A Flexible Integrated Computer System for Engineering Geology Education

by

Matthew Grebner

B.S.C.E., Rensselaer Polytechnic Institute (1985)

Submitted to the Department of Civil Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Civil Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1989

© Massachusetts Institute of Technology 1989

Signature redacted

Signature of Author .....
Department of Civil Engineering
May 19, 1989

Signature redacted

Certified by .....
Herbert H. Einstein
Professor, Department of Civil Engineering
Thesis Supervisor

Signature redacted

Accepted by .....
Ole S. Madsen
Chairman, Departmental Committee on Graduate Students

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUN 01 1989

ARCHIVES LIBRARIES

A Flexible Integrated Computer System for Engineering Geology Education

by

Matthew Grebner

Submitted to the Department of Civil Engineering
on May 19, 1989, in partial fulfillment of the
requirements for the degree of
Master of Science in Civil Engineering

Abstract

The purpose of this study was the development of computer-based tools for engineering geology education. The primary considerations and objectives in the design process were the fulfilling of user needs and the integration of the entire, robust system.

The two modules developed as part of the learning environment were "textbook" and "quiz" modules. These modules were designed to be easy to use for both the student and for the educator who prepares the material. They were developed on a flexible, multimedia platform to allow for future expansion. The textbook module displays text and associated video images, with access controlled by the user. The quiz module asks the student multiple choice questions and checks the answers.

This thesis outlines and explains the processes and motives in the design and evolution of these modules and attempts to indicate the areas in which future development would be particularly useful.

Thesis Supervisor: Herbert H. Einstein
Title: Professor, Department of Civil Engineering

Contents

1 An Introduction to the Engineering Geology Educator **6**

1.1 The Environment 6

1.2 The Engineering Geology Educator 7

1.2.1 Textbook-Learning Module 7

1.2.2 Application Modules 8

1.3 The Platform 8

2 General Design Concepts **11**

2.1 Integration of Modules 11

2.2 Data Interchange and Linking 12

2.3 Data Handler for Complex Operations 13

2.4 Structure 15

3 The Electronic Textbook **16**

3.1 Overview 16

3.1.1 Description 16

3.1.2 Organization 17

3.2 User Interface 18

3.2.1 Layout of Screen 18

3.2.2 Selecting Text 19

3.2.3 Selecting Images 21

3.3 Linking to Other Modules 22
3.4 Construction of the Textbook 23
3.4.1 Preliminary Work 23
3.4.2 Text 23
3.4.3 Images 25
3.5 Updates and Enhancements 25
3.5.1 Modifying the Images and Text 25
3.5.2 Adding Moving Images 25
3.5.3 Other Possible Applications 26
3.6 Summary 26

4 The Workbook **27**
4.1 Overview 27
4.2 User Interface 27
4.3 Constructing a Workbook 29
4.4 Putting it Together 29

5 Other Modules **32**
5.1 Existing Modules 32
5.1.1 Photo Identification 32
5.1.2 NOMAD 32
5.2 New Development 34
5.2.1 Compatibility Requirements 34
5.2.2 Other Modules 34

A Mutex **36**
A.1 The Problem 36
A.2 Mutex Input 38
A.2.1 Text 38
A.2.2 Images 40

A.2.3 Combining and Sectioning Text and Images 43
A.3 Internal Function 44
A.3.1 Input Structure 45
A.3.2 Output Structure 45
A.3.3 Processing 46
A.4 Processing the Input 48
A.5 Possible Expansion 50
A.5.1 Text Processing 50
A.5.2 The Input Stream 51

B Images **53**
B.1 Content of Images 53
B.2 Storage Media 53
B.2.1 Digital Images 54
B.2.2 Video Images 54
B.2.3 The Choice of Video 55

Acknowledgments

Many people have contributed to this project. There are a few who deserve special note.

Gigi Richard provided invaluable advice on the design of the mutex input system. She has been involved in every aspect of the system, except the technical details of writing computer code.

Evelyn Schlusselberg and Matt Hodges of Project Athena have provided special assistance and support.

Siena May and Laura Anderson helped prepare this thesis.

Without Professor Einstein's constant gentle pressure, I would still be trying to figure out the very best way to begin this project.

Chapter 1

An Introduction to the Engineering Geology Educator

The purpose of this study is to develop a complete computerized system for engineering geology education, the Engineering Geology Educator. The system consists of workstation and software integration designed to support education in engineering geology.

1.1 The Environment

This project was developed in a Unix-based environment known as Project Athena. Now in its seventh year, Athena is an Institute-wide experiment for the incorporation of computers into the educational environment. The project offers computing resources to all undergraduates, graduate students and faculty members, and provides the opportunity for innovative use of advanced computer workstations on an extensive, coherent network [Avril, 1989].

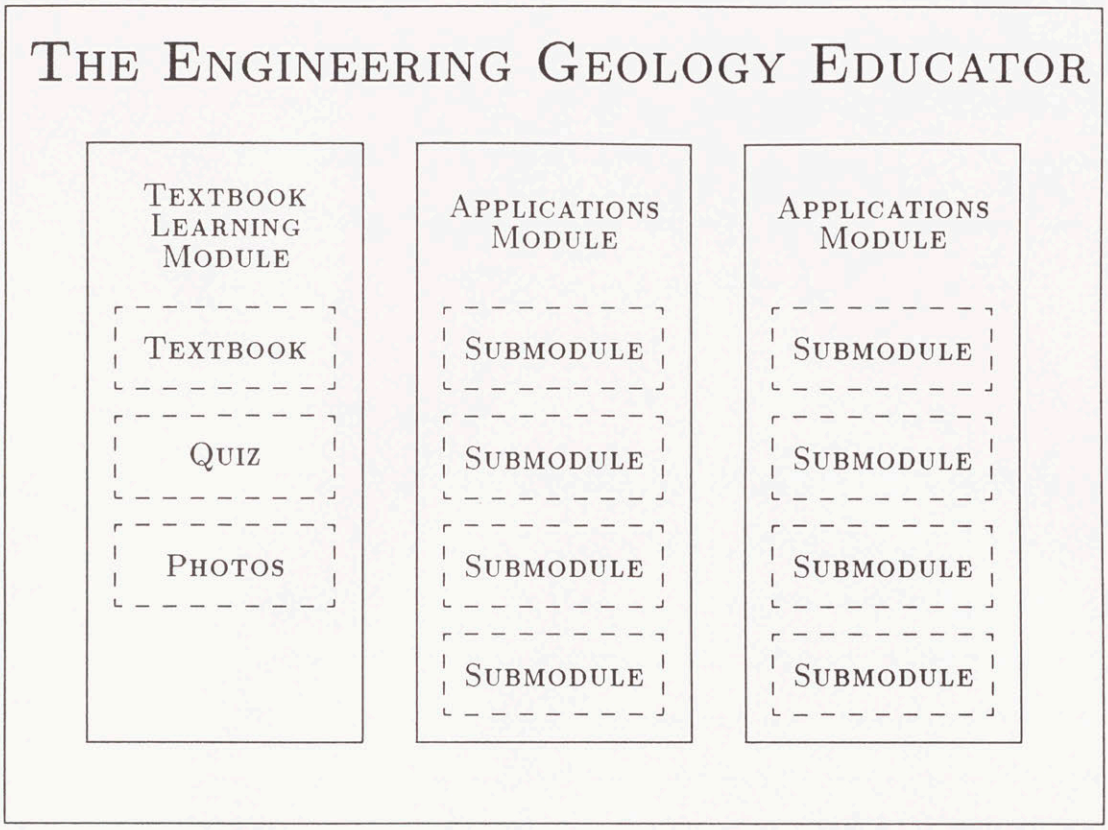

Figure 1-1: Structure of the Engineering Geology Educator

1.2 The Engineering Geology Educator

The Engineering Geology Educator is the system of educational materials produced for engineering geology, using various tools in the Athena environment. It consists of many different modules, as shown in figure 1-1, some of which are still under development.

1.2.1 Textbook-Learning Module

The Textbook-Learning Module is designed to familiarize students with typical geologic features and with underlying geologic processes.

It consists of three submodules. The textbook submodule presents students with written descriptions, drawings, and photographs of typical geologic features. The quiz

module asks students multiple-choice questions and then supplies the correct answers. The third submodule, still under development, provides students with photographs, and the student is asked to identify features by outlining, naming, or both. The schematic drawing in figure 1-2 illustrates the integration of these submodules.

This study is largely concerned with the first two submodules, with the textbook module discussed in Chapter 3, and the quiz module in Chapter 4.

1.2.2 Application Modules

A variety of application modules will allow students to employ the acquired knowledge. These modules will enable students to view a variety of geological and geotechnical data, through such devices as airphotos, borehole logs, and geophysical investigations. Data can be analyzed and manipulated to characterize a particular site. Additional modules allow the student to explore the interactive effects of geology and engineering. Rule-based expert systems (that is, systems based on artificial intelligence techniques) will provide the student with appropriate guidance. A more detailed discussion of the application modules is offered in Chapter 5.

1.3 The Platform

Work on the Engineering Geology Educator began during the development of the Athena environment. Throughout the Educator's design evolution, it has been necessary to work with experimental tools and to make adjustments for changes in the environment.

Until recently, there was no structured, coherent organization base, or design platform, on which to develop visual software tools.

In 1987, Project Athena began experimenting with a "construction set" for building multimedia learning environments. The goal of the project, called the Athena Muse system, or simply Muse, was to reduce the time and amount of skill needed to build such environments [Hodges, 1989].

