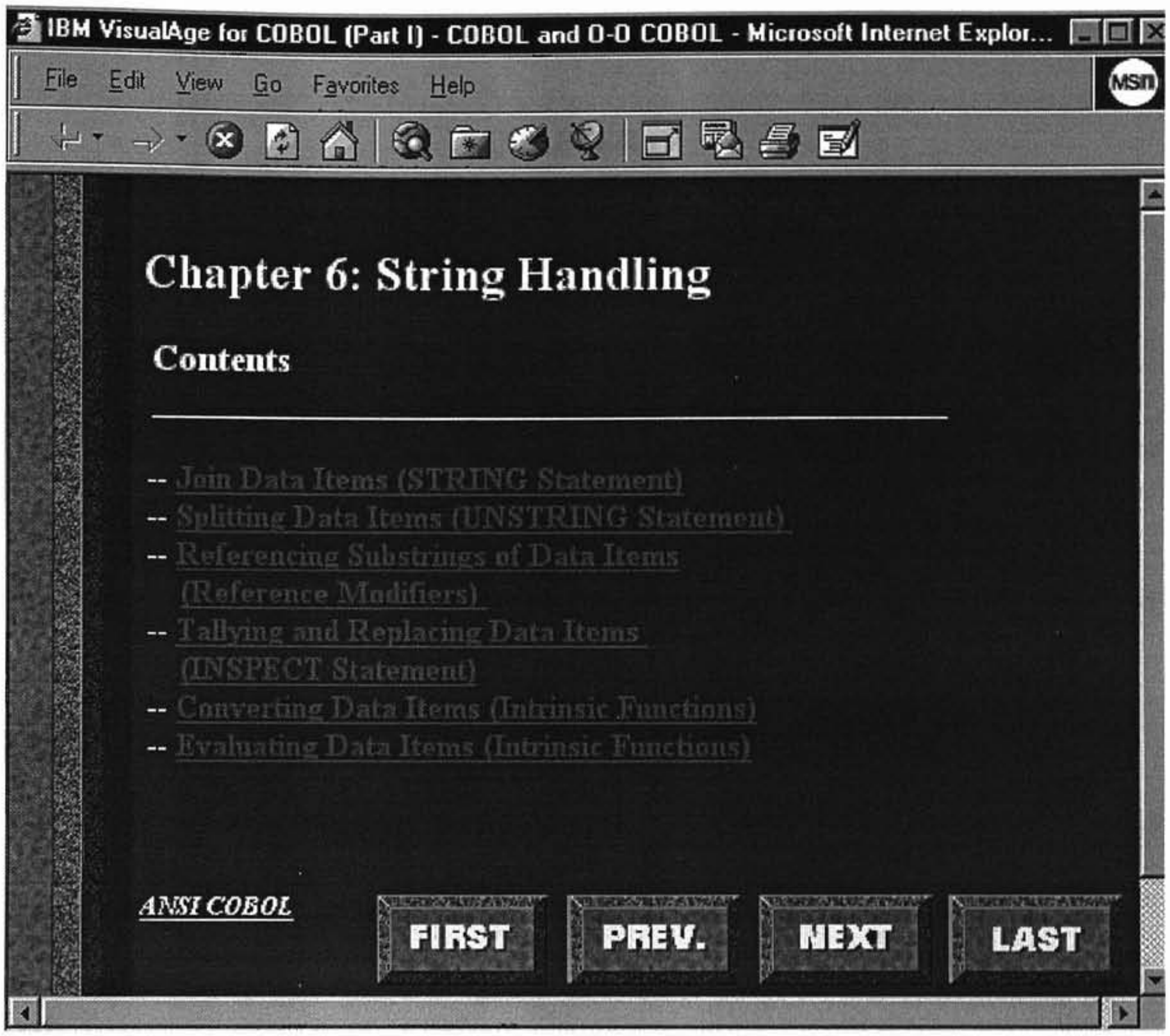

IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL - Microsoft Internet Explorer ...
File Edit View Go Favorites Help
ANSI COBOL
FIRST
PREV.
NEXT
LAST

Chapter 6: String Handling

Contents

-- Join Data Items (STRING Statement)
-- Splitting Data Items (UNSTRING Statement)
-- Referencing Substrings of Data Items (Reference Modifiers)
-- Tallying and Replacing Data Items (INSPECT Statement)
-- Converting Data Items (Intrinsic Functions)
-- Evaluating Data Items (Intrinsic Functions)

Figure 12. A displaced Web page



IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL - Microsoft Internet Explor...

