A Multimedia Presentation System For Interactive Learning

Mai Lan Nguyen

A Major Report
In
The Department
Of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science
Concordia University
Montreal, Quebec, Canada

December 1997

0-612-39990-7

Abstract

A Multimedia Presentation System for Interactive Learning

Mai Lan Nguyen

With the advances in multimedia technology, computers now support learning in many ways. Further, the Internet may make it possible to break the traditional classroom contact model. The major goal of this project work is to develop a prototype Multimedia Presentation System for Interactive Learning based on a graph model. This model is called the CONCEPT GRAPH MODEL (CGM). The CGM is a digraph in which the nodes correspond to "concepts" to be taught (learned) and a directed arc corresponds to the suggested precedence order. The CGM may be structured hierarchically. The navigation of Concept Graph Model is adjusted to suit the level of understanding of the students. While navigating the CGM, a student can listen to the lecture, see the professor's teaching on the screen, request for a quiz and receive quiz answers interactively. For further detailed discussions, the student may communicate to the professor by using the built-in E-mail sub-system. The student's progress and status can be monitored by the proposed system during the course.

The proposed system is developed using Visual Basic 5.0 with Microsoft Access as the database. Module # 2 (prepared by Dr. Radhakrishnan for teaching Assembly language) is used as a sample topic to demonstrate this system.

Acknowledgments

The many months that it took to complete this project work were very rough with my attention divided four ways among my work, project, family, and my new born daughter. I would like to give my special thanks to my great supervisor and professor, Dr. T. Radhakrishnan for giving me full support and good advises during the preparation and writing of this work. I am very glad to have him as my supervisor for my graduate studies. His dedicated work and his care for students are never forgotten.

I would like to thank Vitaly Iourtchenko, who helped me with the video capturing and sharing multimedia data required. The financial support provided towards the Multimedia Research by the IOR grant (NSERC and NORTEL) awarded to Dr. T. Radhakrishnan is gratefully acknowledged.

Finally, I would like to thank my family, especially my mother, my husband and my little daughter for being so patient, caring, and supportive during my studies and the completion of this project work.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Multimedia

In the olden days, computers displayed everything as text. All the information shown to users was in the form of text on a monochrome monitor. This was adequate for certain things, such as word and data processing, however, this media does not appeal to the majority of the public. In order for today's applications to have an improved look and feel, the computer industry has searched for better media or combinations of media. First, color graphics were introduced, then sound animations followed and finally movies appeared. This created the new term "multimedia" as we know it today.

As defined by the Center of Excellence for Education at Indiana University (CEE): *Multimedia is any combination of text, graphic art, sound, animation, and video delivered to us by computer or other electronic means* [CEEP 95].

With the multimedia approach, one does things differently. Instead of using words to describe an object, one uses diagrams, photos, audio, images or video in an effective manner. Particularly, with the wide spread use of Internet, multimedia will be used in every field from commercial to educational.

As a way of learning, when you watch a recorded program from a VCR or a program on a television, you are in fact, watching multimedia. This multimedia learning approach is comprised of moving graphics and sound. You are simply an observer and you have almost no control on the training program. This is a form of *passive multimedia* [Gert 95]. In passive multimedia, you have minimal control of the flow of the presentation. You can only control forward or backward movement of the flow but it will always follow the same path. On the other hand, *interactive multimedia* is different in the sense that the user has better control on the activities of the presentation; the flow of the presentation does not always follow the same path. The participation from the observer could represent itself as different actions; posing questions or requesting more detailed explanations.

The use of computer-controlled media is referred to as *interactive* because it enables a whole new level of user interaction, and control. Users can navigate through a multimedia interactive presentation at their own speed. There is the opportunity for the instructor to tailor materials to individual's needs. Students can review, assess, get feedback when necessary, and follow up in more detail on selected items. As a consequence, students are more likely to understand the complex interplay of the multitude of events.

1.2 Multimedia presentation

Multimedia presentation means giving a presentation using multimedia. This refers to a wide range of techniques and capabilities. With a traditional slide presentation, text and graphics are used along with sound. This indeed could be considered as a multimedia presentation. However, the users can not branch off from a predefined slide sequence during the presentation. To make a presentation even more interesting, animation or movies may be inserted and interactive controls introduced. When users get more control, they can navigate through a multimedia presentation at their own speed.

Producers created multimedia presentations as far back as the early 1980s. The main difference is that in those days it took months, and cost hundreds of thousands of dollars to produce a presentation which today can be done in half the time and at a fraction of the cost [Hols 94]. Audio and video media take huge amounts of storage and require fast CPUs for handling. As current technologies continue to improve by way of increasing speed and storage and decreasing size and cost in the market, multimedia presentation is becoming more common. We can integrate text, graphic art, sound, animation, and digital video into one hardware and software package. The wide availability of multimedia information creates the demand for communication capability, presentation and sharing of information over networks such as the Internet.

It has proved convenient to divide multimedia applications into four classes according to their requirements. These are:

- Multimedia DATABASE applications
- PUBLISHING applications
- CAL (computer-aided learning)
- GENERAL multimedia information services

The traditional DATABASE applications involve large collections of numeric or text data. They require a range of search and retrieval techniques. Users submit a search request using text as the search key. The search engine searches the database for retrieving the best matched documents.

PUBLISHING applications require a range of media types, hyperlinking, and the capability to access the same data using different access paradigms (search, browse, hierarchical, links). Authentication of information and charging facilities are required. Many scientific publishers have plans for electronic publishing of existing academic journals and conference proceedings, either on physical media or on the network [Adie

96]. Some publishers view CD-ROM as an interim step to the ultimate goal of making journals available on-line on the Internet.

CAL applications require sophisticated presentation and interaction capabilities, of the type found in existing multimedia authoring tools. Authentication and monitoring facilities are required. This form of multimedia is currently used in self-training software. However, each software has its own approach of delivering the material to the audience. Our project is concerned with one such category.

GENERAL multimedia information services include on-line documentation, campus-wide information systems, and other systems, which don't conveniently fall into the preceding categories. For example, online documentation - manuals and instruction books often rely on pictorial information and are enhanced by some sound and video effects.

1.3 Computer Assisted Learning

Today, universities and academic institutions have been put under a financial crunch. This creates a need to support too many students with too few resources. The student's progress and faculty instructional contributions are measured by contact-hours. It's impossible to be every where physically, to meet each and every need, and to answer individual questions. It takes a lot of time giving verbal instructions over the telephone and/or in class to individual students. The time-consuming learning process can prove exceptionally frustrating when the student can not understand a term or articulate a problem. One solution is to hire more staff and create new and/or different teaching materials. This is a universal problem, which can be effectively addressed by pooling resources of people and information together through the use of multimedia technology.

With the advances of digital technology, computers are capable of supporting professors in many ways. New digital technologies such as *interactive multimedia* and *Internet* may make it possible to break the traditional classroom contact model. There is an opportunity for the instructor to tailor materials to individual students' needs and reuse the materials several times, and thus save time for development of more difficult subjects. The students understand the material better when it is presented in multimedia formats than through lectures and readings alone. So, it is useful for generating multimedia presentation for selected topics that would benefit a large body of students. The computer can give students diagrams at several different abstractions and with different degrees of complexity or perspective. Self-paced and independent learning help students to develop their understanding. Students can concentrate on parts they don't understand and move quickly through the parts they do. Students are able to repeat more difficult material, or review ideas when needed. Thus, they do not miss important points like they might miss in the lectures. People at different levels of knowledge can use the same single package. This makes the Computer Assisted Learning or the Computer Aided Learning to be even more important today than in the past.

A recent study showed that CAL received high marks in terms of appeal by the learners. It is difficult to measure effectiveness and efficiency but it does appear that CAL has high potential for on-site learning [HeGr 93]. We can design CAL modules to take advantage of the experiences of many teachers. CAL developed in this way could be better than the efforts of a single teacher.

Computer Assisted Learning could take different forms. It can be supported by a stand alone software system where a student has access to it from a *CD-ROM drive*. The software should make it easy to customize lectures and test so as to satisfy the needs of a student. To overcome the lack of network communication, the software could be built on a *network drive (LAN)* and hence a group of students (perhaps in the same department) can access it at the same time. The software then could take the advantages of grouping them to identify the common needs of students. Questions may be grouped based on the degree of commonality and frequency; thus leading to the construction of FAQ (Frequently Ask Questions). Moreover, the knowledge could be shared through a shared bulletin board. To expose to a bigger group of audience, the software then could be posted throughout the *Intranet* or the *Internet*.

1.4 How computer can assist in learning?

In the use of CAL, one should clearly understand what CAL can do and what it can not do.

Exercising - Reviewing

The computer as a tool is used in management and word processing tasks, laboratory instrumentation, simulation of experiments, information storage and retrieval (databases), programming, course review and testing. A student can choose some lecture from the computer, and review it repeatedly as required for his/her comprehension. Students who only need an overview can skim through certain stages of presentation. Students with some prior knowledge can quickly pass simple stages and move to more detailed levels.

The adaptability of details to individual student's needs is very convenient and effectively helps students to learn with different pace. However, it has its own drawback. It is more difficult to identify and quantify the level of understanding of students to adjust the lecture. Modelling the student as a user is essential but difficult.

Interacting with peers through shared bulletin board

The computer can answer many questions, but not all questions raised by all students. One way to optimize the human resources is to collect *frequently asked questions (FAQ)* and pool these questions into a shared bulletin board. The new questions that have never been encountered before by the computer are forwarded to professors to be answered. A list of students interested in such unanswered questions can be kept track by the computer. When the professor answers a certain question, proactively the answer can be delivered to several students.

Supplementing teacher - Tutoring

Self-assessment by students can be included in CAL in such a way that students can use it when they want to. The best forms of self-assessment not only provide feedback but also provide advice to the student about their misunderstandings, misconceptions, or simple lack of knowledge. Computers have been used extensively in tutorial and drill programs [MiDu 93].

Replacing a teacher as much as possible

Using only the computer and eliminating teachers is a dream, at least for today. Students need an opportunity to ask questions and seek further clarification or alternative explanations of an idea. Personalising a CAL system could be one way to enhance the student's confidence. Intelligent and knowledge based systems can be used for personalising [ISKM 93].

1.5 Project overview

It is advantageous to develop a CAL system based on a solid reasoning model. There are different models that have been used to develop interactive aided learning tools. Each model has its own advantages and disadvantages [Pate 96]. Our project is based on the CONCEPT GRAPH MODEL proposed by Dr. T. Radhakrishnan. The details of this model will be discussed in the following sections.

Our system will be a stand-alone. For each node of the concept graph, there is a file or multiple files with the following types: text, power point, audio, or video to associate with it. The concept graph of a selected module for learning is displayed on the screen and the student can click on any node so that, the corresponding file(s) is displayed. Module # 2 (prepared by Dr. Radhakrishnan for teaching Assembly language) will be used as a sample topic to demonstrate this system. When the video file is played, the student can forward, backward, pause, stop, or resume the video clip. As the user completes navigating a node in the concept graph, it will be shaded to indicate that the node has been visited. Based on the performance of the student at a concept node, the navigation of Concept Graph Model will be adjusted according to the level of understanding of the students. The student can choose the quiz that is associated with the current concept graph node. These quizzes are mainly in True/False or multiple choice questions formats (for now). The system will compute the score for all the answers given by the student and display the result as requested. When possible, it will compare the level of achievement of that student with past students. The student can use the FAQ and the bulletin board for self-testing and send e-mail to professors for more detailed clarification. However, the bulletin board will be available in the network version only. The students can exit at any time and resume at later date/time. The partial state will be preserved in a persistent database and reloaded when the student resumes in the future.

Chapter 2: Computer Aided Learning with Multimedia - Requirements Definition

2.1 Focus on learning ASSEMBLY language

Assembly language is a primitive language and it is hard to learn for many beginning level students. However, knowledge of computer at this level helps them to learn the principles of computer organization in a "look-and-feel" manner. In the assembly language course, one learns about CPU registers, primitive hardware operations in the form of machine instructions, and primitive data types like integers, character strings, and Boolean constants. The three control flow concepts (sequential, conditional, and iterative executions) are easy to map to the hardware level features (instruction pointer, flag register, conditional jump instructions and loop instruction).

At the first year B. Comp. Sc. level, at Concordia University we have well over 200 students. They are taught by 7 or more instructors throughout the year. Many of these instructors are working on a part-time basis. In our context, CAL can also contribute to some degree of consistency in teaching these multi-section courses.

In this project, we focus on the Assembly language portion of the first year course. The contents are divided into eight modules. Each module is divided into several sections. One section corresponds to one node in the concept graph model, and every module has an associated concept graph as conceived by the authors. For each module, we have the following entities:

- Text material (including figures, tables, algorithms, Assembly language programs).
- Self-examination questions (True/False type, multiple choice type, ...).
- A set of slides prepared under power point.
- An audio file to be used in conjunction with the slides.
- Optionally, some modules may have algorithm animation or video clippings.

2.2 How to assist in the learning?

The proposed software system for CAL must be able to deliver the material to the audience in an efficient and effective way. In our case, the Assembly Language lectures are the materials that need to be delivered to the students. The following modes of assistance are identified to assist the students in their learning.

Textbook

It is closely tied to the lecture and the textbook is divided into small modules and sections within a module. In our project, the text files are small text files that are abstracted by the professor to illustrate the material of the course. Each text file contains the lectures relevant to current concept graph node. This text file will be displayed on a screen during the time the video is played.