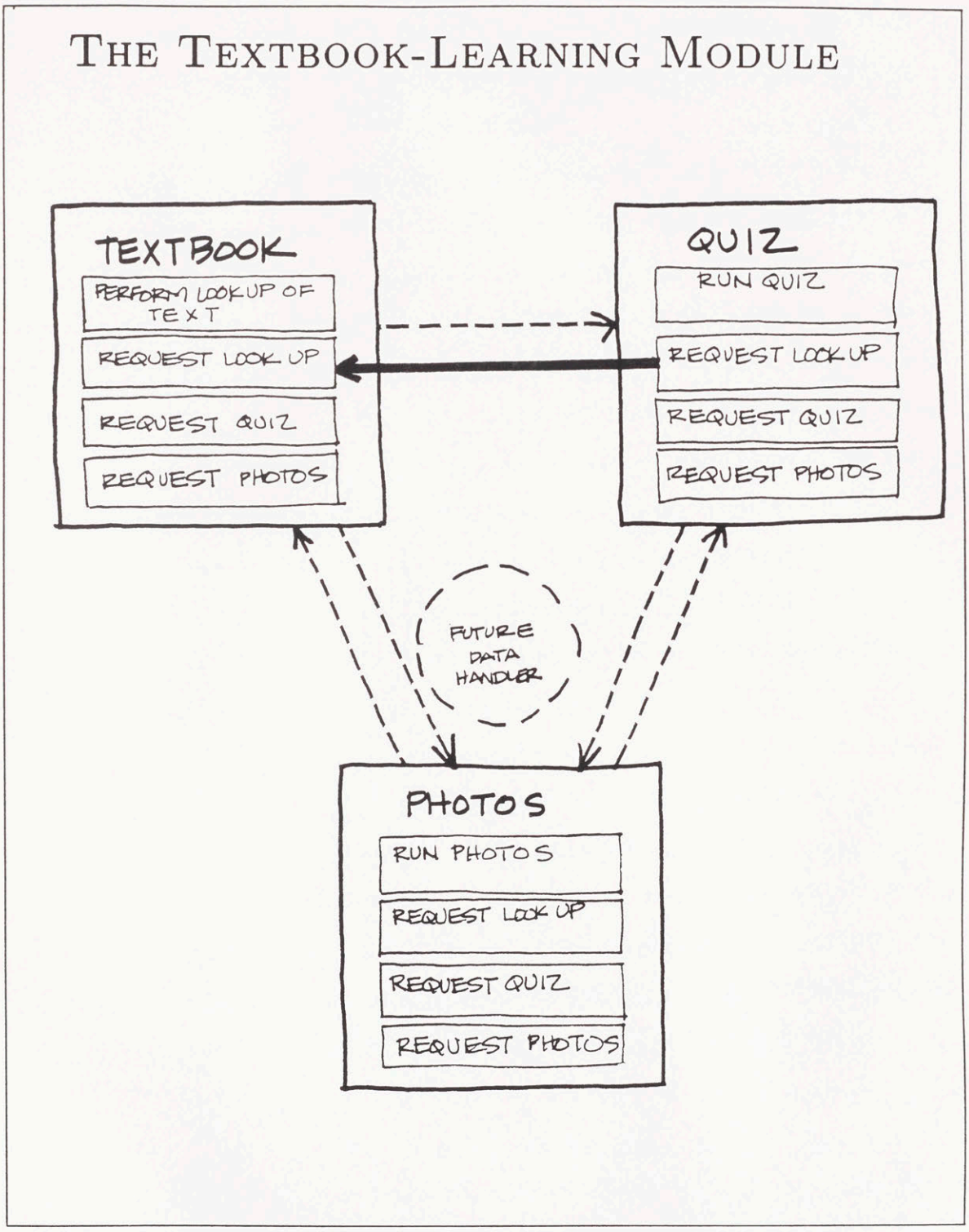

Figure 1-2: Linking between submodules in the Textbook-Learning Module

Muse provides three complementary "authoring environments," or three methods for changing the substance of the data or manner in which it is presented: a simple but constrained "data specification" environment, an intermediate "script" environment, and the highly flexible but complex C programming language.

In the "data specification," one specifies how each object will look (i.e. what it is, its size and location on the screen, border size, colors), at what point it is displayed, and possibly a simple function. These data are then used as input by Muse, which creates the objects, displays them on the screen at the proper point, and interprets all user input. Objects which have functions will perform these functions when the user presses a mouse button when the on-screen cursor on the the object. (This is called "pointing to" an object, or "pressing" a button.) This environment reduces development time, but is highly constraining, as the user is limited to existing in the current Muse version.

In the "script" environment, instead of having a single function executed when the user points to it, an object can execute a series of commands, called a script. In addition to being a list of commands, there can be logical comparisons and some limited use of variables. somewhat like a simple programming language.

The C programming language environment can be used to accommodate most special needs when the previous two environments fail to provide sufficient flexibility.

Some early Athena experiments concentrated on technical aspects of developing limited tools without much emphasis on how they would be used. However, the Muse platform concentrated on its applications, and tools were designed to solve problems encountered by users. This particular platform was chosen for the Engineering Geology Educator because it provides flexibility and facilitates further development.

Chapter 2

General Design Concepts

This chapter discusses programming and system design philosophy in general terms. These concepts, although somewhat intangible, must be applied to the design of the specifics of any part of the system. This chapter explains the framework of the entire system and gives references to specific examples in the Textbook-Learning Module.

2.1 Integration of Modules

The Engineering Geology Educator has the potential to be a comprehensive and powerful system. It consists of many diverse elements, each representing or analyzing a certain area or field of data and design. The extreme diversity of the data and procedures required in engineering geology demands special attention.

In the development of this educational system, the processes used in engineering geology are broken down into modules. Each module is a tool of some kind, allowing the user to access reference materials, practice rote or interpretive skills, develop complex representations and characterizations, or to perform analyses. A single module may be developed in isolation. However, in order to create a broader system that incorporates a number of functions and for that system to act as an integrated whole, there must be defined communication pathways between modules.

Since all work in engineering geology is dependent upon data, the manipulation of these data is the natural basis for the system. Each module is assigned a particular task that is some facet of this manipulation. Some file and organize site description or investigation data, while others analyze or refine this data.

The modules, building upon one another, create a complex characterization of the problem, working toward the final solution. Therefore, it is of paramount importance that the methods for intercommunication between the different modules are well designed.

At present this is achieved by the lookup function, which is a part of the textbook but can be accessed by the quiz and the photos (see figure 1.2). This present linkage concept and possible extensions will be discussed in more detail in the following sections.

2.2 Data Interchange and Linking

Because data handling is the most significant problem in constructing an integrated system, this should be the first consideration in the construction of any module. Only after a specification of the module's input and output requirements is formulated can the internal development of the module begin. These specifications can be modified during development, but the impact on the entire system must always be considered and accounted for.

Once the input and output structures of the module have been defined, the interchange of data serves as the links between the modules, and it is possible to have a fully integrated system.

In the Textbook-Learning Module, there are relatively few data which can be interchanged. However, the lookup function in the textbook can be used to search for information on a particular topic, and section 3.3 describes how it has been constructed to allow linkage to other modules. The quiz makes use of this linkage as described in section 4.4.

2.3 Data Handler for Complex Operations

Since the Textbook-Learning Module does not require complex data representation, it has been relatively easy to keep this module flexible. The current system implements inter-module communication though direct links between modules as described above. A further step would be to incorporate the lookup function into all modules. For this reason, the lookup function to request information from the textbook has been written to allow its placement into any module.

As the system becomes highly developed, it will no longer be practical for modules to communicate directly with one another. At that point, the information will be too extensive for communications to be handled effectively by individual packages.

It will then become necessary to delegate much of the common data handling to a special module designed for this purpose alone. This module, basically a top-level data handler, would be responsible for maintaining the database characterizing the problem (see figure 2-1). Modules would continue to process data internally, but the source of and outlet for data would be this data handler.

In addition, the handler would be responsible for notifying other modules when relevant areas of the database have been modified. This implies that the handler must be capable of tracking all data in order to have a complete representation of the problem. In the development of modules this eventual evolution should be considered, and data should handled internally in structured, modular routines which can be separated for integration into the larger system.

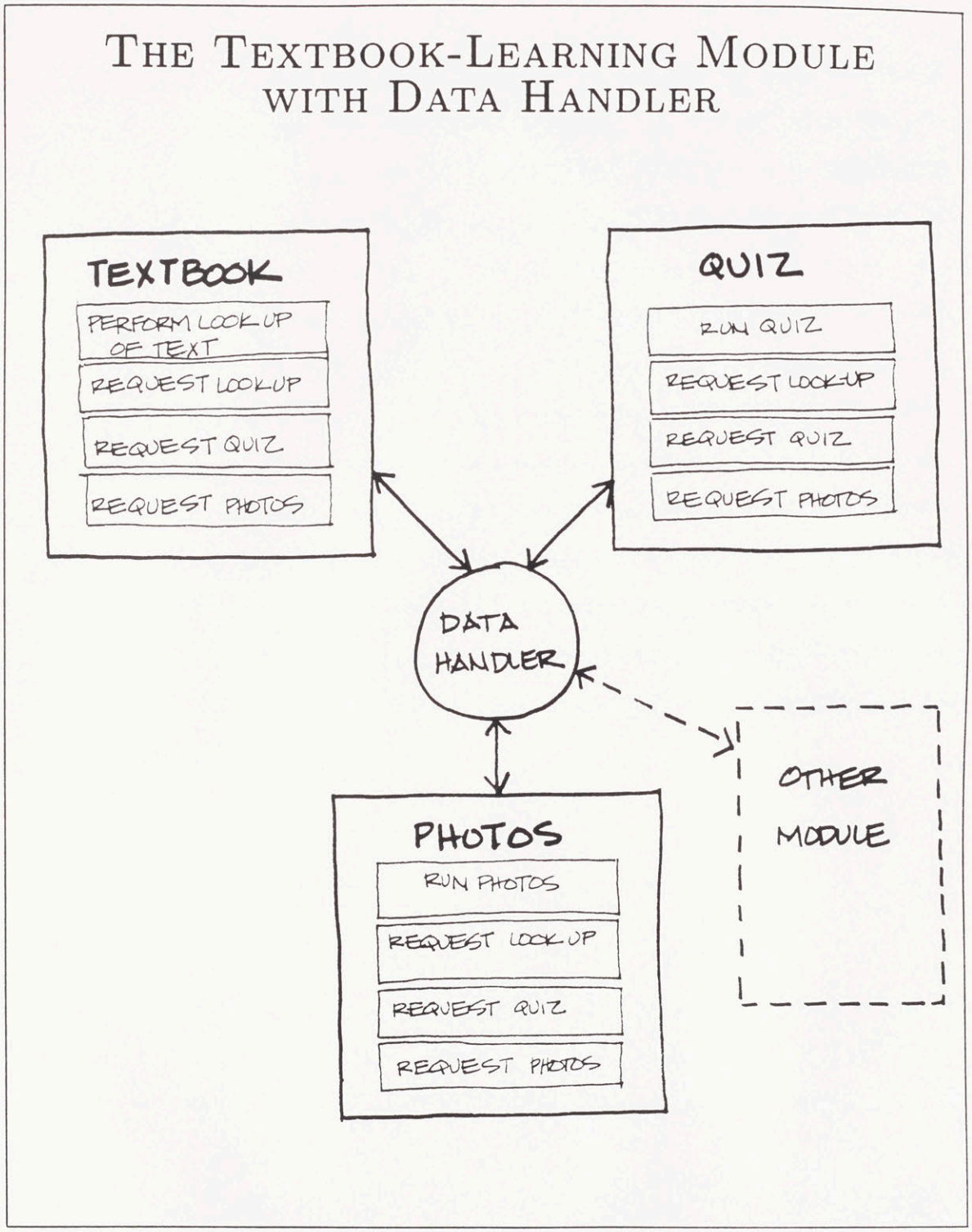

Figure 2-1: Future linking between submodules in the Textbook-Learning Module, using a data handler module.