Chapter 6: String Handling

Contents

-- Join Data Items (STRING Statement)
-- Splitting Data Items (UNSTRING Statement)
-- Referencing Substrings of Data Items (Reference Modifiers)
-- Tallying and Replacing Data Items (INSPECT Statement)
-- Converting Data Items (Intrinsic Functions)
-- Evaluating Data Items (Intrinsic Functions)

Figure 13. A Web page corrected by adjusting HTML code

CHAPTER IV

CONCLUSION

The completed tutorial has two versions: one is a ToolBook II book file; the other is an HTML version. The HTML version is made up of ninety-nine inter-linked HTML files. In order to post on the Web, all these HTML files must be put on a Web server and kept under the same directory. The tutorial is temporarily loaded in a Web server in the Computer Science Department Lab in room MS214. The host IP address is 139.78.113.13. The URL (Uniform Resource Locators) for this host will be assigned by the Computer Science Department later. When other tutorials of the whole curriculum are completed, they will be linked together and loaded in the same Web host under the same directory.

As mentioned earlier, when Toolbook II was used to convert the book to HTML files, it caused a problem of objects being displaced. We can, however, use some Web editor tools (like Netscape Communicator 4.0 Editor, Microsoft Internet Explorer 4.0 Editor) to correct the HTML files. What caused the problem of Toolbook II's built-in HTML converter to function improperly is still unknown. The problem has been reported to Asymetrix Corporation. We hope this problem will be solved in the new version of Toolbok II Instructor.

In this thesis, a tutorial for COBOL and IBM's object-oriented extensions to COBOL was developed with Toolbook II Instructor. The tutorial was converted to HTML files and successfully loaded in a Web server. So any users who are connected to the Internet can use a Web browser to learn COBOL and IBM's object-oriented extensions to COBOL from this tutorial.

REFERENCES

1. IBM International Technical Support Centers, *IBM VisualAge for COBOL for OS/2 Object-Oriented Programming*, IBM Corporation, San Jose, CA, 1996, pp. 3-20.

2. IBM Corporation, *IBM VisualAge for COBOL for OS/2 Getting Started*, IBM Corporation, San Jose, CA, 1996.

3. A. Parkin, R. Yorke, R. Barnes, *COBOL for Students*, Third Edition, Edward Arnold, London, UK, 1990, p. 11.

4. Jean Longhurst, Audrey Longhurst, *COBOL*, Prentice Hall, Englewood Cliffs, NJ, 1989, pp. 1-3.

5. R. Ashley, J. Fernandez, *COBOL Wizard: A Wiley Programmer's Reference*, John Wiley & Sons Inc., New York, NY, 1987.

6. L. W. Horn, G. M. Gleason, *Beginning Structured COBOL*, Boyd & Fraser Publishing Company, San Francisco, CA, 1983, pp. 18-20.

7. Ned Chapin, *Standard Object-Oriented COBOL*, Wiley Computer Publishing, Wiley & Sons Inc., New York, NY, 1997, p. 18.

8. P. E. Hoffman, *The Internet Instant Reference, Second Edition*, SYBEX Inc., Alameda, CA, 1995.

9. I. S. Graham, *HTML Source Book, Third Edition*, Wiley Computer Publishing, New York, NY, 1997, p. xv.

10. Asymetrix Corporation, *A Guide to Creating Interactive Courses TOOLBOOK II INSTRUCTOR*, Asymetrix Corporation, Bellevue, WA. 1996, pp. 48-52.

11. IBM Corporation, *VisualAge for COBOL for OS/2 Programming Guide, Version 1.2,* IBM Corporation, San Jose, CA, 1996.

12. A. S. Tanenbaum, *Computer Networks, Third Edition*, Prentice Hall PTR, Upper Saddle River, NJ, 1996.

13. D. Tkach, W. Fang, A. So, (IBM- ISTO*), Visual Modeling Technique - Object Technology Using Visual Programming*, Addison Wesley Longman, Menlo Park, CA, 1996.

14. V. Paxson, J. Mahdavi, A. Adams, M. Mathis, "An Architecture for Large-scale Internet Measurement", *IEEE Communications Magazine*, Vol. 36 (8), August 1998, pp. 88-89.