Video

Videos are recorded as "avi" files to enrich the text presentation. Video player will play the video clip until it finishes. It can be controlled using VCR like functions (rewind, play, stop, forward).

Slides

Slides to be represented by the teacher are scanned into bit map files to enhance the presentation. Since slides prepared under power point can be considered a visual tool, the slide player will display the slide on the space occupied by the video player. The slides may have to be synchronized with the corresponding audio presentation.

Short quizzes for self-evaluation

Short quizzes are inputted as text files and the answers are multiple choice selection or True/False to ease the process of evaluation. The teacher's answers are stored in an answer file for comparison with the answers entered by the students.

Level of student understanding

Based on the performance of the students, a level of understanding is determined. The navigation of concept graph (whether to go one level deeper or not) is adjusted according to the level of understanding of the students. As the level of nodes are changed and so the quizzes would be adjusted as well.

2.3 Teacher's control

The teacher will reply to e-mail questions from students and post some of them in bulletin board to benefit other students. Teachers will send e-mail replies to students who place requests for more detailed explanation or for additional explanation of advanced topics.

2.4 Student's control or Navigations

A student normally navigates through a concept graph visiting several nodes as many times as are required. The precedence relationship recommended between concepts (or sections of a module) is encoded in the concept graph. While reviewing the questions provided at the end of the module, the student may find out some parts of the module that he/she may have to repeat for better understanding. Then, direct access to a section of the current module is permitted. The various modules of the course, eight modules in our case, have linear relationships, or may be controlled by another concept graph at a level higher than the modules. Thus, the concept graph model is applicable recursively.

2.5 Synchronization

The problem of synchronization between multiple media is important. Related information of different media types must be synchronized for display. For example, while displaying the video for a specific topic, the text related to that topic must be displayed at the same time. Commercial multimedia authoring packages provide many different ways of presenting, synchronizing and interacting with media elements. Some of these techniques are summarized below [Adie 93].

Backdrops

An application may present all its visual information against a single background bitmap. A CAL application might use a background image of an open textbook, with graphics, text and video data all presented on the open pages of the book.

Buttons

A *button* can be defined as an explicitly delimited area of the display, within which a mouse click will cause an action to occur. Typically, the action will be (or can be modeled as) a hyperlink traversal. Applications use different styles of button: some may use *tabs* as in a notebook, or perhaps *bookmarks* in conjunction with the open textbook backdrop mentioned above. Others may use plain buttons in a style conforming to the conventions of the host platform, or may simply highlight a word or phrase in a text display to indicate it is active. In our model, icons are used to represent the nodes that students can click to traverse.

Synchronization in space

When two or more nodes are presented together. The author may wish to specify that they be presented in a spatially related way.

- This may involve x/y synchronization. For example, a video node being displayed immediately above its text caption.
- It may involve contextual synchronization. For example, an image being displayed in a specific location within a text node.
- It may involve z-axis synchronization as well. For instance a text node containing a simple title being displayed on top of an image, with the text background being transparent so that the image shows through.

Synchronization in time

It is also important to have data be synchronized in time as well; the obvious case being audio and video tracks (where these are held separately). Other examples are the synchronization of an automatically scrolling text panel to a video clip (for subtitling); or to an audio clip (e.g., a translation); or synchronizing an animation to an explanatory audio track.

2.6 Traditional learning vs. CAL

At the undergraduate level, the main elements of the traditional teaching methods are the professor's lectures, slides, hand-outs and what is written on black boards. Students do assignments, special projects such as research papers, internships, or field projects.

In an ideal situation, how do each of these elements contribute to learning? In the presentation, the authors of the CAL lessons will analyze each of the elements of the traditional model in light of their contribution to learning. One can divide the contributions to learning into several pedagogical strategies that include student control; interactivity; motivation; meaning of content; knowledge-dependent learning; knowledge constructionism; situational learning; and transfer of content value.

The traditional model has a number of important strengths. Ericksen states that "by precept and example, good teachers give voice to knowledge and beliefs linking the past to the present and to the future." [HeGr 93]. On the other hand, there are certain disadvantages inherent in each of the elements of the traditional learning. For example, the lecture provides very limited student interactivity or learner control. Both of these learning strategies appear important for effective learning.

Similarly, the CAL model has its own important strengths and weaknesses. Even an intelligent interactive software can not replace a good teacher. However, from time to time, from certain points of view, CAL will be preferred over the traditional model.

2.7 The main issues in CAL

- Multimedia network is essential. It refers to sending digital video, animation and multimedia data over network. This allows students to take courses remotely. Especially for the continuing education students who can work during the day and study from home at night.

- CAL applications require sophisticated presentation and synchronisation capabilities. Authentication and monitoring facilities are required. Ability to identify and authenticate the students using the material, to monitor their progress, and to supply on-line assessment exercises for the student to complete.

- Creating multimedia titles for various platforms is one of the biggest challenges. While different platforms are currently incompatible with each other, development tool, therefore, needs to be carefully selected. For example, UNIX platform and Windows are two popular systems.

- Software needs to be distributed and used effectively, efficiently and economically. Courses should be given to right students who have completed prerequisite courses. Otherwise, the students will get lost.

- User interface needs to be flexible and intelligent. It must handle various forms of information. The interface should direct a user's cognitive processing toward learning the content and away from the details of using the system.

- HTTP (HyperText Transfer Protocol) and HTML (HyperText Markup Language) need to be extended in a backward-compatible way to add multimedia facilities [Adie 96]. In our project, the HTML is not used to link topics to topics. The implementation of HTML should be done in a bigger project because of the complexity involved.

- Which topics are most suitable for CAL? To reduce the problem of staff, the university is eager to use CAL as much as possible. Theoretically, any existing course could be replaced by CAL. However, they are not all equally suitable. Courses to be used should be carefully examined by a committee to maximise the benefit to the university and more useful to the students.

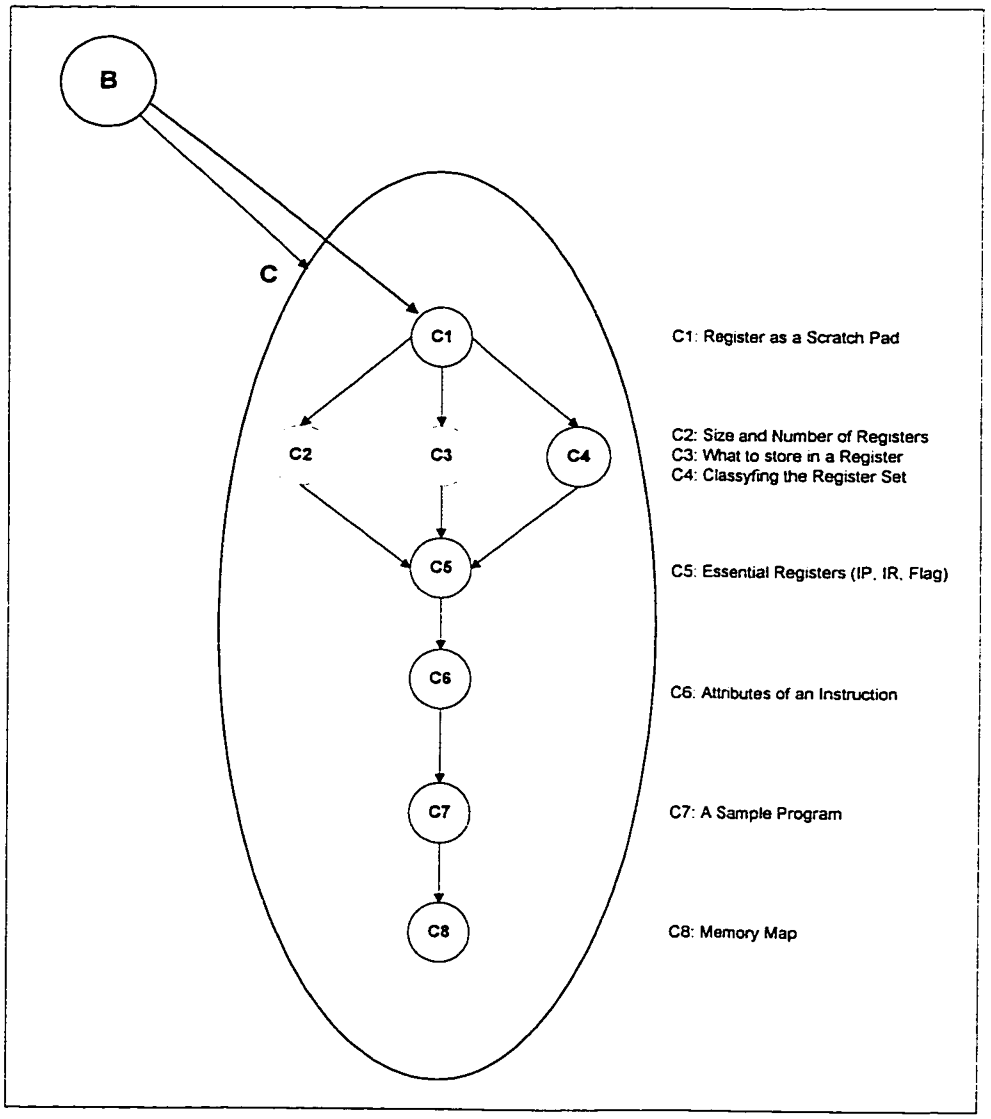
- How can we design a good learning system? To have a good leaning system, a lot of efforts has to be put in. A good system has to be co-ordinated by administrations, teachers and students. The continual change in the system of this sort needs careful design, development, and delivery repeated cyclically often.

Chapter 3: The Concept Graph Model

3.1 Introduction to CGM

Different approaches have been followed by different researchers to provide interactivity in computer aided learning. The work reported herein is "model driven" and a graph based model named *concept graph model (CGM)* is developed for this purpose. Using this model, an author who is an expert teacher of the given subject develops a course consisting of modules. They are stored at a server's site, in the case of networked client-server computing platform, or locally at a workstation for a stand-alone system. Design and development of an interactive software system for the presentation of such modules is the main scope of the work reported in this Major Report. The precedence relationship among the modules would be stored appropriately. A module could be viewed as a *complex multimedia* object that may contain video lectures, oral presentations along with a set of slides, animations of certain concepts or algorithms, a written text perhaps in the form of a supplemental textbook, or any other media object considered suitable by the teacher.

Dividing a course into a set of well-connected modules is an intellectual activity to be performed by the teacher who is also an author in our case. Developing the CGM model for each of the modules also needs the author's intelligence, creativity, and organizational skills. We assume that transforming a module into a lesson that can be interactively presented to learners could be done in more than one way. This manifestation of a module in the form of a lesson can be made appropriate to meet the trade-off necessary in the preparation of materials for computer aided learning. As the experience of the teacher evolves, so will the organization of a course into modules; but one would expect the variations would be more in the initial periods of a course development, whereas they would be relatively less as years pass by. Thus, we believe that well experienced teachers (domain experts) should participate in the generation of modules and the CGM models for a course.



B
C
C1
C2
C3
C4
C5
C6
C7
C8
C1: Register as a Scratch Pad
C2: Size and Number of Registers
C3: What to store in a Register
C4: Classyfing the Register Set
C5: Essential Registers (IP, IR, Flag)
C6: Attributes of an Instruction
C7: A Sample Program
C8: Memory Map

Figure 1: **The Concept Graph Models for modules B and C. Module B has to be completed before going to module C. C is module#2 (A sample program) of the 8 modules in the Assembly Language course. There are 8 concepts in module C.**

The concept graph model is a *directed cyclic graph (DAG)* in which a node corresponds to a 'chunk' of grouped knowledge, and a directed edge joining node x to node y means, in the author's opinion, x should be presented and learnt before y. What should be the knowledge pertinent to a single node is left unspecified, except for the "approximate constraint" that a single module should be linearly presentable in about 40 to 50 minutes, real time. The only permissible interventions during such time-bounded presentation are the direct responses from a learner to the pre-inserted clarification or test questions, inserted into the module by the author. There are two types of nodes in a CGM: a solid or filled circle denoting an atomic concept that is not further refined in this module (or in this course), and an empty or unfilled circle denoting a concept at level *j* that is further refined at the next level *j+1*. For a given node 'm', the author could possibly provide a first-cut explanation called $D_1(m)$, detailed explanation $D_2(m)$, more detailed explanation $D_3(m)$ ... etc. This is called a *degree of explanation.* Thus, the concept graph can be viewed as a two-dimensional DAG in which vertical axis corresponds to the concepts i, i+1,... etc; and the horizontal axis corresponds to the varying degrees of explanation of a specific concept, say 'j'.

3.2 Modules for Teaching Assembly Language:

In the Computer Science Curriculum at Concordia University, at the first year level, we have three *streams:* (a) computer systems stream, (b) programming stream, (c) mathematical foundations stream. Each stream consists of a sequence of two courses. The first course in the computer systems stream is about computer organization and assembly language. Assembly language programming of the Intel 8086 family is used for hands-on practice in the lab, whereas the course itself is not *fully* devoted to IBM-PC. The course provides generic introduction to computer organization and specific programming exercises with the Intel family of processors. Thus, we do not devote too much time for teaching assembly language programming, but we introduce a small subset of the instructions and the basic organization of the processor and its instruction level view. Experience shows that we devote about 8 to 10 hours towards teaching the assembly language aspects. An experienced professor, who has taught this course several times, has organized the materials for this part of the course into the following 8 modules:

Module 1: BEGINNING

This is an introductory lecture to Assembly language part. The student will get an idea about what is meant by hardware, software of a PC, and the ASSEMBLER's input and output.