2.4 Structure

In order to accommodate updates and expansions in the system and the unforeseen needs of future users, the structure of the system and its modules must provide maximum flexibility. Development of experimental software in the Athena environment is exceedingly susceptible to significant shifts in what tools are available. The potential for problems is compounded by the diverse nature of the data used in this project, and because numerous people have been and will be working on it. Therefore, the designer must pay strict attention to the structure of the program, so that it can be later adapted by others. Only by designing programs with a solid, modular structure, can the developer create a flexible, coherent system, capable of changing to meet the expanding needs of the system.

The design of the Textbook-Learning Module has adhered strongly to these principles. The textbook, described in chapter 3, has been designed and implemented as a highly flexible system. Details of the input and output structure and programming methodology are given in Appendix A. Chapter 4 discusses the design of the workbook, which, although not complete, uses a similar design, and strives to provide a framework which can accomodate further expansion.

Chapter 3

The Electronic Textbook

3.1 Overview

3.1.1 Description

The textbook module is an interactive video Muse package which serves the same general purpose as a printed textbook. It contains pages of written text, drawn figures, and photographs. The user is provided with tools for reading through the book and looking up specific information. As in a printed textbook, there are chapters, sections, indexes and tables of contents.

However, the computer is a much different medium than print, and provides unique advantages. For the student, the computer provides faster, simpler tools for looking up information. For the instructor, making changes to the textbook materials can be as easy as changing class notes. Since this module is part of a larger educational system, it can be linked to powerful tools for ground profiling, data interpretation, and engineering design, among others.

Additionally, the computer can present other forms of communication, such a moving video, and sound. Although not used in the current version of the textbook module, moving video is an important enhancement provided by the system. In next version of

Table of Contents		List of titles of all chapters
Chapter 1	Contents	Detailed contents for Chapter 1
	Body	Text descriptions
	Index	Index of entries in Chapter 1
Chapter 2	Contents	Detailed contents for Chapter 2
	Body	Text descriptions
	Index	Index of entries in Chapter 2
	⋮	
Last Chapter	Contents	Detailed contents for last chapter
	Body	Text descriptions of feature
	Index	Index of entries in last chapter
Appendices	Index	Index of entries in all chapters

Figure 3-1: Organization of the textbook.

the textbook currently under development at MIT, some of the more complex geologic processes will be presented as live and animated video. There may also be demonstrations of identification of minerals, rocks, or geologic features.

3.1.2 Organization

The organization of the material presented in the textbook is shown in figure 3-1. The first page of the textbook is a table of contents for the entire book, listing the title of each chapter. Then, each chapter begins with a more detailed table of contents, followed by the text descriptions of individual features, such as "Drumlin," "Kettle" and "Sinkhole," forming the body of the chapter, and ending with an index of features from the chapter. The last chapter is followed by the appendices. In the current version, the only appendix is an index of terms for the entire textbook, but a references list, a bibliography, or a user's manual may be included in future implementations.

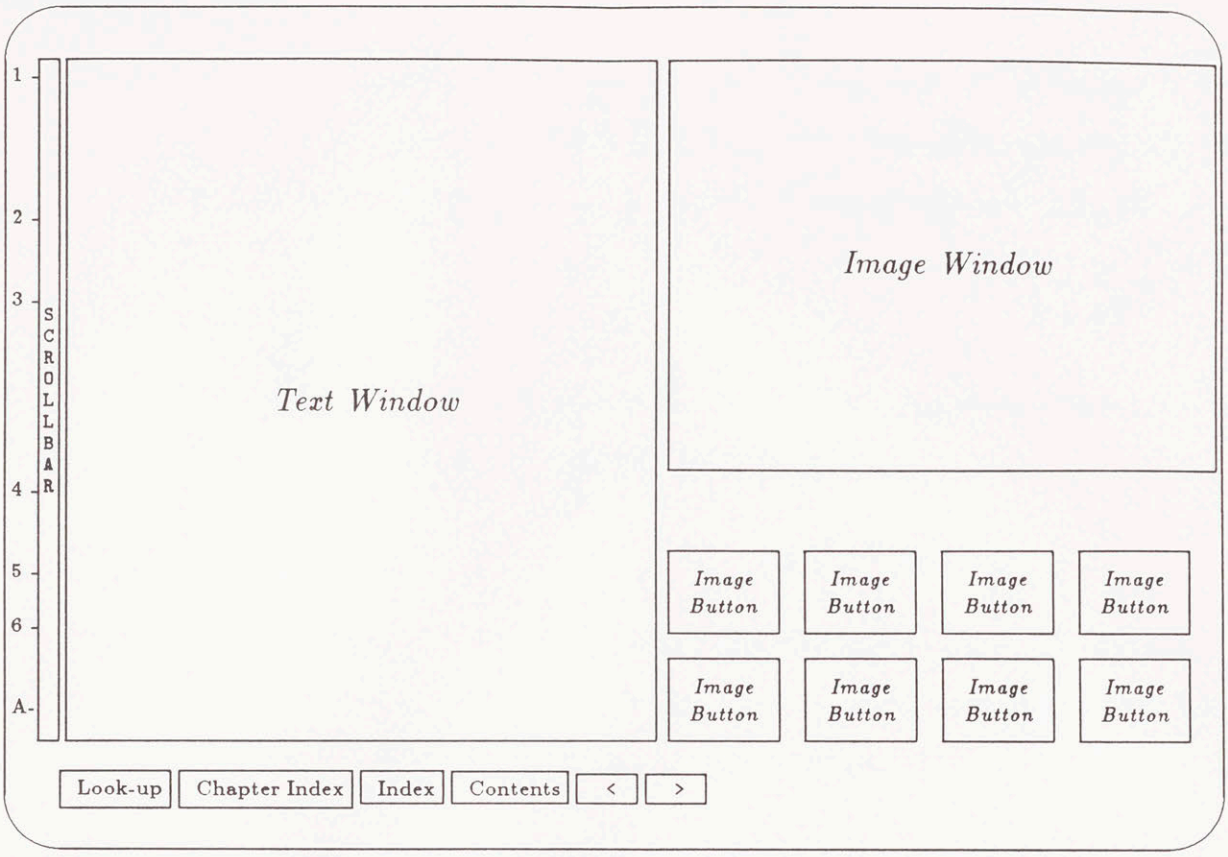

Figure 3-2: Layout of the screen while running the textbook. (Not to scale - Text Window is streched horizontally to show buttons)

3.2 User Interface

This section gives a somewhat detailed description of the appearance and function of the textbook, from a user's perspective. Making this system serve the user in an efficient, logical manner has been a primary design objective.

3.2.1 Layout of Screen

The materials displayed on the screen, whose function is described in section 3.2.2, give the user information, control, or both. The text and the images are the primary information provided by the system. These are displayed in the two large windows (areas of the screen) shown in figure 3-2. Control buttons (small windows where one points for functions) for changing displayed text appear below the text window. Along the left side

of the text, a scrollbar (a graphical indicator of text position) shows the current location in the text, and "beginning of chapter" buttons provide reference and control. Below the large image window, there are smaller "image buttons" showing available images.

3.2.2 Selecting Text

The text window is used for displaying "pages" of text. Most pages are descriptions of geologic features, but the contents and indexes are also pages.

3.2.2.1 Control Buttons

The buttons described in this section are located below the text window, and are used to turn to a different page of the book. The user can turn to any desired page of the book by using these buttons or those described in the next section.

Previous or Next Page The simplest way to move through the textbook is by turning pages. There are two buttons labeled "<" and ">" for turning back one page and forward one page respectively. These buttons can be used at any time (except to move beyond the beginning or end of the book). Thus, it is possible to view the entire textbook, page by page, using only these buttons.

Contents The button labeled "Contents" turns to the main table of contents for the book, which is the first page.

Index The button labeled "Index" turns to the index for the whole book, which is located after the last chapter.

Chapter Index As noted in section 3.1.2, there is a separate index for each chapter. The button labeled "Chapter Index" turns to the index for the current chapter.

```
2.1
2.1.1
Gound Moraine
Terminal Moraine
Recessional Moraine
Lateral Moraine
Drumlin abc
Boulder Train
Erratics
Till Deposit
2.1.2
Esker
Kame Terrace
Kame
Crevasse Deposit
Kettle
Drumlin abcdef
Boulder Train
Erratics
Till Deposit
2.1.3
Outwash
Valley Train
Braided Stream
```

Figure 3-3: Excerpt from look-up key file.

Look Up .

The function of the button labeled "Look Up" is comprehensive and powerful. When the user chooses a word or phrase from the index or table of contents and presses the look-up button, the textbook will jump to the page of the text describing this item.

To use this function, the user must first highlight text shown in the text window, using the mouse, then press the look-up button. Muse tries to find the highlighted text in its list of look-up keys. (An excerpt of this list is shown in figure 3-3.) If a match is found, then it turns to the appropriate page of the book.

Currently, there are two valid types of look-up keys. Any line found in an index is a

valid key. Any section number listed in a table of contents is a valid key. Additionally, the user can try to look up words in the text. If in future versions an indexing command is implemented this will provide the user with even greater flexibility to find information.

As described in section 3.3, this look-up function can also be invoked by other modules, providing a one-way link between the textbook package and the rest of the system.

3.2.2.2 Scrollbar and Chapter-finders

These controls are located directly to the left of the text window. Like the control buttons just described, these are used to turn to a different page of the book.

Scrollbar There is a narrow vertical bar with a marker in it, called a scrollbar. The bar represents the whole book, with the top being the first page and the bottom being the last. The marker moves to indicate the current page of the book. The scrollbar can also be used to set the page by pointing to it. It allows the user maximum flexibility in selecting a page. The numerals and the letter "A" to the left of the scrollbar, called "chapter finders," indicate where chapters and the appendices begin.

Chapter Finders These "chapter finders" are not only graphic indicators of the beginning of chapters, but are also control buttons which can be pressed like those below the text window. Their function is simply to turn to the first page of the selected chapter or appendix.