15. D. Nelson, *COBOL 85 For Programmers*, Elsevier Service Publishing Co., Inc., New York, NY, 1988.

16. J. Schwartz, "Transformation in the Executive Suite", *Internet Week*, September 14, 1998, pp. 20-21.

17. L. Press, G. Burkhart, "An Internet Diffusion Framework", *Communications of the ACM*, Vol. (10), October 1998, pp. 55-57.

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

19. J. P. Duffy, *College Online: How to Take College Course without Leaving Home*, John Wiley & Sons, Inc., New York, NY, 1997.

20. D. Corrigan, *The Internet University: College Courses by Computer*, Cape Software, Harwich, MA, 1996.

21. T. Evans, D. Nation, *Opening Education: Policies and Practices from Opening Distance Education*, Routledge, New York, NY, 1996.

22. S. Shlaer, S. J. Mellor, *Object-Oriented System Analysis: Modeling the world in Data*, Yourdon Press, Englewood Cliffs, NJ, 1988.

23. P. Coad, J. Nicola, *Object-Oriented Programming*, Yourdon Press, Englewood Cliffs, NJ, 1988.

24. I. Miller, J. Schlosbreg, *Distance Learning: Graduate Education that Comes to Your Home*, Kaplan Education Center and Simon & Schuster, New York, NY, 1997.

25. M. R. Brown J. Honeycutt, *Using HTML, Fourth Edition*, Que Corporation, Indianapolis, IN, 1998, pp. 80-99.

26. P. Naughton, H. Schildt, *JAVA: The Complete Reference*, Osborne McGraw-Hill, Berkeley, CA, 1997.

27. IBM International Technical Support Centers, *IBM VisualAge for COBOL: Visual Builder User's Guide*, IBM Corporation, San Jose, CA, 1997.

28. IBM International Technical Support Centers, *IBM VisualAge for COBOL: User's Guide for Windows*, IBM Corporation, San Jose, CA, 1997.

29. D. Gosselin, *Quick Reference Guide: World Wide Web,* DDC Publishing, Inc., New York, NY, 1995

30. M. Walker, *How to Use the Internet, Fourth Edition*, Ziff-Davis Press, an imprint of Macmillan Computer Publishing USA, Emeryville, CA, 1997.