Module 2: A SAMPLE PROGRAM

In this module, the student will learn about the structure of registers, some of the registers, how an instruction looks, a sample program as input to the ASSEMBLER and as output by it, and memory map.

Module 3: ASSEMBLING

In this module, the student will learn about OPCODE table, SYMBOL table, and some assembler instructions; student can start to write a simple loop-free program.

Module 4: SEGMENT REGISTERS

In this module, the student will learn about the use of segment register, conditional branch, relative addressing so that he can write a more complicate ASSEMBLER program.

Module 5: INDEXING AND IMMEDIATE ADDRESSING

In this module, the student will learn about array data structure, the use of indexing and immediate addressing, loops instruction, the distinction between compile time and run time.

Module 6: PROGRAM RELOCATION

In this module, the student will learn about the program relocation in memory, the idea of memory stack, and the protection of one program against another program in RAM.

Module 7: MODULARITY

In this module, the student will learn about the modularity. Student will be able to write a modular program using CALL, RETURN, passing parameters.

Module 8: ASSEMBLE, LINK, LOAD

In this module, the student will learn about LOADER, LINKER and how an ASSEMBLER program is executed in more detail.

For this major report, we will be using Module#2 as a sample data. Later, we will describe the lesson created in a multimedia form corresponding to the CGM model of Module#2. In the next section, we describe how a node in CGM is characterized.

3.3 A node in the CGM model:

A node in the CGM model corresponds to a **concept** that is an identifiable unit for discussion and presentation in a classroom or authoring. We view it as a quintuple, consisting of the following details:

D: {d}	D is the natural language description of the concept; d is a text unit.
E: {e}	e = example to describe the concept.
N: {n}	n = an analogy used by the teacher to explain the concept, possibly empty.
P: $\{p_1, p_2, \ldots p_n\}$	probe questions to examine student's level of understanding [p_is are linearly ordered or partial ordered]. A probe question can be answered with

True/False choice, or a multiple choice type question. These are used on-line by the presentation software system to judge how far the learner has understood what is presented.

Q: $\{q_1, q_2, \dots q_n\}$	q_i = test questions that can be used (optionally) by teacher or student for practicing the "skill" taught; they are in the form of short quizzes, practice problems etc.

Each module when completed by the author consists of the following six entities. These are generated from <D,E,N,P,Q> that are used in the planning stage of the authoring.

1. Running text embedded with figures, tables and equations like a conventional book . **(T)**
2. A set of test Question. **(F)**
3. An ordered sequence of power-point slides. **(S)**
4. An audio file that is segmented into partitions with one-to-one correspondence between partitions and slides. **(A)**
5. Keywords and phrases used is introduced in this module. **(K)**
6. A set of zero or more animations, videos corresponding to different algorithms or concepts discussed in the module. **(M)**

A node in DAG can be one of the following types:

➢ Start node	No predecessor.
➢ End node	No successor.
➢ Milestone node	Marked by the author (they could be used by the presentation system to advise learners about their achievement, speed, etc.).
➢ Choice node	Having multiple successors (they could be used to select which concept to learn next).

➢ Solid node	Not further refined in the module.
➢ Empty node	Further refined in the module.

3.4 Concept Graph Organization of Module 2

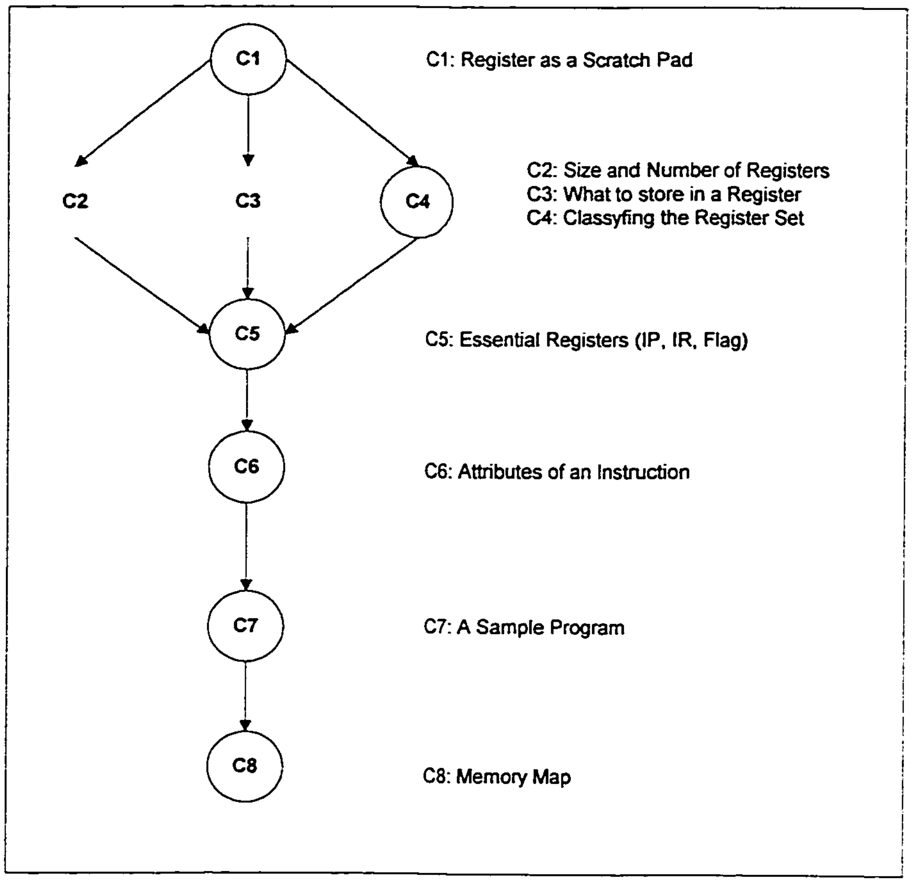

C1
C2
C3
C4
C5
C6
C7
C8
C1: Register as a Scratch Pad
C2: Size and Number of Registers
C3: What to store in a Register
C4: Classyfing the Register Set
C5: Essential Registers (IP, IR, Flag)
C6: Attributes of an Instruction
C7: A Sample Program
C8: Memory Map

Figure 2: **Concept Graph Model of Module 2**

3.4.1. Register as a Scratch Pad (C1)

➢ D:

* Registers are temp storage
* Like scratch pad - quick access
* Accessing a RAM takes 100s of ns; registers are 10 times faster

* Every computer has many registers
* Read from RAM into registers, process and then leave it in registers
* Register to RAM = STORE
* RAM to register = LOAD

➢ E

* IBM S/370 has 16 general purpose registers (GPRs)
* PC has 14 registers partitioned

➢ N

* Short term memory

➢ $\overline{P}$

* Load from address X destroys contents of memory location named X , this is denoted as C(X): T/F?
* Store into address X destroys C(X): T/F?

➢ $\overline{Q}$

* Null

3.4.2. Size and Number of Register (C2)

➢ D:

The size and the number of registers vary depending on the machines.

* Length of the register: 8, 16 , 32 or 64 bit register
* Longer register can store more information; However, if it is not used effectively, it is a waste
* Register is used to store the address of RAM hence RAM size is determined
* Register could also be used to store the data range

➢ E

* 8086 has both 8 & 16 bit registers

* SMAC2 has 32 bit register

➢ N

* Null

➢ $\overline{P}$

* Length of sub-register = Length of register: T/F?
* Disadvantages of long register?
* Disadvantages of short register?
* A 3 bit address register can address --- item (fill up)

➢ $\overline{Q}$

* Null

3.4.3. What to Store in the Register (C3)

➢ D:

* Data, bits, bytes, integers
* Instruction (op_code / zero or more operands)
* Address of operands or destination in RAM or other registers
* Address 2^n registers need n bits

➢ E

* add ax, bx

  if op_code (add) = 0B Hex

  address (ax) = 2 Hex

  address (bx) = 4 Hex

  then code it as 0B24 16-bit long instruction.

  Semantics is C(ax) + C(bx) replaces as C(ax)

➢ N

* Null

➢ $\overline{P}$

* How many bits are needed to address 13 items?
* A bit sequence, in general could be a data or an instruction: T/F?
* Op_code table is fixed for the life of a computer hardware: T/F?

➢ $\overline{Q}$

* Add ax, bx where ax, bx are 8-bit registers, ax = 255, bx = 255
    - What happens due to addition?
    - Explain this phenomenon
    - Explain op_code decoding

3.4.4. Classifying the Register set (C4)

➢ D:

* The register set can be partitioned
* Each partition has a pre-determine role to play
* To partition or not to partition is the question
* Index register for vector operation (SI, DI)
* Segment register for extending addressable range (CS, DS, SS)
* Data register for computation (AX, BX, CX, DX)
* Pointer register for holding base address, for stacks, etc.. (SP, BP)
* General purpose register - Context will determine the role played and encoded in the instruction

Add	b	x	DISP	b	x	DISP

b: base register reference

x: index register

DISP: displacement value

➢ E

* Set of all humans = set of men U set of women

➢ N

* Null

➢ $\overline{P}$

* A ∩ B = where A and B are partitions of a set T/F?
* What prevails having too many addresses with an instruction
  - instruction length becomes too long
  - too hard to program
  - too hard to code

➢ $\overline{Q}$

* Advantages / Disadvantages of partitioning or keeping as GPRs of a register set?

3.4.5. Three Essential Registers (C5)

➢ D:

* Three essential registers in any computer are:
  1. IP: Instruction pointer or program counter - Hold the address of next instruction
  2. IR: Instruction register - Hold the current instruction
  3. FLAG: FLAG register or condition code register - Hold the conditions resulting from operations
* At the end of non-branch instruction IP is incremented to point to the next instruction. At the end of a successful branch, it contains the "branch address" where the next instruction is

➢ E

* Null

➢ N

* Null

➢ $\overline{\text{P}}$

* IP will be incremented by 1 if all instructions are unit length T/F?
* IP always contains an address T/F?
* IR may contain data in some cases T/F?
* FLAG register bits are:
    - set at the end instruction execution
    - reset at the beginning of every execution
    - reset at the beginning of some execution
    - not touch by the CPU but only programmer sets and resets

➢ $\overline{\text{Q}}$

* How does the CPU know the length of an instruction to increment the PC?
* Difference between branch and conditional branch?
* How does the FLAG register help conditional branch instructions

3.4.6. Attributes of an Instruction (C6)

➢ D:

* An instruction has
    1. Op_code
    2. One, two or more operands

        → operand value

        → operand address

➢ E

* Null

- ➢ N
    - * Null
- ➢ $\overline{P}$
    - * Op_code and operand address are two essential parts of an instruction
      T/F?
    - * Op_code table is always sufficient to decode an instruction
      T/F?
- ➢ $\overline{Q}$
    - * Can there be 4 addresses in an instruction?
    - * How is an instruction with zero address work?

3.4.7. A Sample Program (C7)

- ➢ D:
    - * A sample program to read X, Y and then print the lager of the two
      1. Read X
      2. Read Y
      3. Compare
      4. Print the lager of two
- ➢ E
    - * Null
- ➢ N
    - * Null
- ➢ $\overline{P}$
    - * What is/are the basic part(s) of the program?
    - * What is meant by assembler instruction?
- ➢ $\overline{Q}$
    - * Modify the program to find the largest of three numbers

3.4.8. Memory Map (C8)

- D:
  - Contiguous area in RAM
  - Code block
  - Data block
- E
  - Null
- N
  - Null
- $\overline{P}$
  - What happens if program is relocated
- $\overline{Q}$
  - Relocate the program to start at address 1000

Chapter 4: Software Design of the proposed "Presentation System".

4.1 Introduction

The main goal of this project is to illustrate the concepts used in a Multimedia Presentation System for Interactive Learning. Some courses taught over a long period of time with a stable set of concepts could be adopted for computer based interactive learning. Students learning in this manner may use a network like Internet or use a stand-alone system for self-learning. Our project is concerned with the presentation aspects of such courses in a stand-alone mode.

4.2 Objective & Scope

Our objective is to build a Multimedia Presentation system to be used by one or more students. The multimedia course material could reside either in a network (LAN) or on a CD-ROM (stand-alone application). To accomplish this objective, the following tasks are identified:

1. The author records material of the course to be presented. For now, no special support is provided by our system for this purpose.
2. The students access our software to have:
   - Lectures delivered whenever they wish.
   - Monitored by professor for certain aspects (i.e., performance of the student is logged in the database and the professor could monitor student's achievement off-line).

- Student receives answers from professor for questions asked at an earlier time through e-mail.
- System adjusts the level of difficulty in presentation according to a few predetermined categories: level 0, 1, 2 etc.

The scope of this project is limitted to design and build a system to the required specification and give a demo of the prototype system. It should provide online activities, namely a student can listen to the lecture, request for quiz and receive quiz answers interactively. For more detailed interactions, the student will communicate with the professor using the e-mail facility.

Our software should be able to handle text, graphical animation (embedded in slides), audio and videos all in the form of a shared medium. Module # 2 (prepared by Dr. Radhakrishnan for teaching Assembly language) will be used as a sample topic to demonstrate this system.