3.2.3 Selecting Images

For every page of text there may be associated images. If there are no images for a particular page, the right side of the screen remains blank. If there is one image, it is displayed in the large image window.

If there are two or more images, all the available images are displayed in postage stamp sized window-buttons in the bottom right corner of the screen. The first image is

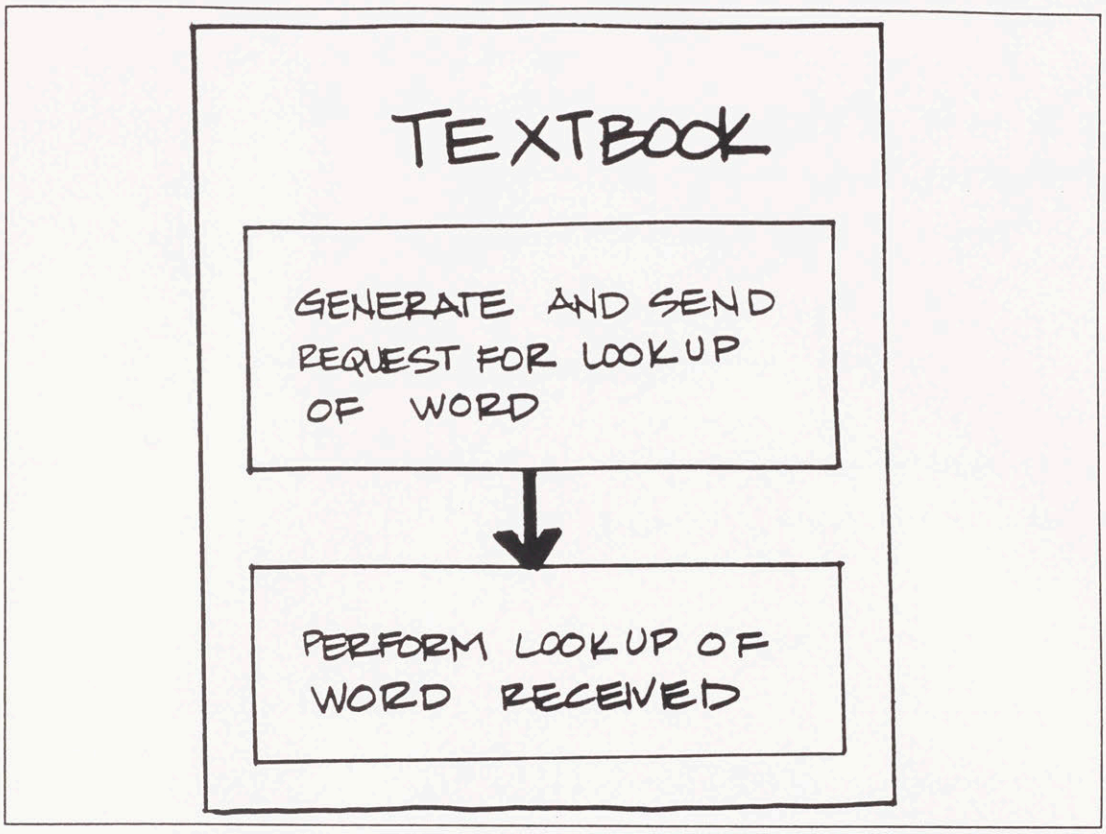

Figure 3-4: Functions of the Look Up button in the textbook. Two distinct functions are used to look up a word.

also displayed in the large image window. The user can select any of the small images for display in the large window by simply pointing to it.

3.3 Linking to Other Modules

Use the look-up function is described in section 3.2.2.1. However, this function is more powerful than it first appears. It is not simply a function through which the book resets its own position – it is actually two functions working together. This is shown schematically in figure 3-4.

The first function is activated by the user pressing the "Look-up" button. This generates a request for the textbook to look up a word. It then sends this request, which

includes the word, to itself. The textbook then receives its own request and the second function looks up the word to fulfill the request.

Because the word is included in the request, it is possible for other modules, such as the quiz or photos submodules, to send requests to the textbook using only these existing functions. These requests will have the same format as above : look up a word.

3.4 Construction of the Textbook

This section gives an abbreviated description of the textbook creation system, mutex. A detailed description of this system is given in Appendix A. Figure 3-5 is a schematic diagram of the whole system.

3.4.1 Preliminary Work

The first step in the creation of the textbook was to write the text and to select images. This work was begun by Don Gray and Prof. Einstein [reference]. Since new organizational structure was designed as part of this project, much of the previous work needed to be reformatted.

3.4.2 Text

After the materials have been collected and organized, it is necessary to formulate a description of the textbook which can be converted to Muse. A primary design objective of this system was to enable the textbook authors to create and edit their book without a knowledge of the internal workings of Muse. Therefore, the basic format for the textbook is patterned after text processing programs.

To create a book, the author creates text files to be used as input to mutex. These files contain commands such as `\chapter{Glacial Geology}` or `\section{Erosional Features}`, the actual text descriptions, and lists of figures. Mutex then processes these commands, producing the complete Muse textbook package.

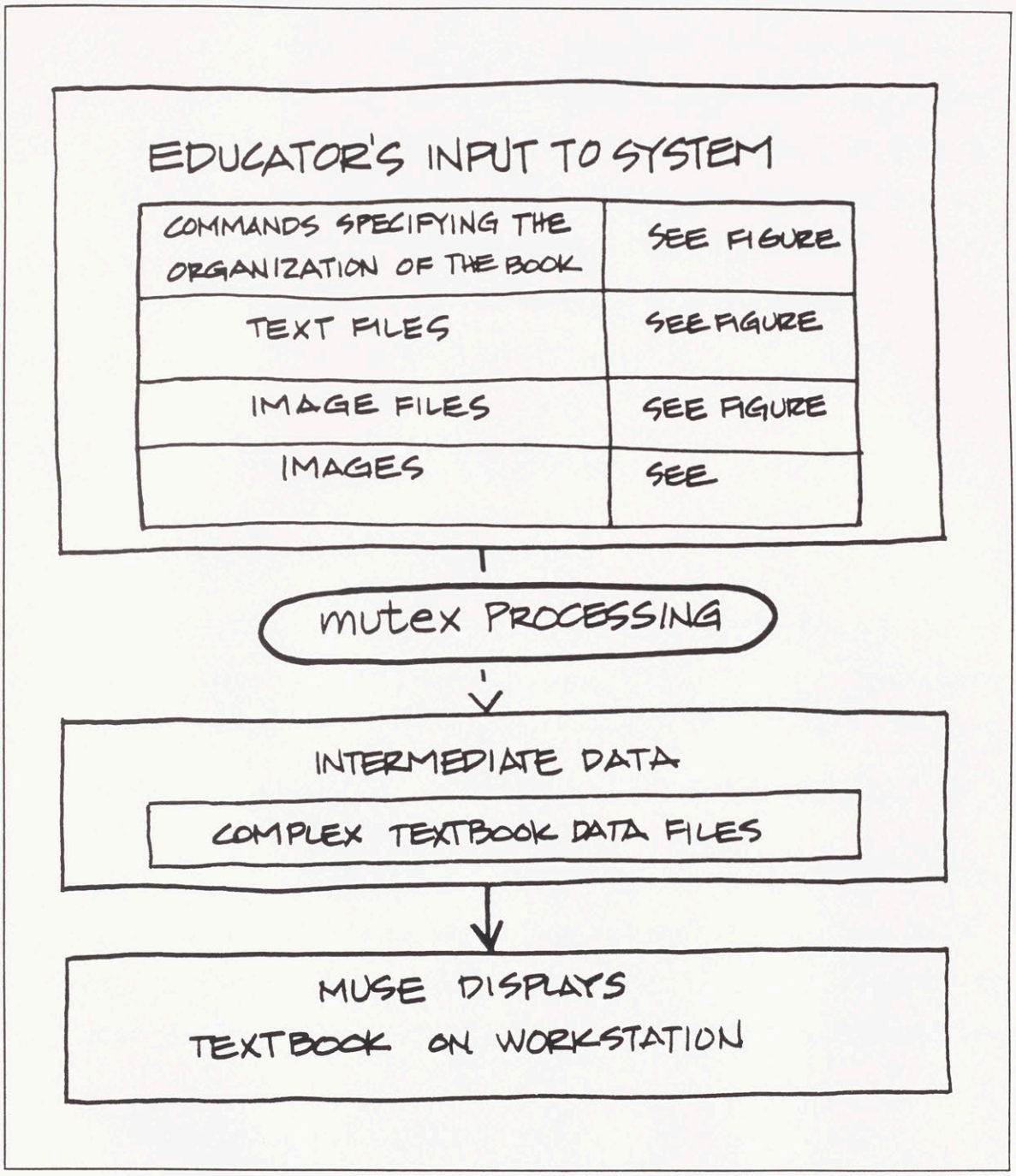

Figure 3-5: Function of different parts of the textbook preparation system.

3.4.3 Images

Images can be stored in any format supported by Muse. Currently the only supported source of images is the standard United States television signal, NTSC. For the textbook, NTSC images are stored on a video disc. These images can be created from a variety of sources, and for the textbook, are photographs and drawings. Images are linked to individual entries of the text by commands, such as `\image{Braided Stream}` stored with the text. When mutex processes the text files, these images are linked in the Muse package. A detailed description on how to link images is given in Appendix A. Image creation and storage is described in Appendix B.

3.5 Updates and Enhancements

3.5.1 Modifying the Images and Text

To modify the material in the textbook, no special knowledge or skill is required. The person doing the modifications simply makes changes, such as adding or deleting new images, text descriptions, or even entire chapters, by editing the text files as described in section 3.4 and running mutex. It is possible to create a revised version without disturbing the current working version of the textbook.

For example, if a new chapter is desired, the constructor of the text can simply make a copy of the current version of the textbook. Then new material can be added to the copy by typing in text descriptions, creating images, and specifying how these are to be organized. When the new version is complete, it can be used in place of the old version.

3.5.2 Adding Moving Images

Muse supports moving video, and the next version of mutex will include a video display function. The structure within mutex already exists, but full implementation was not needed for the current version of the textbook.

To add animated video, live video or film to the textbook, it is necessary to have the images on videodisc. Like still images, moving video will be linked to the specific pages of text by adding commands specifying what is to be displayed.