APPENDIX A

A Sample of the HTML codes for an adjusted Web page

```
<HTML>
<HEAD>
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
<META NAME="GENERATOR" CONTENT="Mozilla/4.03 [en] (Win95; I) [Netscape]">
<META NAME="Author" CONTENT="Minzhe Xu">
<TITLE>IBM VisualAge for COBOL (Part I) - COBOL and O-O COBOL</TITLE>
<!-- HTML produced by ASYMETRIX ToolBook II -->
</HEAD>
<BODY TEXT="#000000" BGCOLOR="#29424A" LINK="#FF00FF" VLINK="#663366"
ALINK="#FF00FF" BACKGROUND="b108.gif" TOPMARGIN="0" LEFTMARGIN="0">
&nbsp;
<TABLE BORDER=0 CELLSPACING=0 CELLPADDING=0 WIDTH="564" HEIGHT="384" >
.
.
.

<TD COLSPAN="11" WIDTH="470" HEIGHT="34"><B><FONT FACE="TimesRoman"><FONT
COLOR="#FFFFFF"><FONT SIZE=+2>Chapter
6: String Handling</FONT></FONT></FONT></B></TD>
.
.
.
<TD COLSPAN="10" WIDTH="468" HEIGHT="50"><B><FONT FACE="TimesRoman"><FONT
COLOR="#FFFFFF"><FONT SIZE=+1>Contents</FONT></FONT></FONT></B>&nbsp;
<BR><B><FONT FACE="TimesRoman"><FONT COLOR="#FFFFFF"><FONT
SIZE=+1>____________________________________________</FONT></FONT></FONT></B></TD
>
.
.
.
<TD COLSPAN="12" WIDTH="478" HEIGHT="202"><B><FONT FACE="TimesRoman"><FONT
COLOR="#FFFFFF"><FONT SIZE=+0>--
<A HREF="Cobo_39.htm">Join Data Items (STRING
Statement)</A></FONT></FONT></FONT></B>&nbsp;
<BR><B><FONT FACE="TimesRoman"><FONT COLOR="#FFFFFF"><FONT SIZE=+0>-- <A
HREF="Cobo_40.htm">Splitting
Data Items (UNSTRING Statement)</A></FONT></FONT></FONT></B><A
HREF="Cobo_40.htm">&nbsp;</A>
```

```
<BR><B><FONT FACE="TimesRoman"><FONT COLOR="#FFFFFF"><FONT SIZE=+0>-- <A
HREF="Cobo_41.htm">Referencing
Substrings of Data Items</A>&nbsp;</FONT></FONT></FONT></B>&nbsp;
<BR><B><FONT FACE="TimesRoman"><FONT COLOR="#FFFFFF"><FONT
SIZE=+0>&nbsp;&nbsp;&nbsp;
<A HREF="Cobo_41.htm">(Reference Modifiers)</A></FONT></FONT></FONT></B><A
HREF="Cobo_41.htm">&nbsp;</A>
<BR><B><FONT FACE="TimesRoman"><FONT COLOR="#FFFFFF"><FONT SIZE=+0>-- <A
HREF="Cobo_42.htm">Tallying
and Replacing Data Items</A></FONT></FONT></FONT></B><A HREF="Cobo_42.htm">&nbsp;</A>
<BR><B><FONT FACE="TimesRoman"><FONT COLOR="#FFFFFF"><FONT
SIZE=+0>&nbsp;&nbsp;&nbsp;
<A HREF="Cobo_42.htm">(INSPECT Statement)</A></FONT></FONT></FONT></B>&nbsp;
<BR><B><FONT FACE="TimesRoman"><FONT COLOR="#FFFFFF"><FONT SIZE=+0>-- <A
HREF="Cobo_43.htm">Converting
Data Items (Intrinsic Functions)</A></FONT></FONT></FONT></B>&nbsp;&nbsp;
<BR><B><FONT FACE="TimesRoman"><FONT COLOR="#FFFFFF"><FONT SIZE=+0>-- <A
HREF="Cobo_44.htm">Evaluating
Data Items (Intrinsic Functions)</A></FONT></FONT></FONT></B></TD>
.
.
.
<TD COLSPAN="4" ROWSPAN="2" WIDTH="104" HEIGHT="2"><B><I><U><FONT
FACE="Dialog"><FONT COLOR="#FFFFFF"><FONT SIZE=-1>ANSI
COBOL</FONT></FONT></FONT></U></I></B></TD>
.
.
.
<TD ROWSPAN="2" WIDTH="84" HEIGHT="34"><IMG SRC="b101.gif" ALT="First" BORDER=0
HEIGHT=45 WIDTH=85></A></TD>

<TD WIDTH="10" HEIGHT="34"></TD>

<TD ROWSPAN="2" WIDTH="84" HEIGHT="34"><IMG SRC="b103.gif" ALT="Previous"
BORDER=0 HEIGHT=45 WIDTH=85></A></TD>

<TD WIDTH="10" HEIGHT="34"></TD>

<TD ROWSPAN="2" WIDTH="84" HEIGHT="34"><IMG SRC="b105.gif" ALT="Next" BORDER=0
HEIGHT=45 WIDTH=85></TD>
.
.
.
<TD COLSPAN="4" ROWSPAN="2" WIDTH="84" HEIGHT="34"><IMG SRC="b107.gif" ALT="Last"
BORDER=0 HEIGHT=45 WIDTH=85></TD>
</TABLE>
&nbsp;
<BR>&nbsp;
<BR>&nbsp;
</BODY>
</HTML>
```

VITA

Minzhe Xu

Candidate for the Degree of

Master of Science

Thesis: USING MULTIMEDIA AND WWW TO TEACH COBOL AND IBM O-O COBOL EXTENSION

Major Field: Computer Science

Biographical:

Personal Data: Born in Harbin, Heilongjiang, P. R. China, the son of Longye Xu and Yushan Tian.

Education: Graduated from Harbin No.24 high school, Harbin, Heilongjiang China, in July 1981; received Bachelor of Science degree in Atmospheric Physics from Nanjing Institute of Meteorology, Nanjing, Jiangsu, China in July 1985; Completed requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December, 1998.

Experience: Weather forecast engineer, Heilongjiang Provincial Weather Station, Harbin, Heilongjiang, China from August 1985 to December 1994; graduate research assistant, Computer Science Department, Oklahoma State University from June to December 1997.