Based on the performance of the student "on a node", the level of difficulty on the next node is determined and the student will be guided to a pre-determined "next node". If a student does not pass the quiz at the default level "n" then the system will pass the student to a sibling node using simpler examples and conceptually "low level" concepts at level n-1. If the student still does not understand and the system can not identify a lower level sibling node, then the system will lead him to a consultation with the professor. In our sample, the performance of node C1 will determine the next node, the next node could either be C2, C3 or C4 if student passes the C1 quiz (grade > 50). In case the student fails the C1 quiz, then the system will lead him to a consultation with the professor.

4.3 System Requirement

Hardware

User's equipment:

- IBM PC or compatible 486 SX or later
- Sound blaster or compatible
- CD ROM drive (if it is stand alone version)
- Speaker(s)

Developer's equipment:

Same as user requirement plus

- Video capture facility
- Scanner for image capturing

Software

Windows 95/NT

Visual Basic 5.0

Database

Microsoft Access

4.4 Multimedia Presentation System Process Definition

The Multimedia Presentation System is a GUI system. Information exchanges between student and the system is controlled by a series of graphical panels where students are able to login the system, listen to the lecture and perform various tasks. In this section, we present the GUI aspects through a set of screen print-outs.

4.4.1 Welcome Panel - Getting Started

There are a few different ways to start the Multimedia Presentation System. One of the easiest ways is that the student just double click on the icon on the desktop to start the application.

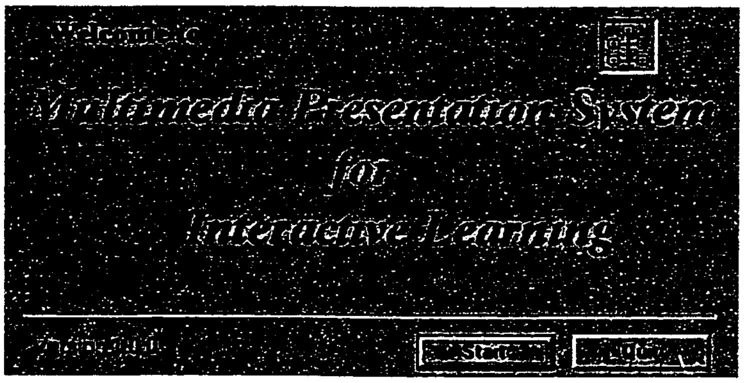
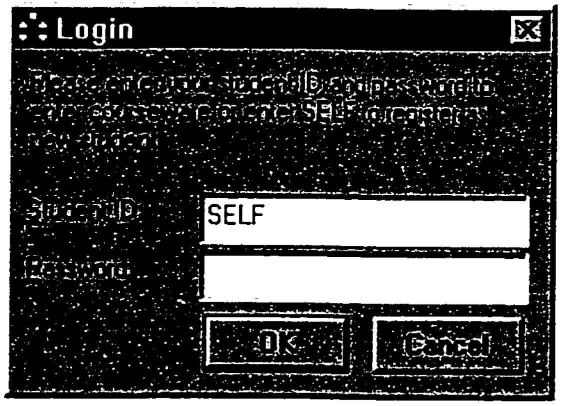


Figure 3: **Welcome Panel**

Upon starting the application, a welcome panel will pop-up. Student, then, is asked to click the command START to start the software or the command QUIT to quit the application.

4.4.2 Login panel - Register to the System

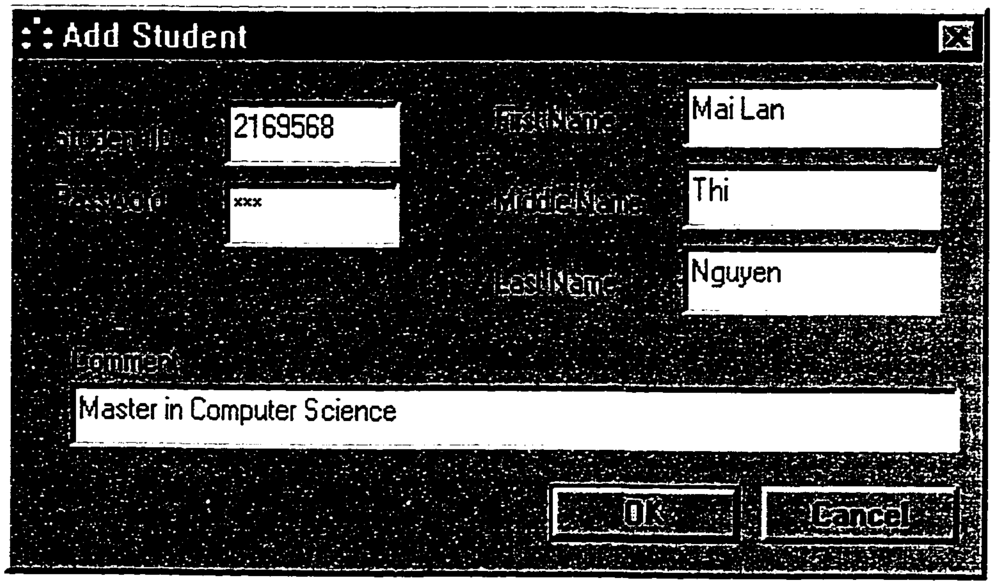


SELF

Figure 4: **Login Panel - Registering student to the system**

For the first time log in, the student has to enter '**SELF**' in student ID dialog box of the **login panel** to initiate the registration panel. This process should be done by the student if he/she is using stand-alone version and it should be done by the University's staff if it's a network version.

By clicking the OK button, the 'Add Student' panel is displayed. Student has to enter the requested information. The information is save in local ID file, which will be used to validate subsequent log in of this student.

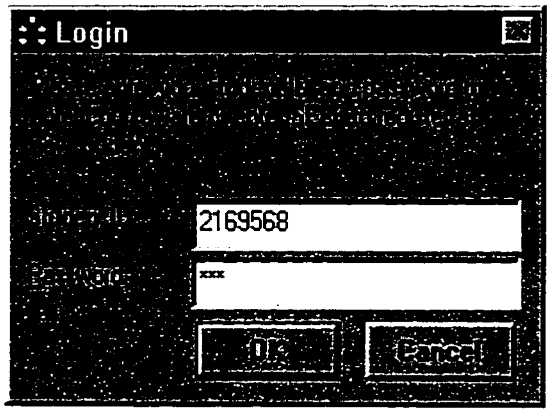

Add Student

StudentID: 2169568
Password: xxx
FirstName: Mai Lan
MiddleName: Thi
LastName: Nguyen
Comment:
Master in Computer Science

Figure 5: **Add new student panel**

4.4.3 Login panel - Sign on the system

The login panel is used to log the student into the multimedia presentation application. The student ID and password will be asked for verification. The password is validated against the ID file, which was created in the first time login.



2169568

xxx

Figure 6: **Login Panel - Sign on the system**

The student has to enter the student ID and the password to get into the system. If either the student ID or the password was incorrect, a dialog box will display requesting the student to try again.

4.4.4. Main Panel - Select a Module

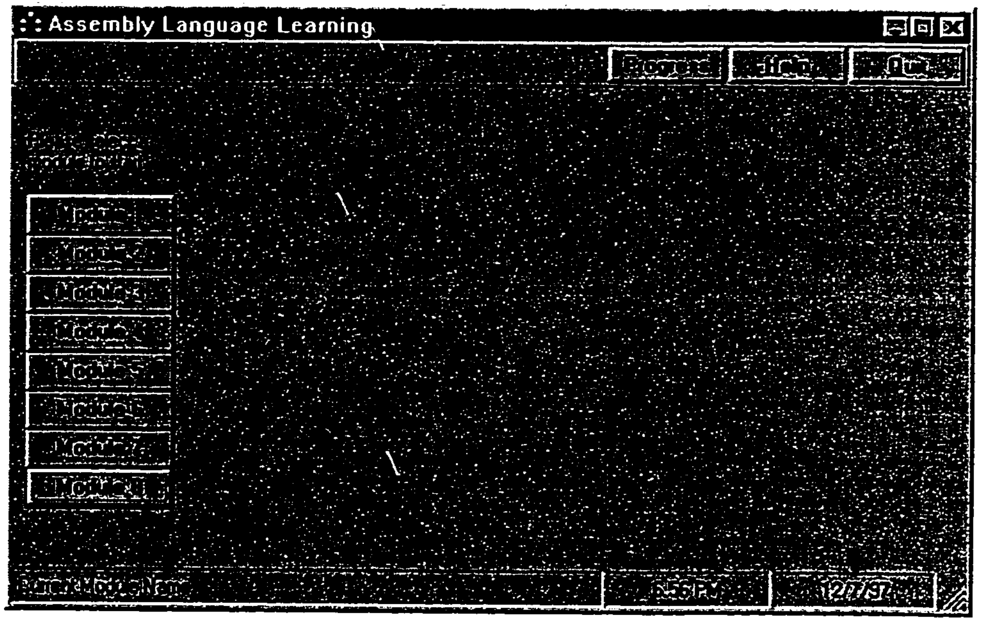


Figure 7: **Main Panel - Select a module**

The Main panel, as the name stands for, is the main form where the student can select a module that he or she is interested on. This panel contains eight command buttons, which represent the eight modules of the course.

The student can click on

- Progress button on the tool bar to view his performance from previous sessions, or
- Help button on the tool bar to initiate the help on using the application, or
- Quit command on the tool bar to quit the application, or
- One of the buttons labeled Module to select the module that he/she would like to learn.

4.4.5 Main Panel in Action

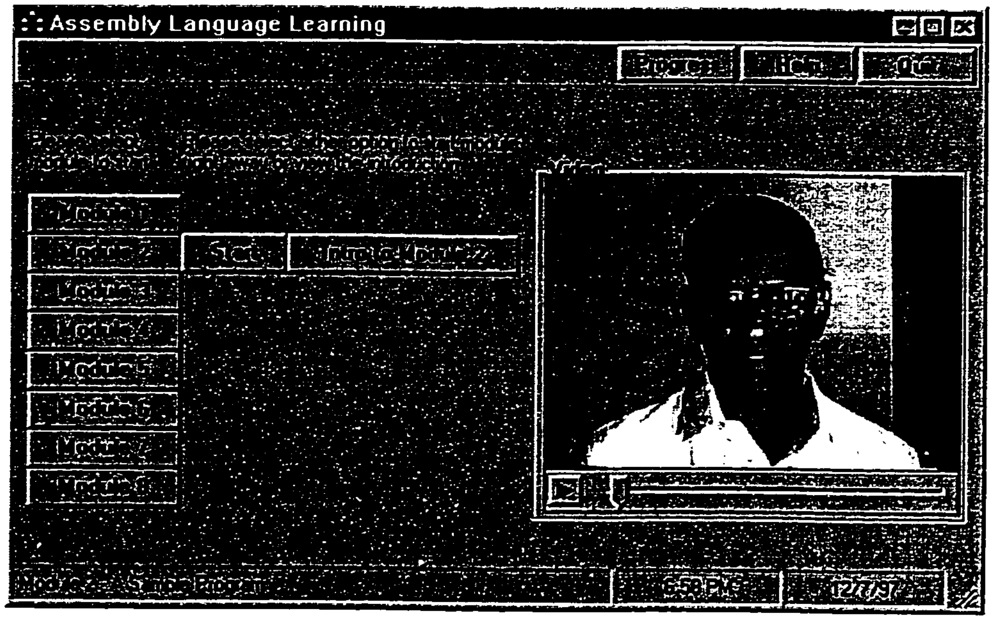


Figure 8: **Mail panel - In action**

Once a module is selected, additional command buttons will be popped up so that the student could either select to start the module right away or view the introduction of the module that he/she is interested on. If the introduction button is clicked, a video clip will be played to introduce the module that the student just selected. While the introduction is playing, the student is allowed to START the module or select another module. Moreover, for convenience, the command under the video window allows the student to

- Pause/resume the movie by clicking the command
- Fast forward/backward the movie by dragging the control on the slider

4.4.6 The Concept Graph Model Panel

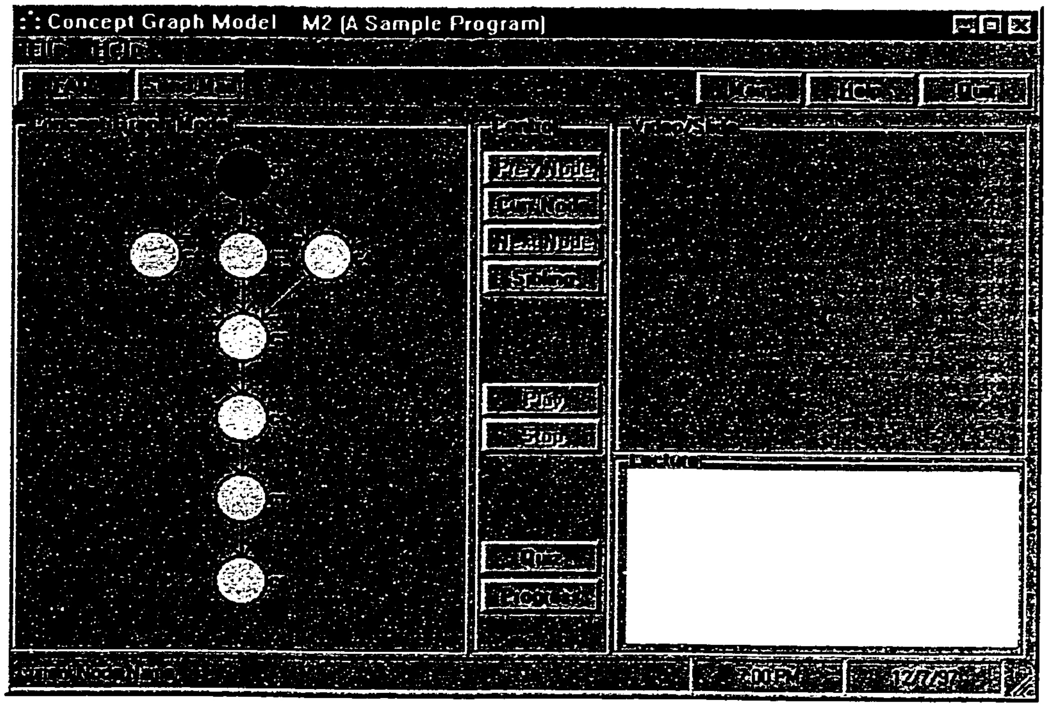


Figure 9: **Concept Graph Model panel**

The concept graph model CGM panel is actually the heart of the application. This panel is composed of seven sections:

Menu bar:

The menu bar is introduced for future expansion. On this release, menu bar only has 2 options

1. File - Exit: Exit the multimedia presentation system.
2. Help-About: Display a brief message about the application.

Tool bar:

1. FAQ button allows student to access to Frequently Asked Questions.
2. Mail button allows student to activate the mail sub-system so that he/she could communicate with the professor or other students.
3. Main button allows student to go back to the main panel.
4. Help button allows student to get help of current application.
5. Quit button allows student to quit the application at any time.