3.5.3 Other Possible Applications

Since mutex, which is described in detail in Appendix A, has been designed to be a simple, flexible system for the author of the Engineering Geology Educator Textbook, it is possible to apply it also to constructing textbooks for other subjects. The fundamental structure of mutex, although designed for engineering geology, is essentially the same as needed for most textbooks. It is hoped that mutex will become a sort of "third party" software contribution to other developers using the Muse platform.

3.6 Summary

All features and capabilities described in this chapter have been implemented, at least in a prototype version. Except for what is specifically noted here, all features and functions work in the production version. Images have not been completed, and have not been linked. "Dummy" images have been used successfully in their place for the development of the display system. References for images have been implemented only in the prototype because references will be entered when the images are linked to the text.

Chapter 4

The Workbook

4.1 Overview

The purpose of the workbook module is to provide students with a means of practicing what they have learned in the textbook module.

The first part of the workbook, called the "quiz," asks the student multiple choice questions. The second part involves outlining features in images, and is discussed in Chapter 5. This chapter describes the work in progress on the quiz.

4.2 User Interface

For each chapter of the textbook, there will be several sets of questions focusing on part of the chapter. The student will select which quiz he or she would like to take.

The computer will then display pages of the quiz, shown in fig 4-1. each with one question and a set of possible answers. The student will select answers by pointing to them, and will have the ability to page through the particular quiz answering other questions, as well as checking and changing any answers.

When the student feels ready, he or she can press a grading button. This causes the computer to display correct answers along with the student's answers. To view the

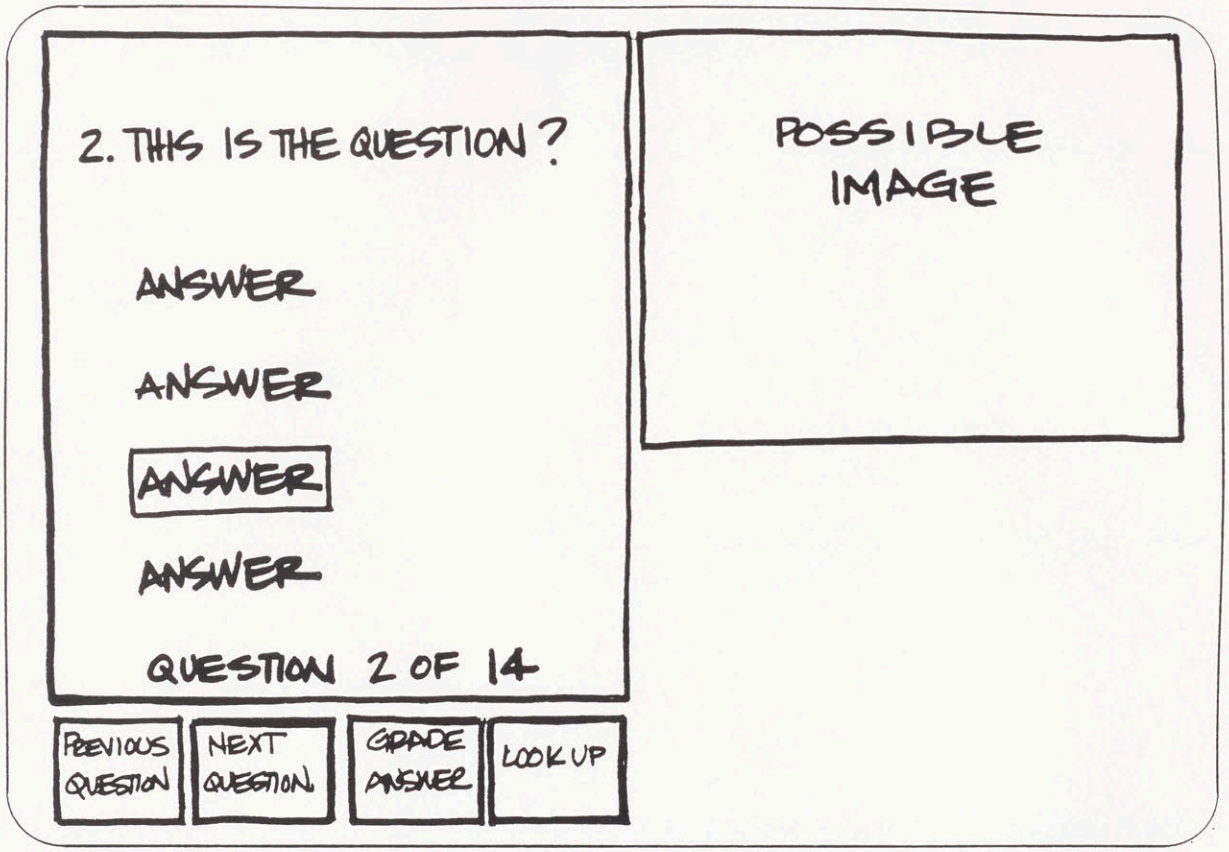

Figure 4-1: Layout of the screen while running the quiz. (Not to scale)

corrected answers, the student pages through the quiz, forward or backward, as before. Currently the only function of the grading button is to indicate correct answers (by pointing an arrow to the correct answer) together with the student's answers, but future versions may allow the student answers to be checked automatically by the computer, or saved for future reference by the student or professor.

4.3 Constructing a Workbook

To construct a quiz, one needs to list questions and answers in a file. There are only three commands in the quiz:

`\question{`*Question text*`}`

`\right{`*Right answer*`}`

`\wrong{`*Wrong answer*`}`

The file consists of a question followed by a right answer and one or more wrong answers.

4.4 Putting it Together

The program to create the quiz uses many of the same functions as the textbook program. Indeed, it is being developed as a special case of the textbook. Many functions are reused, and the overall design structure of listing commands is the same. To facilitate linking, the quiz will be able to send a request to the textbook asking the textbook to display information on a particular topic. This function will work the same way as the textbook looking up entries in its own index. This is illustrated in figure 1-2 and in detail in figure 4-2.

Thus, the basic structure of the input has been designed, and the function to make the screen layout from this input has been implemented. Preparation of the questions is

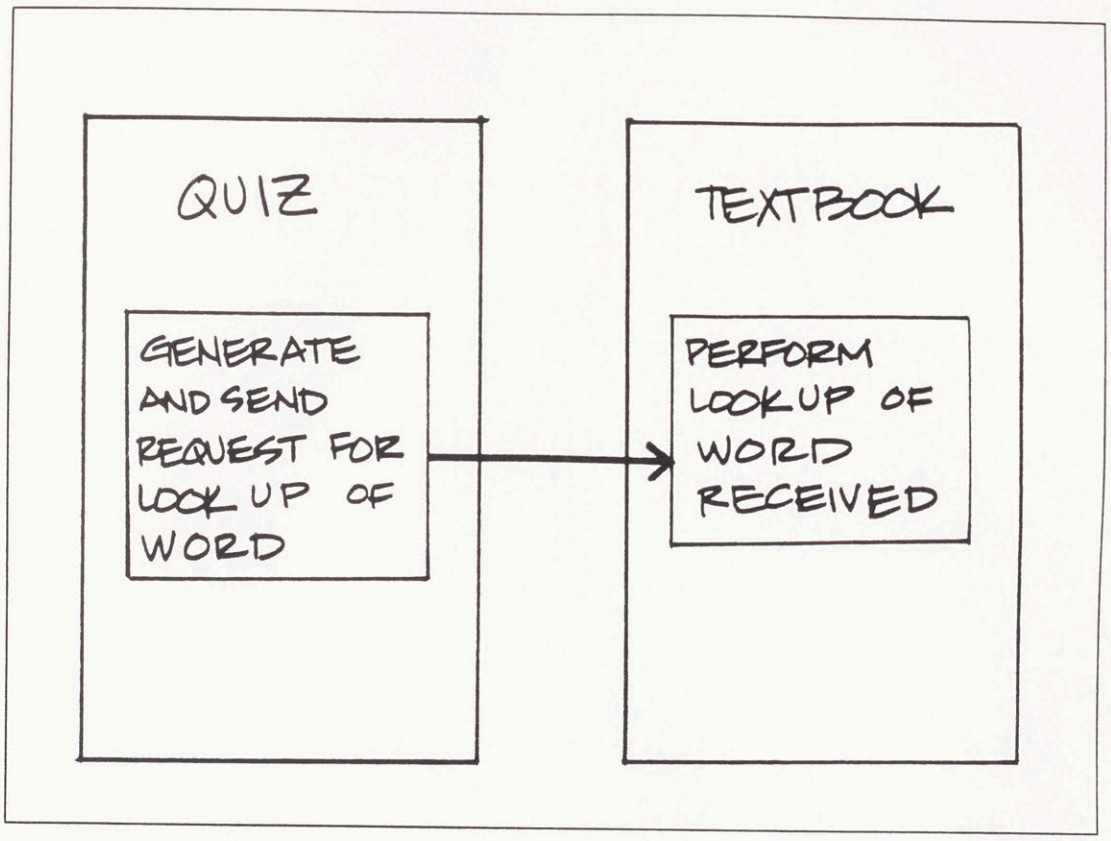

Figure 4-2: Linking of the textbook and quiz.

underway, and this should be finished by the end of May 1989. Checking and correcting the student's answers is not yet possible in the Athena environment; however, Project Athena is expanding the Muse system to provide this capability for grading, and expects this to be done by June 1989. Thus, the integration of grading with the questions will be completed as soon as the tools are available.

Chapter 5

Other Modules

5.1 Existing Modules

5.1.1 Photo Identification

The photo identification section is a submodule of the the Textbook-Learning Module. The current version allows students to select a photograph and then asks the student to identify geologic features by naming outlined features, outlining named features, or by outlining and naming features. Although this module already exists, it needs to be revised to enhance capabilities. This module was written using tools which are no longer compatible or supported by the latest Athena environment. The method for displaying images has changed from digital to video, which requires that the images be reentered into the data base (see Appendix B for details). In addition, the ability to draw lines on images is not yet possible with Muse.

5.1.2 NOMAD

The purpose of NOMAD is to aid the user in the construction of a three-dimensional ground profile based only on initial borehole information [ref carlos]. NOMAD allows a user to input selected borehole information and create a reasonable ground profile

through different graphic and editor capabilities. A link to an expert system can be used to evaluate the proposed ground profile and critique its validity based on a set of knowledge rules.

Information can be input by hand or from a data file. During the operation of NOMAD, data can be output into a file. A three-dimensional ground characterization is created through linear connections between borehole ground information to form layers. The user can then change and correct the profile, or check the profile using an expert system link.