CGM Section:

This section displays the CGM nodes where nodes are linked together. The structure of the node is top-down and color coded where

1. Red means that the nodes have been *completed.*
2. Yellow means that the nodes are not *visited* yet.
3. Green means that node is current *active* node.

Control section:

This section contains the control buttons to the CGM nodes.

1. Prev Node button allows student to play the previous node.
2. Curr Node button allows student to start playing a node.
3. Next Node button allows student to play next node in the concept graph model. However, this command is enabled only if the student has already completed the current node. In another word, the student has to pass the quiz of the current node before he/she can advance to the next level of the course.
4. Sibling button allows student to play the sibling node.
5. Play button resumes the current movie.
6. Stop button stops the current movie.

7. Quiz button requests or quiz of the current node.
8. Progress button requests for student performance status.

Video/Slide Section:

This section displays the movie of the current selected node. It can also be used to display the slides of the current active node.

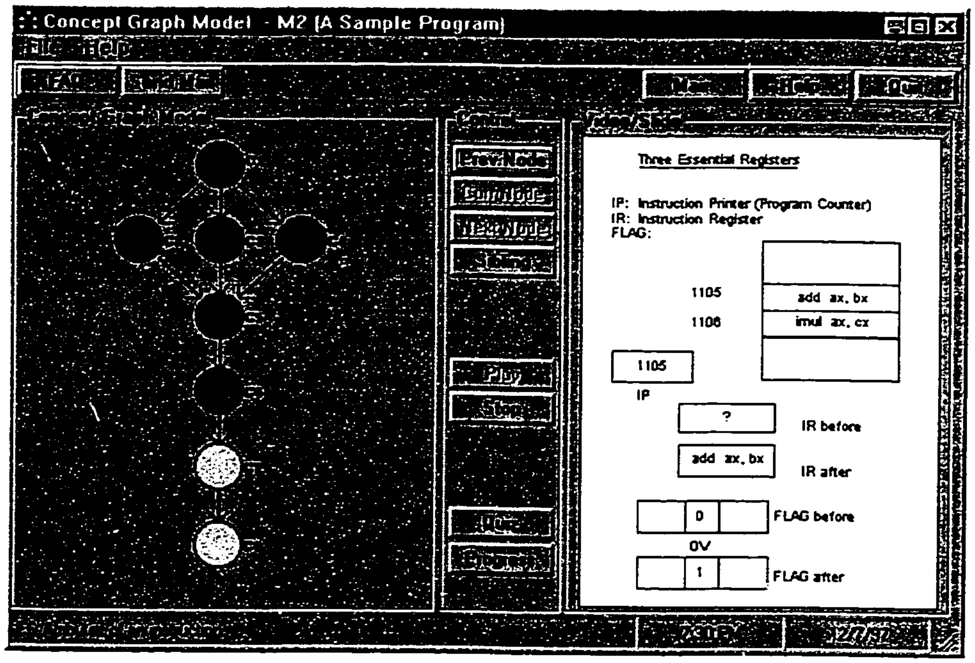
Lecture Section:

This section displays the lecture (including probe questions, examples, analogies...) of the current active node. The lecture is a text file. Usually the lecture text file is displayed in conjunction with a movie.

Status bar

Displays the status of the current node. For the time being, it's used to display the long description of the active node and the current system time, date. In future, its use can be enhanced.

4.4.7 The CGM Panel - Slide Section in Action



Three Essential Registers

IP: Instruction Printer (Program Counter)
IR: Instruction Register
FLAG:

1105	add ax, bx
1106	imul ax, cx

1105

?

add ax, bx

	0	

	1	

Figure 10: **CGM panel - Displaying slides**

The student could click a node on the CGM section. The information pertinent to this node will be played. It could be either an audio presentation in conjunction with the slides or a video presentation in conjunction with lecture.

The audio has the following characteristics:

- The audio is a sound file with the extension of '.wav'.
- No display (it's sound).

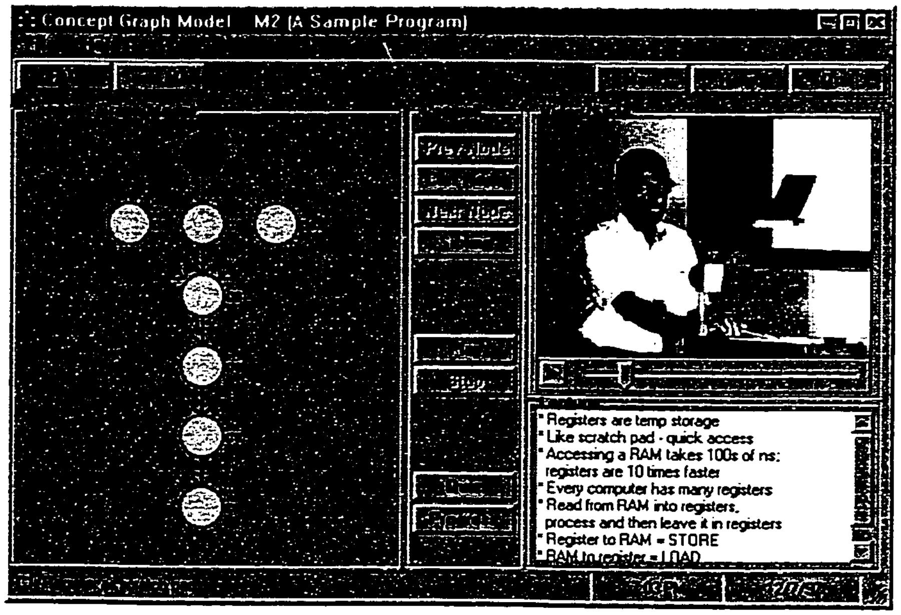
The slides have the following characteristics

- The slide could be either a windows meta file (extension .wmf) or a bit map file (extension .bmp).
- The slide of the current node is displayed on in the Video/Slide extended section.
- The slides are displayed as a slide show where the time intervals between the start time and end time of the given slides are predefined by the professor.

4.4.8 The CGM Panel - Movie Section in action

A node could also be composed of a video clip with other media. Then the clip is constrained as follows:

- The video clip is played and displayed in the video section.
- The size of the clip is fixed.
- The clip is controlled by the control commands in the middle of the screen or the control button positioned underneath the clip.



- Registers are temp storage
- Like scratch pad - quick access
- Accessing a RAM takes 100s of ns: registers are 10 times faster
- Every computer has many registers
- Read from RAM into registers, process and then leave it in registers
- Register to RAM = STORE
- RAM to register = LOAD

Figure 11: **CGM panel – Playing movies**

Every lecture has the following overall characteristics

- The lecture actually is a text file.
- The lecture of the current node is displayed in the lecture frame.
- Scroll bars are used if the lecture text does not fit in the lecture frame.

4.4.9 Quiz Panel – Performance

The Quiz panel is designed to test the student's understanding of the subject described in the node. Normally, most of the nodes have the quizzes to evaluate the level of understanding of the student.

To activate the quiz panel, the student simply clicks the Quiz button. The quiz will be popped up showing the question on the top part of the panel. The proposed answers are in the Answer frame. Radio buttons are used for the student to enter his or her answers.

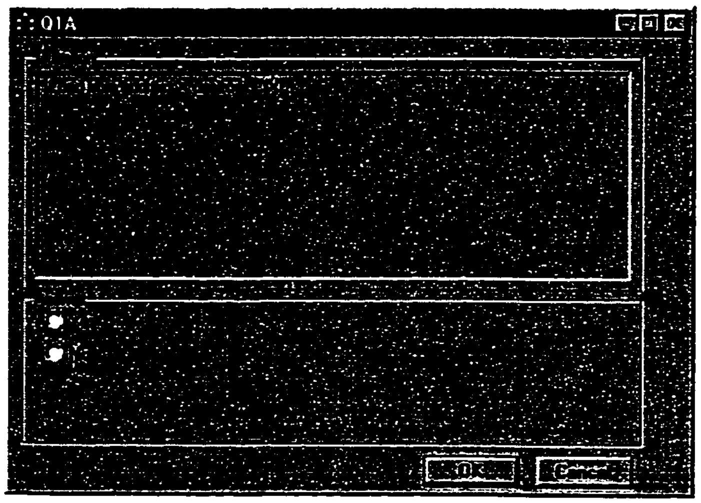
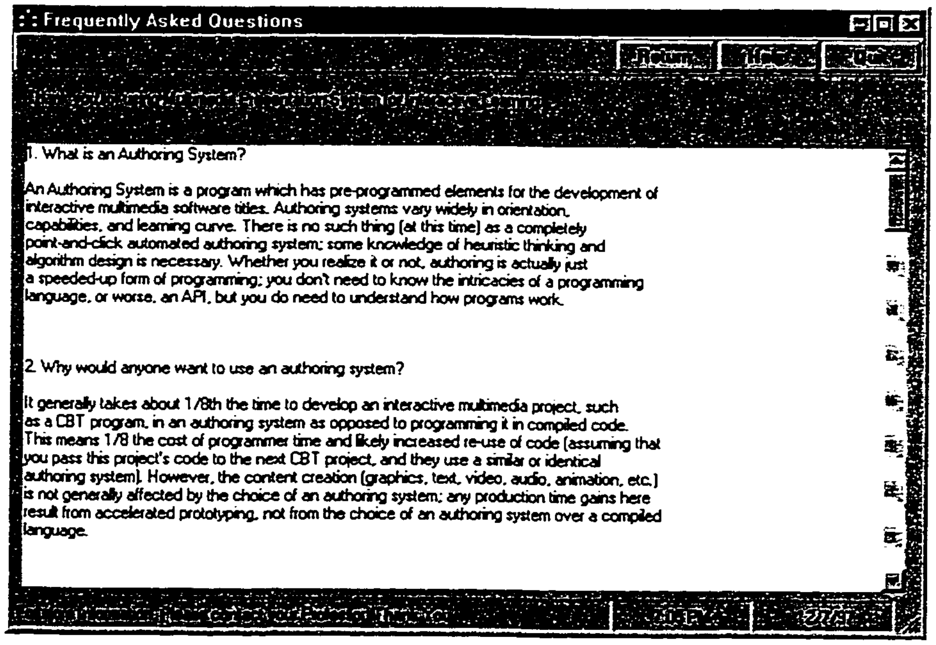


Figure 12: **Quiz panel**

4.4.10 FAQ - Frequently Asked Question

The FAQ panel is nothing more than an electronic board to display the answers to the frequently asked questions. The FAQ is a text file, developed by the professor. This file should be updated by the professor whenever there is need.



1. What is an Authoring System?

An Authoring System is a program which has pre-programmed elements for the development of interactive multimedia software titles. Authoring systems vary widely in orientation, capabilities, and learning curve. There is no such thing (at this time) as a completely point-and-click automated authoring system; some knowledge of heuristic thinking and algorithm design is necessary. Whether you realize it or not, authoring is actually just a speeded-up form of programming; you don't need to know the intricacies of a programming language, or worse, an API, but you do need to understand how programs work.

2. Why would anyone want to use an authoring system?

It generally takes about 1/8th the time to develop an interactive multimedia project, such as a CBT program, in an authoring system as opposed to programming it in compiled code. This means 1/8 the cost of programmer time and likely increased re-use of code (assuming that you pass this project's code to the next CBT project, and they use a similar or identical authoring system). However, the content creation (graphics, text, video, audio, animation, etc.) is not generally affected by the choice of an authoring system; any production time gains here result from accelerated prototyping, not from the choice of an authoring system over a compiled language.

Figure 13: **FAQ Panel**

4.4.11 Mail Panel - Contacting The Professor

The mail panel allows student to log on to the Windows 95 mail system and send E-mails to the professor or to his/her fellow students. Basically, the student has to log into a mail profile that was assigned to him or her. (This option is practical if the workstation is used by more than one student). Once successfully logged on to the mail system, the student will be able to send and receive mails.

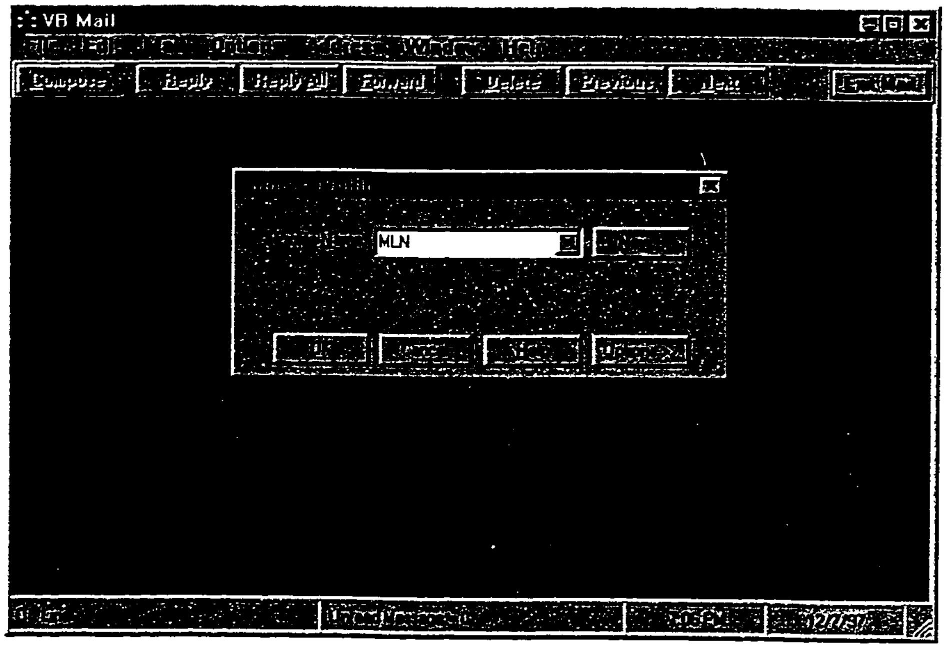


Figure 14: **Mail Panel - Contacting Your Professor**

The mail panel has the following control:

<u>On the Menu bar:</u>

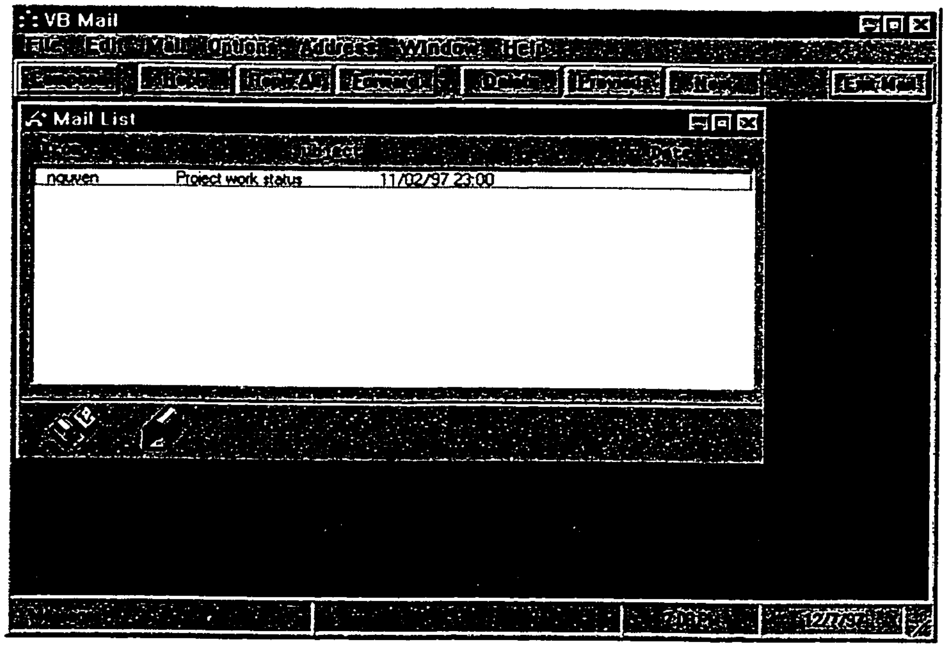
1. File - Print Message: Allows student to print a message.
2. File - Printer Setup: Allows student to change the set up of printer (using Windows printer control).

3. Edit – Delete: Allows student to delete messages.
4. Mail – Log on: Allows student to logon his or her mailbox.
5. Mail – Log off: Allows student to log off from his or her mailbox.
6. Options – Mail: Allows student to change mail references.
7. Windows – Cascade: Change the view of mail system.
8. Windows – Horizontal/Vertical: Change the display of the messages.
9. Help – About: Display help.

Upon successful log-on, the buttons on the tool bar are activated and the student could perform the basic functions of a mail system.

On Tool bar:

1. Compose button: Allows student to compose a new message. The tool for the mail editing is actually the default mail editor of the workstation. For example, if the default mail of the workstation is the Internet mail, then the Internet mail editor will be used to compose a new message.



nguyen Project work status 11/02/97 23:00

Figure 15: **Mail Panel - Composing Messages**

2. Reply or reply all buttons: Allows the student to reply to the authors all the mails that he or she has received.
3. Delete button: Allows the student to delete his or her E-mails in the mailbox.
4. Previous/Next buttons: Allows student to select another message in a linear sequence.

4.5 Database Definition for Multimedia Presentation

In this section we present the various databases along with their definitions as used in our system. Each sub-section describes one table of the database and they are implemented in Microsoft's Access.

4.5.1 CGM_Student_Grade

Description

This table contains all grades which the student has obtained after visiting node(s) of the CGM.

Purpose

We use this table to keep track of student's grades and student's performance level.

Primary keys

1. Curr_Student_Id
2. Curr_Module_Name
3. Curr_Node_Name

Field Name, Data Types and Field Size

1. Curr_Student_Id	Text	10
2. Curr_Module_Name	Text	10

3. Curr_Node_Name	Text	10
4. Curr_Node_Grade	Number	Integer

Example

In this example, student '2169568' has visited nodes 'C1','C3','C5' of module 'M2'. Student '1234567' has visited node 'C1' only.

Curr_ Student_ID	**Curr_ Module_ Name**	**Curr_ Node_ Name**	**Curr_ Node_ Grade**
2169568	M2	C1	80
2169568	M2	C2	0
2169568	M2	C3	90
2169568	M2	C4	0
2169568	M2	C5	100
2169568	M2	C6	0
2169568	M2	C7	0
2169568	M2	C8	0
1234567	M2	C1	100
1234567	M2	C2	0
1234567	M2	C3	0
1234567	M2	C4	0
1234567	M2	C5	0
1234567	M2	C6	0
1234567	M2	C7	0
1234567	M2	C8	0

Table 1: **Student Grade Table**

4.5.2 CGM_Module_Info

Description

This table contains all information regarding a module such as name, description, avi /audio file name.

Purpose

We use this table to find the module's description and the introduction video/audio file associated with it.

Primary keys

1. CGM_Module_Name

Field Name, Data Types and Field Size

1. CGM_Module_Name	Text	10
2. CGM_Module_Short_Desc	Text	10
3. CGM_Module_Long_Desc	Text	40
4. Mod_Avi_FName	Text	10
5. Mod_Wav_FName	Text	10

Example

In this example, we have 8 modules of the Assembly Language Course. Each module has either video/audio introduction file.

CGM_ Module_ **Name**	**CGM_ Module_ Short_ Desc**	**CGM_ Module_ Long_** Desc	**Mod_ Avi_ FName**	**Mod_ Wav_ FName**
M1	M1	Beginning	M1	NA
M2	M2	A Sample Program	NA	M2
M3	M3	Assembling	M3	NA
M4	M4	Segment Registers	NA	M4
M5	M5	Indexing & Immediate	M5	NA
M6	M6	Program Relocation	NA	M6
M7	M7	Modularity	M7	NA
M8	M8	Assembler Link Load	NA	NA

Table 2: **Module Table**

4.5.3 CGM_Node_Info

Description

This table contains all information about a node such as name, position, type, its sibling, description, video + text file name, audio file name, number of questions in the quiz for this node.

Purpose

We use this table to find the CGM node's name, position, type, sibling node, description , video + text file name, audio file name, number of quizes.

Primary keys

1. CGM_Module_Name
2. CGM_Node_Name

Field Name, Data Types and Field Size

1. CGM_Module_Name	Text	10
2. CGM_Node_Name	Text	10
3. CGM_Node_Row	Number	Integer
4. CGM_Node_Col	Number	Integer
5. CGM_Node_Type	Text	3
6. CGM_Sibling_Node	Text	10
7. CGM_Module_Short_Desc	Text	10
8. CGM_Module_Long_Desc	Text	40
9. CGM_Avi_FName	Text	10
10. CGM_Txt_FName	Text	10
11. CGM_Wav_FName	Text	10
12. CGM_Quiz_Count	Number	Integer

Example

In this example, we have 8 nodes in module #2 of the Assembly Language course. Each node has either video + text files or audio file. There are 3 types of CGM_Node_Type: TOP, MID, BOT.

CGM_ Module_ Name	**CGM_ Node_ Name**	**CGM_ Node_ Row**	**CGM_ Node_ Col**	**CGM_ Node_ Type**	**CGM_ Sibling_ Node**	**CGM_ Node_ Short_ Desc**
M2	C1	1	3	TOP	PROF	C1
M2	C2	2	2	MID	C3	C2
M2	C3	2	3	MID	C4	C3
M2	C4	2	4	MID	PROF	C4
M2	C5	3	3	MID	PROF	C5
M2	C6	4	3	MID	PROF	C6
M2	C7	5	3	MID	PROF	C7
M2	C8	6	3	BOT	PROF	C8

CGM_ Node_ Long_ Desc	**CGM_ Avi_ FName**	**CGM_ Txt_ FName**	**CGM_ Wav_ Fname**	**CGM_ Quiz_ Count**
Register as scratch pad	C1	C1	NA	2
Size and number of registers	NA	NA	C2	1
What to store in a register	NA	NA	C3	1
Classifying the registers	C4	C4	NA	1
Essential registers	NA	NA	C5	4
Attributes of an instruction	C6	C6	NA	2
A sample program	C7	C7	NA	1
Memory map	NA	NA	C8	1

Table 3: **Node Information Table**

4.5.4 CGM_Prev_Node

Description

This table contains the previous node(s) of the current node.

Purpose

We use this table to find the previous node with respect to the current node.

Primary keys

1. Curr_Module
2. Curr_Node_Name
3. Prev_Node_Name

Field Name, Data Types and Field Size

1.	Curr_Module	Text	10
2.	Curr_Node_Name	Text	10
3.	Prev_Node_Name	Text	10

Example

In this example, current node C5 has 3 different previous nodes C2, C3, C4. Current node C2, C3, C4, C6, C7, C8 have only 1 previous node.

Curr_ Module_ Name	**Curr_ Node_ Name**	**Prev_ Node_ Name**
M2	C2	C1
M2	C3	C1
M2	C4	C1
M2	C5	C2
M2	C5	C3
M2	C5	C4
M2	C6	C5
M2	C7	C6
M2	C8	C7

Table 4: **Previous Node Table**

4.5.5 CGM_Next_Node

Description

This table contains the next node(s) and the different range(s) of grade of the current node.

Purpose

We use this table to find the next node based on the grade of the current node.

Primary keys

1. Curr_Module
2. Curr_Node_Name
3. Next_Node_Name

Field Name, Data Types and Field Size

1. Curr_Module	Text	10
2. Curr_Node_Name	Text	10
3. Next_Node_Name	Text	10
4. Curr_Node_Min	Number	Integer
5. Curr_Node_Max	Number	Integer

Example

In this example, current node C1 has 3 different grade ranges for the 3 different next nodes C2,C3,C4. Current node C2, C3, C4, C5, C6, C7 have only 1 grade range because they have only 1 next node.

Curr_ Module_ Name	**Curr_ Node_ Name**	**Next_ Node_ Name**	**Curr_ Node_ Min**	**Curr_ Node_ Max**
M2	C1	C4	50	70
M2	C1	C3	71	85
M2	C1	C2	86	100

M2	C2	C5	50	100
M2	C3	C5	50	100
M2	C4	C5	50	100
M2	C5	C6	50	100
M2	C6	C7	50	100
M2	C7	C8	50	100

Table 5: **Next Node Table**

4.5.6 CGM_Slide_Info

Description

This table contains the information about the slide(s) of the CGM node.

Purpose

We use this table to show the slide(s) of the CGM node using the sequence and time interval given a priori for synchronization purposes.

Primary keys

1. CGM_Module
2. CGM_Node_Name
3. CGM_Slide_Name
4. CGM_Slide_Seq

Field Name, Data Types and Field Size

1. CGM_Module	Text	10
2. CGM_Node_Name	Text	10
3. CGM_Slide_Name	Text	10
4. CGM_Slide_Seq	Number	Integer
5. CGM_Slide_Start_Time	Number	Integer
6. CGM_Slide_End_Time	Number	Integer
7. CGM_Slide_FName	Text	10

Example

In this example, current node C1, C3, C4, C5, C8 have 1 slide; C2, C7 have 2 slides; C6 has 4 slides.