One potentially serious problem with NOMAD is that it was developed upon a platform that is not supported by Athena. It makes use of three-dimensional graphics routines that do not currently exist in a supported environment. Although development work in this area is proceeding, the new three-dimensional platform may have significantly different functionality. This will probably necessitate rewriting at least portions of the code, and perhaps the entire package.

Additionally, NOMAD was not developed in the context of an entire system, and cannot be controlled by external programs. It would be useful to allow other modules access to the capabilities of NOMAD directly, rather then limiting it to human interaction. The data streams should also be reconstructed to allow more flexible data exchange.

As noted by its author, NOMAD also has limitations as to its ability to handle different types of data [ref carlos]. Design of a general format, suitable for representing all types of data, is probably a difficult task, but steps must be taken in this direction if a truly integrated system is to succeed.

Other functional limitations and possible enhancements, such as a different layer boundary representation, are discussed in detail by its author [ref carlos].

5.2 New Development

5.2.1 Compatibility Requirements

As emphasized in this thesis, the initial planning stages of a new module should consist of defining the data representation of the input and output of a module, or how its function will affect an external data set. In order for modules to be compatible with one another, this data representation must be carefully designed.

5.2.2 Other Modules

This section describes some additional, potential modules for development. For the most part, the methodology already exists; it is the computer implementation that has yet to be developed.

5.2.2.1 Surface and Airphoto Interpretation

An extension of the photo identification module, this section allows students to use the photos to identify general as well as particular geologic features. The existing set of rules judges the the student's interpretations, and will eventually be extended to include these interpretations, the class discussions and corrections from the teacher. This module is similar in format to a case study, and allows the students to practice the application of knowledge acquired through the tutor.

5.2.2.2 General Data Interpretation

This submodule provides students with short geologic reports, excavation reports and selected boring profiles, and requires the student to make verbal and sketch interpretations. These interpretations are used to update the individual base of knowledge on the rules associated with geologic interpretation. The student will not only have learned these rules, but will have access to geologic "expert" knowledge.

5.2.2.3 Planning Further Investigation

In this submodule, the student is given a certain amount of money and cost figures for different exploration methods. By experimenting with different methods, the student learns which of these methods, or combination of methods, is most effective for acquiring information under the given monetary and physical constraints. This module is an exercise in recognizing the advantages and limitations of different exploration methods under realistic conditions of uncertain information, and it will teach students how to effectively apply knowledge acquired through the tutor.

5.2.2.4 Engineering Consequences and Exploration

In this module, the student explores the effects of geology upon engineering, and vice versa. The student, in the first submodule, is presented with geologic information and is asked to choose a site for a given project. After the site is chosen, the student, in the second submodule, determines the construction aspects of the project. Given costs for different types of foundation work, the student decides which method or methods of construction will be utilized and derives total cost for the project. In the third submodule, the student selects different exploration procedures and uses the results to calculate potential changes in construction costs. These cost differentials are then compared to the costs of different exploration methods, and, in this process, the student learns to estimate exploration results prior to the actual performance of this exploration.

Appendix A

Mutex

A.1 The Problem

To run the textbook, Muse input must be created. However, Muse resembles a programming language, with complex internal structure, and which cannot be easily understood. It is necessary to have a description system that can be easily used by the educator creating the textbook. If the textbook were created directly in Muse, even if the educator learned the Muse structure, minor modifications would demand an inordinate amount of tedious work. Therefore, the mutex system was developed to translate a simple, organized, human-oriented description into the cryptic Muse form.

The mutex system both provides a tool for the constructor of the textbook, and the input to muse. It consists of a defined set of input commands, text and figures; a program, mutex, to interpret these; and the output of mutex, which is the input to Muse. This is shown schematically in figure A-1.

It was realized in the development of the system that this need also exists in the preparation of printed documents. Concepts for the format of the input were taken from a popular document preparation system named LaTeX which had taken these concepts from an older document preparation system named Scribe [Lamport, 1986]. Thus, the input to mutex and LaTeX is similar, but the processing and output differs greatly between

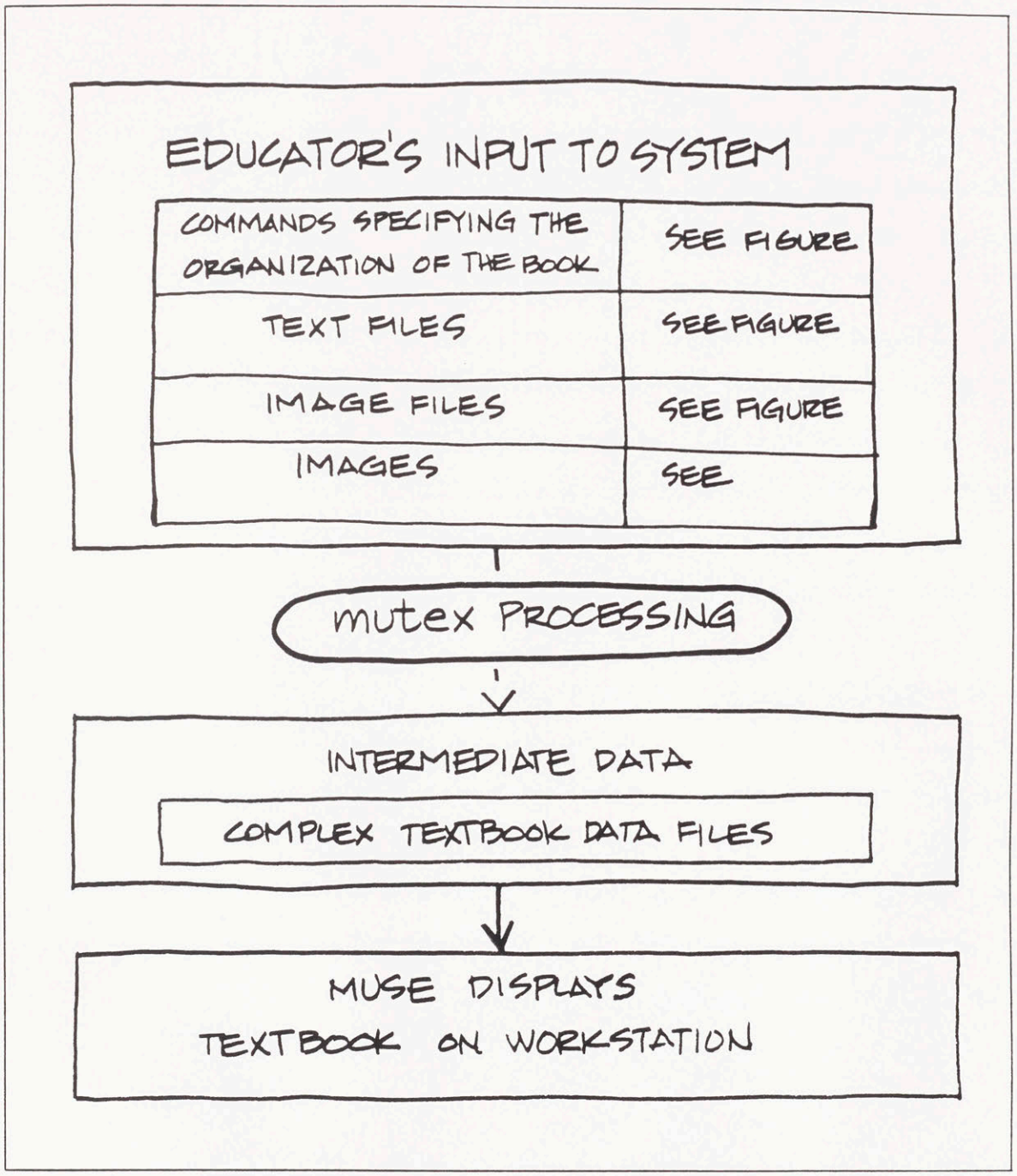

Figure A-1: Function of different parts of the textbook preparation system.

the two.

A.2 Mutex Input

Mutex input consists of three basic types of information. There is written text to be displayed, images associated with specific parts of the text, and commands which describe the logical structure of the text. This input is split into many files and referenced from a single main input source file, as shown in Figure A-2.

At present, the system requires a very structured set of files. There must be separate input files for structural commands such as `\chapter` and `\section`, for text descriptions, and for image commands, such as `\image` and `\caption`. The main input source file in figure A-2, is shown in figure A-3 and contains only commands referring to chapter contents files. An excerpt from one of the chapter contents files is shown in figure A-4, and contains only structural commands. An example of a text file is shown in figure A-5, and the corresponding image file is shown in figure A-6.

Mutex first reads the main input source file which lists the order and names of the chapter files. When Mutex reads the name of a chapter file, it begins reading that file, which contains commands defining section structure and references to text and image files. When it encounters a reference to text and images, it then reads those files. The following subsections discuss in detail how the text, images, and grouping commands work. Section A.3.3 explains how mutex interprets these commands, and section A.4 illustrates how these commands work together to fully describe the textbook. Details of the structure of the input are given below.

A.2.1 Text

The most basic elements of the textbook are characters. The text must exist in special files formatted manually by the author. (An example of a text input file is shown in figure A-5.)