CGM_ Module_ Name	**CGM_ Node_ Name**	**CGM_ Slide_ Name**	**CGM_ Slide_ Seq**	**CGM_ Slide_ Start_ Time**	**CGM_ Slide_ End_ Time**	**CGM_ Slide_ FName**
M2	C1	S11	1	0	10	S11
M2	C2	S21	1	0	10	S21
M2	C2	S22	2	30	50	S22
M2	C3	S31	1	0	40	S31
M2	C4	S41	1	0	20	S41
M2	C5	S51	1	0	45	S51
M2	C6	S61	1	0	30	S61
M2	C6	S62	2	31	56	S62
M2	C6	S63	3	57	100	S63
M2	C6	S64	4	200	300	S64
M2	C7	S71	1	0	40	S71
M2	C7	S72	2	41	90	S72
M2	C8	S81	1	0	100	S81

Table 6: **Slide Information Table**

4.5.7 CGM_Quiz_Info

Description

This table contains the information about the quiz(s) of the CGM node.

Purpose

We use this table to show the quiz question(s) of the given CGM node. The quiz questions can be one of the following types:

- YES/NO or TRUE/FALSE
- Multiple choices

Primary keys

1. CGM_Module
2. CGM_Node_Name
3. CGM_Quiz_Name
4. CGM_Quiz_Seq

Field Name, Data Types and Field Size

1. CGM_Module	Text	10
2. CGM_Node_Name	Text	10
3. CGM_Quiz_Name	Text	10
4. CGM_Quiz_Seq	Number	Integer
5. CGM_Quiz_Grade	Number	Integer
6. CGM_Quiz_FName	Text	10
7. CGM_Ans1_FName	Text	10
8. CGM_Ans2_FName	Text	10
9. CGM_Ans3_FName	Text	10
10. CGM_Ans4_FName	Text	10
11. CGM_Correct_Answer	Number	Integer
12. CGM_Ans_Desc	Text	100

Example

In this example, current node C2, C4, C7 have 1 quiz; C1, C6 have 2 quizzes; C5 has 4 quizzes. Each quiz has maximum 4 choices and one correct answer with the description associated with it.

CGM_ Module_ Name	**CGM_ Node_ Name**	**CGM_ Quiz_ Name**	**CGM_ Quiz_ Seq**	**CGM_ Quiz_ Grade**	**CGM_ Quiz_ FName**	**CGM_ Ans1_ FName**
M2	C1	Q11	1	50	Q11	Q11A1
M2	C1	Q12	2	50	Q12	Q12A1
M2	C2	Q21	1	100	Q21	Q21A1
M2	C4	Q41	1	100	Q41	Q41A1
M2	C5	Q51	1	25	Q51	Q51A1
M2	C5	Q52	2	25	Q52	Q52A1
M2	C5	Q53	3	25	Q53	Q53A1
M2	C5	Q54	4	25	Q54	Q54A1
M2	C6	Q61	1	50	Q61	Q61A1
M2	C6	Q62	2	50	Q62	Q62A1
M2	C7	Q71	1	100	Q71	Q71A1

CGM_ Ans2_ FName	CGM_ Ans3_ Fname	CGM_ Ans4_ FName	CGM_ Correct_ Answer	CGM_ Ans_ Desc
Q11A2	Q11A3	Q11A4	1	The correct answer is number 1
Q12A2	Q12A3	Q12A4	2	The correct answer is number 2
Q21A2	Q21A3	Q21A4	3	The correct answer is number 3
Q41A2	Q41A3	Q41A4	2	The correct answer is number 2
Q51A2	Q51A3	Q51A4	1	The correct answer is number 1
Q52A2	Q52A3	Q52A4	4	The correct answer is number 4
Q53A2	Q53A3	Q53A4	2	The correct answer is number 2
Q54A2	Q54A3	Q54A4	3	The correct answer is number 3
Q61A2	Q61A3	Q61A4	2	The correct answer is number 2
Q62A2	Q62A3	Q62A4	4	The correct answer is number 4
Q71A2	Q71A3	Q71A4	1	The correct answer is number 1

Table 7: **Quiz Information Table (Part 1 & 2)**

4.6 Data Flow Diagram & Process Definition

In this section we describe the overall data flow in the operation of the proposed system. The various databases explained in the previous section are used here.

4.6.1 Database connectivity

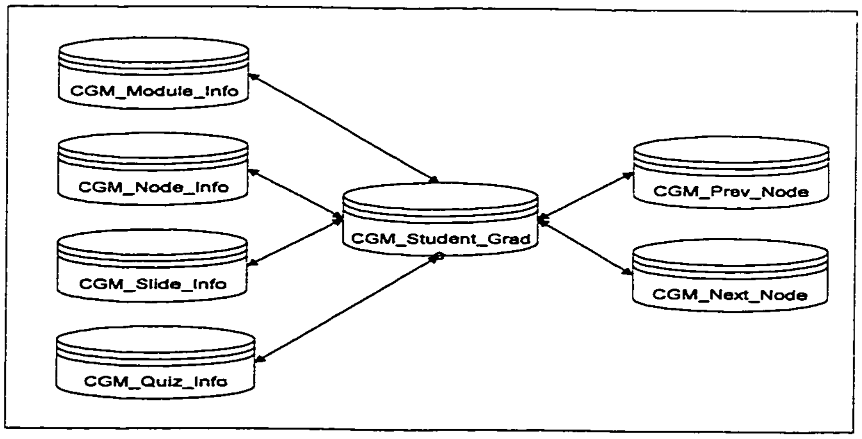


Figure 16: **Database Connectivity**

The above diagram illustrates the overall data flow between the various databases (tables). The left side of the diagram displays the four information tables to complement the current node. The information contained in these tables will be used to drive the application. The information can be the location of the files, the time attribute to drive the slide show or the description of the node etc. The right side of diagram displays the two control node tables. These tables are used to control the flow of software. For example, the CGM_Next_Node ids used to join with the CGM_Student_Grade table to identify the next node that the student has to visit.

4.6.2 Process Model

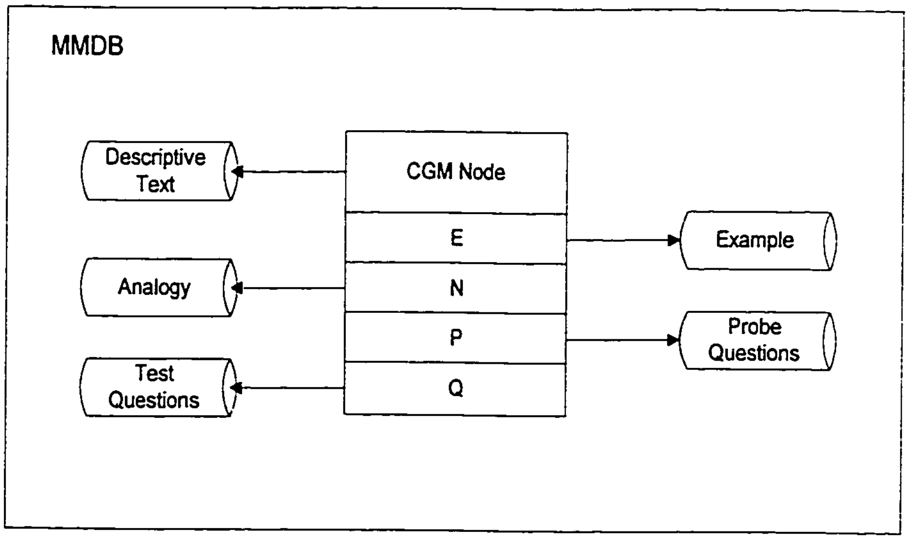


Figure 17: **Process Model**

Legend:

E: {e}	e = example to describe the concept.
N: {n}	n = analogy used by the teacher to explain the concept.

P: $\{p_1, p_2, \ldots p_n\}$ Probe question to examine student's level of understanding $[p_i s$ are linearly ordered or partial ordered$]$.

Q: $\{q_1, q_2, \ldots q_n\}$ q_i = test questions that can be used by teacher or student for practicing the "skill" taught, like mid-terms, exams, quizzes etc.

(Based on the CGM proposed by Dr. Radhakrishnan & Dr. V. Rajaraman).

4.6.3 Detailed Process Model

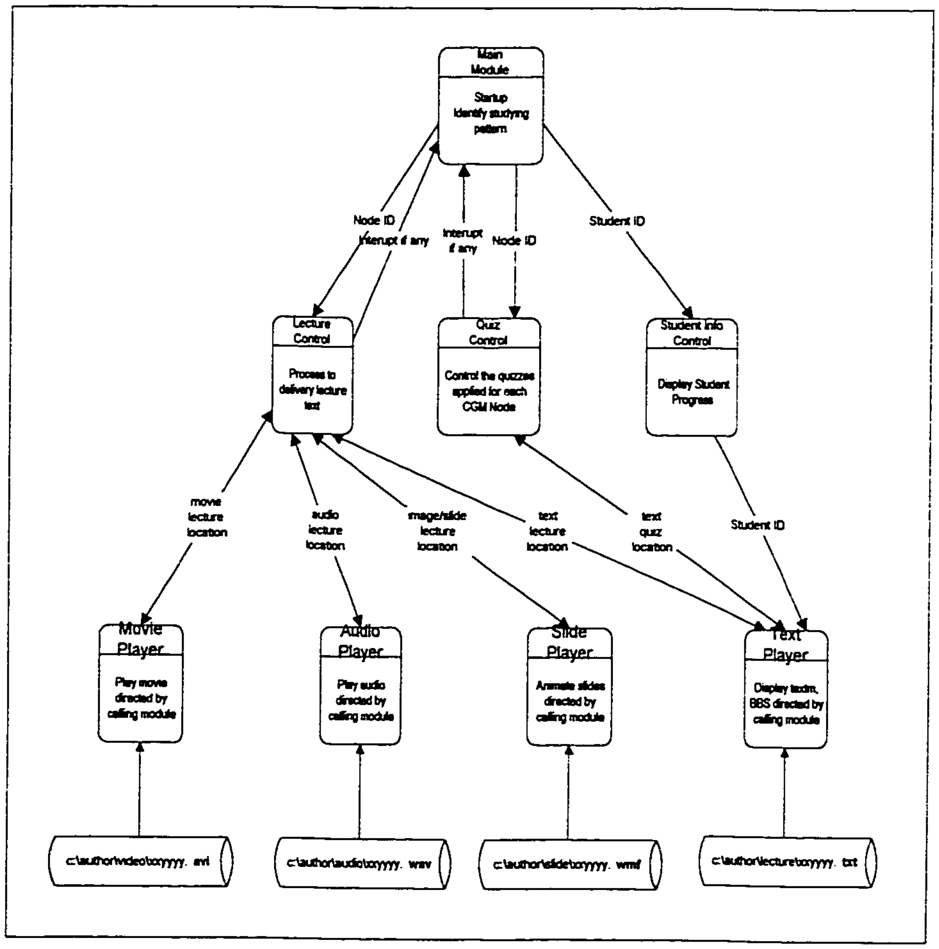

Main Module
Startup Identify studying pattern
Node ID
Interupt if any
Interupt if any
Node ID
Student ID
Lecture Control
Process to delivery lecture text
Quiz Control
Control the quizzes applied for each CGM Node
Student Info Control
Display Student Progress
movie lecture location
audio lecture location
image/slide lecture location
text lecture location
text quiz location
Student ID
Movie Player
Play movie directed by calling module
Audio Player
Play audio directed by calling module
Slide Player
Animate slides directed by calling module
Text Player
Display textm, BBS directed by calling module
c:\author\video\xyyyy. avi
c:\author\audio\xyyyy. wav
c:\author\slide\xyyyy. wmf
c:\author\lecture\xyyyy. txt

Figure 18: **Detail Process Model**

4.6.4 Process Definition

Main Module:

Main driver to control Multimedia Presentation system. By calling appropriate sub modules. Multimedia Presentation system gives lectures to student, monitors students activities, populates quiz questions.

- ➢ Start up Multimedia Presentation system software.
- ➢ Display log in screen.
- ➢ Call initialize module to get student information.
- ➢ Query studying level database to identify the studying pattern of this student.
- ➢ Start lecture, example, quiz where the student left off from last session by calling appropriate player.
- ➢ Display log off screen.

Lecture Control:

Delivery lecture, probe questions, concept example sections following the order from main module. However, in case of interrupts, it should be able to stop and return to main module.

Base on the option selected from student perform:

- ➢ Display video & text

  Call video player and text player at the same time having the same node ID. For example, the lecture would be: C1.Avi and C1.Txt where video clip can be found from c:\author\video and text file can be found from c:\author\text.
- ➢ Display audio & slides

Call audio player and slide player at the same time having the same node ID. For example, the lecture would be: C1.Wav and C1.Wmf where audio file can be found from c:\author\audio and slides can be found from c:\author\slide.

Quiz Control:

This control populates the quizzes to student upon requested by the student. This control calls the text player to display the questions and accepts responses from the student in the form of Yes/No, True/False, and multiple choices. Answers from students are captured, accumulated and diagnosed to identify the next appropriate node to the student.

Student Info Control:

The student can ask for his or her performance during the course. It also compares the level of achievement of that student with the group's average where group's average is calculated upon accumulating the grades of other students. Note that, the group's average is achievable only if it's the network version.

Video Class (video player)

According to the request from the calling module, the appropriate video clips are searched from correct folders and played. N.B: Video player could be interrupted and returned to the calling module. The video player can restart from the frame where it left off or restart from the beginning. Moreover, the video player can perform the fast forward/backward as well.

Audio Class (audio player)

According to the order from the calling module, the appropriate audio files are searched from correct folders and played. N.B: Audio player could be interrupted and returned to the calling module. In this event, the audio player can not restart from where it left off. This module can not be run concurrently with video player otherwise, it makes no sense at all.