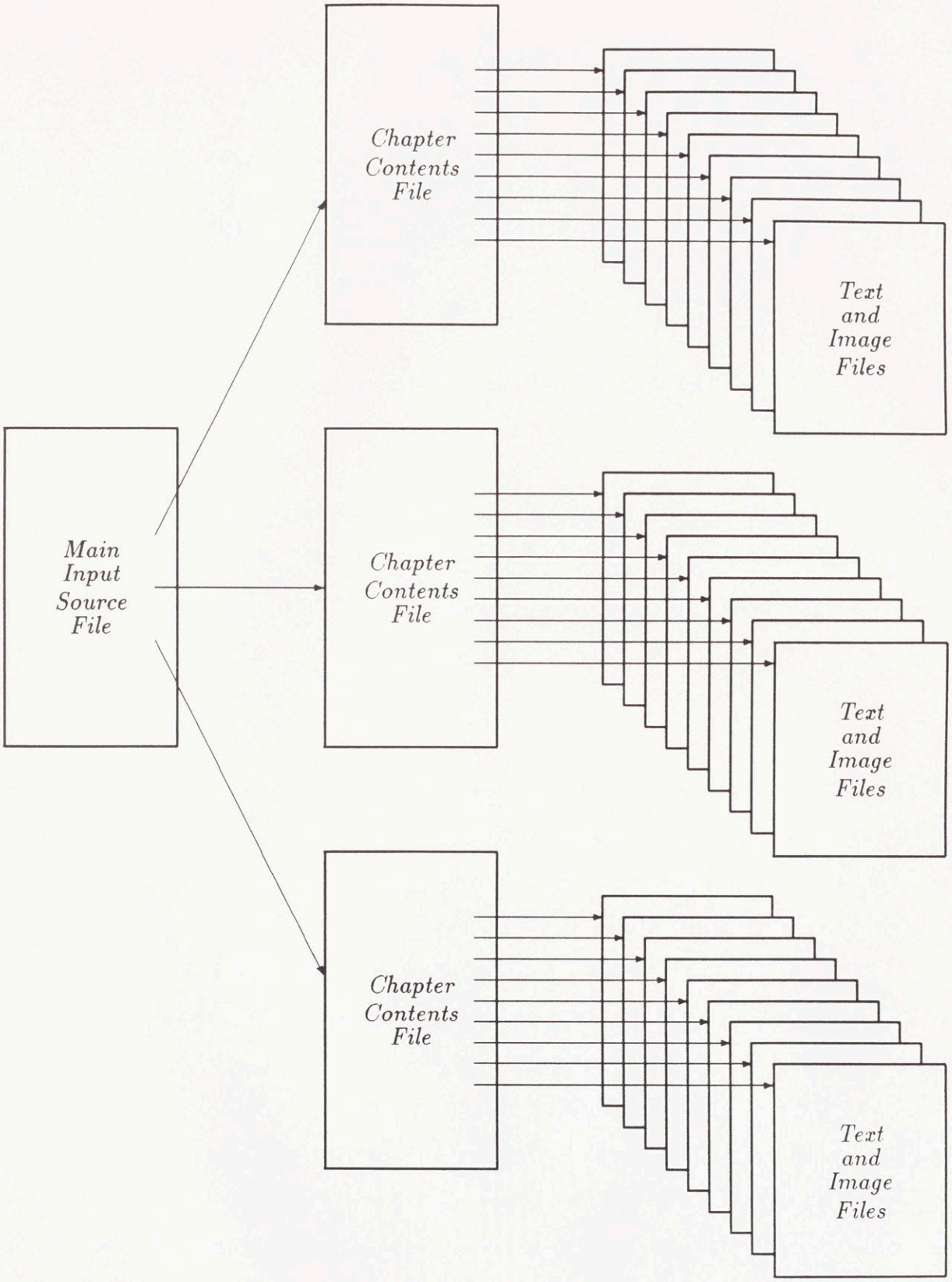

Figure A-2: File organization used for mutex input.

```
# Main table of contents for the textbook:

\contents{karst}
\contents{glacial}
\contents{fluvial}
\contents{shoreline}
\contents{lacustrine}
\contents{eolian}
```

Figure A-3: Main input source file for the textbook.

A.2.2 Images

Images related to a given text description are integrated by listing them in a file with a name similar to that of the text file. An example of an image information file is shown in figure A-6. In these image files, information must fit into the format described below.

For each image, there must be an `\image` command:

`\image{`*optional description*`}`

This marks the beginning of a series of commands relating to a single image and the end of commands relating to any previous image. The optional description, ignored by mutex, is space for the textbook author to put in a note, and is provided so that a verbal description of the image can be associated with this image. The braces, { }, are not optional and must be included even if there is no description.

Although image storage and retrieval are discussed in detail in Appendix B, a few points should be noted here to explain the image commands. Currently the only image display format Muse supports is video, while the earlier version of the textbook used a digital format. Images are being recreated for video format, but will not be completed and integrated into the new textbook until June 1989. The old digital images are stored in the Unix file system, and the new video development uses a recordable video disc called a "DRAW disk." Once development is complete, these images will be transferred

```
\chapter{Glacial}

        \section{Depositional Features}

                \subsection{Ice Laid Features}

                        \subsubsection{Ground Moraine}
                                \item{grnd_mor}

                        \subsubsection{Terminal Moraine}
                                \item{term_mor}

                        \subsubsection{Erratics}
                                \item{erratic}

                        \subsubsection{Till Deposit}
                                \item{till}

                \subsection{Ice Contact Features}

                        \subsubsection{Esker}
                                \item{esker}

                        \subsubsection{Kame Terrace}
                                \item{kam_ter}

        \section{Erosional Features}

                \subsection{Valley Glacial Erosional Features}

                        \subsubsection{Cirque}
                                \item{cirque}

                        \subsubsection{Arete}
                                \item{arete}
```

Figure A-4: Excerpt from the glacial geology chapter contents file.

```
KETTLE

    A kettle is a depression or basin in the glacial drift formed by
the ablation (melting) of a mass of glacial ice which was wholly or
partly buried.  These forms may have any shape in plan, although most
tend to be somewhat circular, and vary in depth from less than 8
meters to greater than 45 meters.  A kettle may be partially or
completely filled with silt and clay (varved clay).
    Not all depressions or basins in glaciated areas are kettles.
Basins or depressions may also form as a result of irregularly
deposited moraines.  Other basins, which are not kettles, may be
formed by the blocking of a pre-existing valley, differential thawing
of perennially frozen ground, and subsidence in bodies of silt.
```

Figure A-5: Text file for the feature "Kettle."

```
\image {Drawing of a kettle}        # indicates beginning of an image
        \digital {kettle.iff}       # filename for digital image
        \draw {2433}                # frame number for DRAW disc
        \optical {}                 # frame number for optical disc
\image {}
        \digital {outw_ket.iff}
        \draw {2452}
        \optical {}
```

Figure A-6: Image file for the feature "Kettle."

to an industry standard read-only optical videodisc. (These media are described in Appendix B.) One of the three following commands is used to tell mutex where an image can be found (Usually only one will be used.) :

`\digital{`*image filename*`}`

`\draw{`*frame number*`}`

`\optical{`*frame number*`}`

For development of the textbook, a DRAWdisc is being used, so mutex will look for only `\draw` commands and ignore the other two. The argument of the `\digital` command is the filename of an image stored on the Unix file system in digital format. The arguments of the `\draw` and `\optical` command are the frame numbers of the image on the two types video disc. The current version of the textbook does not actually include images because the old format cannot be displayed and the new format has not been completed. However, a test version using "dummy" images has been implemented, and no problems with the full working version are anticipated.

A.2.3 Combining and Sectioning Text and Images

The two files described in the previous sections, one containing text and the other an image list, together form one "item" of the textbook e.g. the verbal description of a geologic feature and associated drawings and photos. The textbook user views a single item at a time, seeing one text description and any associated images. These items are grouped into chapters by listing them with appropriate sectioning commands in chapter contents files, whose structure is shown in figure A-4 and described below.

Items are included in the chapter contents file with the `\item` command:

`\item{`*item filename*`}`

These items are organized, in descending order, into chapters, sections, subsections, and subsubsections. The mutex input file contains these commands:

`\chapter{`*chapter name*`}`

`\section{`*section name*`}`

`\subsection{`*subsection name*`}`

`\subsubsection{`*subsubsection name*`}`

In order for the table of contents to be numbered correctly, each item should exist within a named chapter, section, subsection, or subsubsection. Although it is not necessary to have a specific depth of sectioning, each section should be ordered so that there are no missing levels, and there should be only one item at the lowest sectioning level. This is necessary only because the names of the sections are listed in the table of contents, and will not be numbered correctly if the sectioning is not structured correctly.

Although sections and lower levels are strictly an organizational convenience, the chapter is a more significant level. Each chapter has its own table of contents and index. To facilitate production of the textbook, it is required that each chapter exist in its own Unix directory. Each directory, containing a single chapter, is listed in the main input source file using the `\contents` command :

`\contents{`*directory name*`}`

This command causes mutex to read the chapter contents file in the named directory, including the chapter into the book.

Thus, the main source input file tells mutex to read chapter contents files. Chapter contents files contain sectioning commands and item commands. Item commands tell mutex to read text and image files. (See figure A-2.)

A.3 Internal Function

This section explains how mutex processes its input and what it produces. These processes are complicated by the fact that mutex is a growing experimental system, and

parts of the design have been structured to accomodate growth which has not yet been implemented.

A.3.1 Input Structure

There are two distinct types of input to mutex: commands and text. Commands all take the form of a leading backslash, a keyword, and an argument enclosed in braces, and are read from the main source file, the chapter contents files, and the image files. Text is read from pre-formatted text files.

Input is separated in this way because the mutex system uses old data files from a previous textbook program [Gray, 1987]. The effort required to relax this constraint and the benefits derived from this effort are discussed in section A.5.

A.3.2 Output Structure

As mutex processes its input, it creates data files which comprise the Muse textbook: the input for Muse, the lookup function file, and text files for the contents, indexes, and the body of the text.

The basic units of Muse input are called "elements" and "maps." An element definition contains most of the information needed to create a display, such as the type, screen position, size, color, and contents. A map tells specifically at what point in the program's running a part of the element is to be displayed.

Mutex creates these elements and mappings in separate files for each different object on the screen, as described in the next section. It also specifies the size of the book and creates control buttons which are combined with a textbook skeleton to form the Muse textbook.

A.3.3 Processing

This section describes how the mutex program runs and the function of each mutex command. A sample of how these commands work is given in section A.4.

When mutex starts, it opens data files for each type of element to be created and initializes them when appropriate. In addition, it creates files for the table of contents of the book, the index of the book, pages of text, the image window, image buttons, and the look-up function, among others.

The input source is opened, and mutex begins reading the input stream. When a `\contents` command is read, the argument is taken as the name of a directory containing a chapter. The contents file of that directory is then opened, and input is read from that stream.

When a `\chapter` command is read, if a previous chapter was in progress, that chapter is ended, its files are closed, the index is sorted, paginated and added to the book, and a new chapter is begun. Files are opened for the chapter table of contents, the chapter text, and the chapter index. The chapter contents file and text file are added to the book, and the name of the chapter is added to the main table of contents and the look-up table. The chapter counter is incremented, and the section counter is reset.

When a `\section`, `\subsection`, or `\subsubsection` command is encountered, the counter for that level is incremented and the next level counter is reset. The name of the section is added to the chapter table of contents and the section number is added to the look-up key list.

The `\item` command causes mutex to process a text file and an image file, and to place the argument in the look-up key list. The text file is added to the chapter text file with the section number added at the top, and a Muse mapping is added to process its display. The image file is read, and, for the development system, the `\draw` commands are used in producing image window mappings. If there is more than one image for the item, image buttons are also mapped. Currently, the other commands in the image file do not produce any output.

When all input has been processed, mutex ends the last chapter, sorts and paginates the main index file, creates additional control buttons and the chapter finder buttons, and finishes all the element files begun at the startup.

There is one remaining task before the book is ready to run. During creation of the Muse elements, it is sometimes necessary to know how long the book or a section will be before the end of it is reached. This problem, needing information before it exists, can be resolved by a number of different techniques. One simple method would be to process the entire input twice, once to find out where elements begin and end, and the other to create output, but this would be slow, and cumbersome to program. An alternative would be to keep temporary lists and files containing parts of the output until the needed information enabled completion, but this would require delicate and careful organization of the code and output. The approach used by printed document preparation systems is to record this information in a separate file during one run and to use it in the next. This is fast if one makes revisions that do not change the length of a book or a section, but, because of this very limitation, it is not sufficiently robust to meet the flexibility standard of the system.

The solution chosen makes unconventional use of a conventional tool, the C programming language preprocessor [Kernighan and Ritchie, 1978]. Although rarely used outside the C programming environment, this is a general and powerful tool for combining files, making substitutions in text, and deleting comments. As mutex runs, when an unknown datum is needed for the output, it substitutes a special string of characters. Later, when this datum is known, a substitution command is written to a special file. After mutex has finished, the C preprocessor is automatically run, combining different files and making the appropriate substitutions. Because this system allows the programmer to use data before they are known, simple clear programs can be written.

A.4 Processing the Input

This section describes in detail what mutex does when it reads commands and characters from input. First, mutex does preliminary work as described in section A.3.3, then starts reading the main input source file. The first line is

```
# Main table of contents for the textbook:
```

this line is ignored because the # means that it is a comment. The next line is

```
\contents{karst}
```

This tells mutex to take its next input from the contents file in the directory called "`karst`." Mutex opens that file for input and reads

```
\chapter{Karst Geologic Features}
```

This causes mutex to :

- add the name "`Karst Geologic Features`" to the main table of contents
- open a new table of contents file for the chapter and print a heading in it
- open a new index file for the chapter
- open a file to hold the body of the text for this chapter
- increment the chapter counter from zero to one.
- sets the section counter to zero.

The next line

```
\section{Rock Type}
```

increments the section counter to one, sets the subsection counter to zero, adds the line

```
1.1 - Rock Type
```

to the chapter table of contents, and adds the key "`1.1`" to the look-up file. The next line

```
\subsection{Limestone}
```

increments the subsection counter to one, sets the subsubsection counter to zero, adds the line

```
1.1.1 - Limestone
```

to the chapter table of contents, and adds the keys "`1.1.1`" and "`Limestone`" to the look-up file. The next line

```
\item{l_stone}
```

tells mutex that a page of text and images belong here. Mutex opens the text file for reading and copies it, adding the section number, 1.1.1, to the file holding the body of the text for this chapter. It then opens the image file and reads

```
\image {Limestone drawing}
```

This line tells mutex to expect an image. The next line is

```
\digital {limest1.iff}
```

This says that there is a digital image stored in a file called "`limest1.iff`" but mutex does nothing, since Muse cannot display digital files. The next two lines

```
\draw {2591}
\optical {}
```

provide video frame numbers for the images. Mutex currently is looking for DRAWdisc images, so it links frame number 2591 to this page, and it ignores the missing frame number for the optical disc. That is the end of the image file, so it reads the next line of the chapter contents file:

```
\subsection{Dolomite}
```

It processes this line the same way as the `\subsection{limestone}` command, adding a line to the chapter table of contents and keys to the lookup file. The command

```
\item{dolom}
```

tells mutex to read another text file and image file. It processes the files as above. When mutex has reached the end of the karst contents file, it returns to the main input source file and reads

```
\contents{glacial}
```

This says to repeat the whole process for the contents file in the "`glacial`" directory. When mutex has finished reading the main input source file, the textbook input is complete. It finishes the few remaining tasks described in section A.3.3, and the textbook is ready to run.

A.5 Possible Expansion

Not only does the mutex system serve the current needs well, but it has been designed to be expandable. This section discusses some of the possible expansions that would benefit the system.

A.5.1 Text Processing

As noted, text must be manually formatted by the creator of the textbook. Unfortunately, there are currently no text-formatting capabilities in mutex. This limitation can be an inconvenience or a serious problem, depending upon what changes need to be made. Certain letters are wider than others when displayed by Muse, so no set number of characters can be deemed a line length; to check changed text, one must recompile the textbook and view it using Muse. Both of these are relatively minor inconveniences.

On the other hand, if one wishes to make more extensive changes to the format, such as using a different text font or changing the size of the text window, the entire text in hundreds of files might need to be manually reformatted. Additional problems arise if one wishes to implement new features integrated with the text itself, such as indexing of words in the text, or referencing other sections of the text. Therefore, there are practical reasons to develop and include a text-formatting function, and indeed, a skeletal version exists now.

This version simply adds the appropriate section number to each item but does no reformatting of the text. In order to implement reformatting, all that is needed is to expand this simple function to change how text is broken across lines. Although this requires writing a rather arduous routine to find the width of text characters and the width of the text window, this routine can fit very neatly into the structure of mutex.

If text-formatting was implemented it would be possible to embed commands in the text, allowing image and text information to be combined into one file, and facilitating the compilation and editing of the textbook materials. A caption for the image may be included with a `\caption` command, and a reference could be noted with a `\cite` command.

A.5.2 The Input Stream

It is useful to expound here upon the concept of an "input stream." When a program is taking input from a source, such as an open file, it is said to be reading an input stream. One can think of input streams as the flow of many characters, one after another, into a module. As a program reads a stream, it expects to encounter a certain format of input, such as formatted text characters from the text files, or sectioning and item commands from the chapter contents files. The current structure requires that there be at least two open files, called input streams, namely a chapter contents file or main input source file (the left side of figure A-2), and a text or image file (the right side of figure A-2). There is, however, no inherent reason that these must exist.

The requirement that distinct input streams exist was a fundamental to the design of a previous textbook program which used two separate data files for each item and a program in which the structure of the book was embedded [Gray, 1987]. The design of the mutex system still uses the format of some of these old data files. and, as mentioned earlier, a shortcoming of mutex is that these distinct input streams still exist.

However, the overall design of mutex is based on one input stream. This does not mean that the input needs to come from only one file, because a basic feature of the design is the ability for this stream to interrupt itself causing input to be taken from another stream.

The advantages of a design based on a single input stream are significant. For the user of the mutex system (the creator of the textbook), placing information into the proper order is much more logical and convenient; it is no longer necessary to think of placing information in a designated place and then referring to it from somewhere else. Thus, one can concentrate on the content of the text rather than its construction.

Furthermore, mutex processes a variety of commands in a straightforward manner. Many different functions can be provided, and they all exist on the same level. This makes the code very simple to understand, and allows expansions without the need to modify the existing structure. In order to implement any new function, it can be defined in terms of an input command, which is placed in the input stream, and a corresponding mutex program function. Although the current version of mutex does not fully implement this single stream design because text and images must be read from separate files, this would be done if a text-formatting command were implemented.

Appendix B

Images

B.1 Content of Images

Images in the textbook are either drawings or photographs. Drawings are used to show schematic representations of geologic processes and features. Full color original designs are being created by a professional artist using an Apple MacIntosh II computer and Aldus Freehand desktop publishing software. Slides are used to supplement the drawings, making the textbook more informative and more attractive to the user.

Although not every text description has associated images, it is an objective of the system to display visual information wherever appropriate.

B.2 Storage Media

Images must be displayed on the screen as discreet points of light called picture elements, or "pixels." The computer screen and supporting electronics require that these pixels be represented as three integers between 0 and 255, representing the intensity of the three components of light, red, green and blue. The two methods available for the storage and retrieval of these images on Athena video workstations are video and digital.

B.2.1 Digital Images

In digital format, the images are stored as data in regular Unix files, with each pixel represented by the three integers mentioned above. To display these images requires only a color monitor, but the process for displaying digital images has significant disadvantages.

Reading information from a data file is slow because the files are quite large and must be transmitted over the Athena network, so display of an image takes several seconds. These large files also require a large amount of expensive disc storage space. Furthermore, digital images, because they represent pixels with set sizes, cannot be resized.

B.2.2 Video Images

In order to display video images, workstations need to be equipped with a special computer circuit board. This circuit board, converts a video signal, in this case, a standard U.S. television signal, NTSC, into pixels. NTSC, an analog signal, is not generally considered high quality, and cannot display some bright colors. The circuit board is capable of doing real-time conversion of images if the displayed size is the same as NTSC, 480 by 640 pixels. Images can be displayed at any other scale, and this takes less than one second. Video images are stored economically on video-discs, of which there are two types, DRAWdisc and VideoDisc.

For development, a write-once disc, called a Panasonic DRAWdisc, can be used to store and retrieve images. This enables developers to collect and modify a set of images as the rest of the package is developed. However, these discs cannot be produced economically in quantity, and a limited number of players are available. Once the set of images for a package has been completed, the images can then be transferred to industry standard 12-inch VideoDiscs, which are essentially manufactured reproductions of the DRAWdisc and cannot be modified.

B.2.3 The Choice of Video

For this system, video image display was chosen over digital. Video images are less expensive to store, quicker to process, and more flexible in terms of the image display. It is also the only medium currently supported by Muse.

The quality of NTSC is not a serious concern. Indeed, video image quality is usually as good as or better than digital images, especially when bright colors are avoided. Furthermore, before drawings are stored in NTSC on video discs, they are created using Aldus Freehand, a graphics package which defines the elements mathematically, and has essentially no limit on the resolution produced. When a higher quality image format is adopted as a standard, which may be many years away, these original drawings can be used to generate new high resolution images. Similarly, the slides used for the photographic images can also be transferred to any new high resolution medium or format.

Bibliography

[1] Avril, Catherine, Project Athena Information Officer, Personnal communication, April 28, 1989.

[2] Einstein, Herbert H., "The Engineering Geology Educator," Memo, 1988.

[3] Gray, Donald P., MIT Thesis, 1987.

[4] Hodges, Matthew, et al., "A Mutlimedia Construction Set for Educational Applications," IEEE Software Journal, January, 1989.

[5] Kernighan, Brian W., and Pike, Rob, *The UNIX Programming Environment,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.

[6] Kernighan, Brian W., and Ritchie, Dennis M., *The C Programming Language,* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978.

[7] Lamport, Leslie, LaTeX, *A Document Preparation System,* Addison-Wesley, Reading, Massachusettes, 1986.

[8] Noack, Carlos, "Program and Expert System Development for Tutorial Applications in Engineering Geology Development," MIT Thesis, 1988.