Slide Class (slide player)

According to the order from the calling module, the appropriate slides are searched from correct folders and displayed. Since the slide show is time dependent, timer is used to control the flow of the slides. N.B: Slide show is executed at the same time as the audio show.

Text Class (text player)

According to the order from the calling module, the appropriate text files are searched from correct folders and displayed. This class is also a gateway to BBS/FAQ (FAQ will be text only environment).

Chapter 5: A Typical User and the System - Summary & Conclusion

5.1 A Typical User

As we have mentioned in previous chapter, the multimedia presentation system is built to assist beginning students in self-learning of the Assembly language for a PC. Since, the users are novices; the system is built in such the manner that the GUI based user interface is helpful to them. As an example, let us assume that a new student who just joined Concordia University in the Computer Science program, nick named Sam with student ID 1234567 is using our system.

5.2 Walk-through

In order to access and operate the system, the following two conditions have to be satisfied:

a) Sam is registered and authorized to use the system.

b) Sam selects a password.

The system administrator registers a new student by using ***Login & Add Student Panels.*** (See section 4.4.2 to register a new student into the system).

1. Sam uses the ***Login Panel*** to login the system (see section 4.4.3) by entering his student ID and his password. This will lead him to the main menu.

2. From the ***Main Menu Panel***, Sam has the choice to select the module that he wishes to learn. In this menu, he has no restriction in selecting a module at all. This means that he can start module #2 before module #1 (this is subject to change in future extension - see section 5.4).

3. Sam decides to learn the module #2 for now. He, then, clicks the button "Module 2". Two additional buttons will pop up allowing him to start the module #2 or view the introduction.

4. Since, he is new to the system, he would choose "Intro to Module 2" button. He clicks this button. A video clip will be played. The movie will introduce Sam to the content of Module 2 - An assembly sample program.

5. He clicks the "Start" button. The ***Concept Graph Panel*** will appear (see section 4.4.6). Since it's the first time Sam accesses to the system, the first node, C1, in the CGM panel will be green and enabled. All the other nodes will be yellow indicating that these nodes are not visited yet and these nodes are not enabled. Sam will not be able to click on these nodes to start the lecture.

6. To start the lecture of the node C1, Sam could either click on the node C1 itself or click on the button "Curr Node" to start the lecture of node C1. A video clip will be played in the Video/Slide frame to delivery the C1 lecture. At the same time, the text lecture of C1 will be displayed in the "lecture" frame.

7. The control buttons (in the "Control" frame") are changed according to the state of the lecture. For example, while the video clip is playing, the "Stop" button will be activated but the "Play" button will be deactivated.

8. Since the ***CGM panel*** is the most important panel of the application, Sam has to understand the meaning and functionality of each control on this panel. As per our design, the panel is friendly to use and simple to understand. Following is the list of CGM panel characteristics:

The CGM node:

- Green nodes - This is current active node. This node will be played if the button "Curr Node" is clicked. The state of the node is enabled which means that Sam can click the node to start the lecture.
- Red nodes - These nodes were visited and quizzes have been passed (Sam had obtained a mark of 50 or more). These nodes are enabled; Sam can click on these nodes for a review.
- Yellow nodes - These nodes are not visited yet and most of the time, but not all, they are not activated unless Sam has already passed the previous node.

CGM path:

A path (sequence of nodes) in the CGM is appropriately chosen to suite Sam's level of understanding. For example, if Sam passed the node C1 after obtaining a mark of 50, the node C4 will be "green" to suggest that C4 is the next node he should visit. On the other hand, if he passed C1 with mark of 86 or over, the node C2 will turn "green".

Video frame:

When a video clip is played, Sam could move forward or backward the video by dragging the control on the slider under the movie screen. Sam could also stop or restart the video by clicking the square/triangle button under the movie screen. (See section 4.4.8 for more detail).

Slide frame:

This frame shares the space with the movie frame. This means that the movie and the slides will never go together. As Sam will find out that the slides are displayed in conjunctions with an audio clip. The software assumes that the audio and the slide are a pair while the video and the lecture are another pair.

The Sibling button:

This button allows Sam to choose an easier node. For example, Sam passed node C1 with the mark of 86. The application suggests that the next node is C2. Sam visited node C2 and found that it was too difficult for him. He, then, could click "Sibling button" to select an "equivalent node" but simpler in presentation. Upon clicking "Sibling Node" button, the node C3 will be activated. Again, Sam might click "Sibling Node" button if he finds C3 is still difficult to him. And so on, until there are no sibling node to the current active node.

As Sam keeps going on clicking "Sibling" button, at a certain point, there will be no more sibling node and the system will suggest Sam to use the e-mail sub-system to contact his professor. Sam, then, click the "Mail" button on the tool bar. The ***Mail Panel*** (section 4.4.11) will pop up and Sam is ready to sign in the mail system to contact his professor.

At any time, Sam can check his progress status (compared to another students) by clicking "Progress" button.

Sam is also able to get a certain kind of information if he is uncertain about some concepts in the lecture. He can click the "FAQ" button on the tool bar to access the frequently asked question bulletin board. From the ***FAQ Panel***, Sam might find the answers to his questions. On the contrary, Sam might find that the FAQ does not contain his specific question. He, then, can click on the "Send Mail" button and sign on the e-mail sub-system to contact his professor. The mail sub-system in this project has all the basic functions of a mail system.

While the lecture attached to the current node is being delivered, Sam will be able to pause and test his understanding by clicking the "Quiz" button to access to ***Quiz Panel*** (see section 4.4.9). In the Quiz Panel, Sam will encounter a series of questions with multiple choice answers. Upon answering a question, Sam will be advised if his answer is correct or not. In the event if his answer is incorrect, a brief description of the answer will be displayed. When the quiz is completed, the system will update the database with the accumulated mark and the next node suggested by the system will be enabled and marked "green".

5.3 System Summary and Conclusion

The Multimedia Presentation System presented in this report has a GUI interface. Information exchanges between student and the system is controlled by a series of graphical panels where students are able to log into the system, listen to the lecture and perform various tasks.

The subject matter to be taught is organized into several modules where each module is divided into several sections like in the textbook. One section corresponds to one node in the concept graph model. Every module has an associated concept graph as conceived by the authors. For each node of the concept graph, there is a file or multiple files with the following types: text, power point, audio, video or slides to associate with it. As a rule of thumb, the audio will go with slides where the video will work with lecture. This makes sense, because the student will not watch the video and the slides at the same time. Moreover, the real estate (area of screen displayed) is limited.

The concept graph of a selected module for learning is displayed on the screen and the student can click on any *valid* node so that, the corresponding file(s) is displayed. When the video file is played, the student can forward, backward, pause, stop, or resume the video clip. Each text file, which contains the lectures relevant to current concept graph node, will be displayed on a screen during the time the video is played.

A student normally navigates with the help of a concept graph, visiting several nodes as many times as required. The various modules of the course, eight modules in our case, have linear relationships, or may be controlled by another concept graph at a level higher than the module level.

For self-evaluation, the student can choose the quiz that is associated with the current concept graph node. At the same time, the system based on the student's achievement will adjust the flow of the module by routing the student to the most suitable path of nodes. Based on the performance of the student "at a node", the level of difficulty on the next node is determined and the student will be taken to a pre-assisted level of complexity of the next node. If a student does not pass the quiz at the default level "n" then the system will pass the student to a sibling node using simpler examples and conceptually "low level" concepts at level n-1.

To enrich the application, an e-mail sub-system is integrated so that student can send adhoc questions (not appearing the FAQ panel) and receive answers from the professor.

The main goal of this project has been illustrated through a prototype showing how a Multimedia Presentation System can be used for Interactive Learning. In general, a student learning in this manner may use a network like Internet for cooperative learning or use a stand-alone system for self-learning. The current project is aimed as a stand-alone system intended for self-learning. The proposed system for interactive learning was designed, developed, tested and implemented successfully. Extensive usability testing of the system remains to be done.

5.4 Future Extension

1. Adding intelligent heuristics during navigation of the CGM so that it can adjust the details of presentation to match the student needs.

2. Adding voice recognition and speech I/O for the user's convenience.

3. Modifying the system to have the network/Internet version. It will have to send digital video, animation and multimedia data over network. This allows students to take courses remotely. It will help the continuing education students who work during the day and study from home at night.

References

[Adie 96] Chris Adie, *Network Access to Multimedia Information - Summary,* Jan. 96, http://far.mit.edu/diig/NII_info/nami.html.

[CEEP 95] CEE Publication, *Introduction to Multimedia,* Center for Excellence in Education at Indiana University, Oct. 95, http:/cee.indiana.edu/workshops/multipres/MM.html.

[HeGr 93] Brian E. Heckman, Thomas M. Graziano, *Integrating Computer-aided Learning into the University Classroom: A Revised Teaching Model,* Preprint Volume Of The First International Conference On Computer-Aided Learning And Distance Learning In Meteorology, Hydrology, And Oceanography (CALMet), 5-9 July 1993.

[Hols 94] Erik Holsinger, *How multimedia works*, Ziff-Davis Press, CA, 1994.

[ISKM 93] Hiroshi Ishikawa, Fumio Suzuki, Fumihiko Kozakura, Akifumi Makinouchi, *The model, language, and implementation of an object-oriented multimedia knowledge base management system*, ACM Transactions on Database Systems, Vol.18, No. 1 (March 1993), pp. 1-50.

[MiDu 93] Chunsheng Miao & Charles Duncan, *International Cooperation in Developing CAL Material,* Preprint Volume Of The First International Conference On Computer-Aided Learning And Distance Learning In Meteorology, Hydrology, And Oceanography (CALMet), 5-9 July 1993.

[Pate 96] Kinshuk and Ashok Patel, *Intelligent Tutoring Tools - Redesigning ITSs for Adequate Knowledge Transfer Emphasis,* International Conference on Intelligent and Cognitive Systems - ICICS'96, Iran, 23-26 September 1996.

[Gert 95] Nat Gertler, *Multimedia Illustrated*, Que Corporation, IN, 1994.

Appendix – Source Code

FrmAbout – About the application

```
' -------------------------------------------------------------
' Form Name..: frmAbout.frm
' Description: Display information about the application
' -----------------------------------------------------------

Option Explicit

' Reg Key Security Options...
Const READ_CONTROL             = &H20000
Const KEY_QUERY_VALUE          = &H1
Const KEY_SET_VALUE            = &H2
Const KEY_CREATE_SUB_KEY       = &H4
Const KEY_ENUMERATE_SUB_KEYS   = &H8
Const KEY_NOTIFY               = &H10
Const KEY_CREATE_LINK          = &H20
Const KEY_ALL_ACCESS           = KEY_QUERY_VALUE + KEY_SET_VALUE + _
                                 KEY_CREATE_SUB_KEY + KEY_ENUMERATE_SUB_KEYS + _
                                 KEY_NOTIFY + KEY_CREATE_LINK + READ_CONTROL

' Reg Key ROOT Types...
Const HKEY_LOCAL_MACHINE       = &H80000002
Const ERROR_SUCCESS            = 0
Const REG_SZ                   = 1        ' Unicode nul terminated string
Const REG_DWORD = 4                       ' 32-bit number

Const gREGKEYSYSINFOLOC        = "SOFTWARE\Microsoft\Shared Tools Location"
Const gREGVALSYSINFOLOC        = "MSINFO"
Const gREGKEYSYSINFO           = "SOFTWARE\Microsoft\Shared Tools\MSINFO"
Const gREGVALSYSINFO           = "PATH"

Private Declare Function RegOpenKeyEx Lib "advapi32" _
                         Alias "RegOpenKeyExA" _
                        (ByVal hKey As Long, _
                         ByVal lpSubKey As String, _
                         ByVal ulOptions As Long, _
                         ByVal samDesired As Long, _
                         ByRef phkResult As Long) As Long

Private Declare Function RegQuerColValueEx Lib "advapi32" _
                         Alias "RegQuerColValueExA" _
                        (ByVal hKey As Long, _
                         ByVal lpValueName As String, _
                         ByVal lpReserved As Long, _
                         ByRef lpType As Long, _
                         ByVal lpData As String, _
                         ByRef lpcbData As Long) As Long
```

```
Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long


Private Sub cmdQuit_Click()
    End
End Sub

Private Sub cmdSysInfo_Click()
  Call StartSysInfo
End Sub

Private Sub cmdOK_Click()
  frmLogin.Show
  Unload Me
End Sub

Private Sub Form_Load()
    ' Center the form
    Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height) / 2
    lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." & App.Revision
End Sub

Public Sub StartSysInfo()
    On Error GoTo SysInfoErr

    Dim rc As Long
    Dim SysInfoPath As String

    ' Try To Get System Info Program Path\Name From Registry...
    If GetKeColValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO, gREGVALSYSINFO, SysInfoPath) Then
    ' Try To Get System Info Program Path Only From Registry...
    ElseIf GetKeColValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC, _
          gREGVALSYSINFOLOC,SysInfoPath) Then
                ' Validate Existance Of Known 32 Bit File Version
                If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then
                      SysInfoPath = SysInfoPath & "\MSINFO32.EXE"
                Else       ' Error - File Can Not Be Found...
                      GoTo SysInfoErr
                End If
       Else    ' Error - Registry Entry Can Not Be Found...
                GoTo SysInfoErr
       End If
    End If

    Call Shell(SysInfoPath, vbNormalFocus)

    Exit Sub

SysInfoErr:
    MsgBox "System Information Is Unavailable At This Time", vbOKOnly
End Sub

Public Function GetKeColValue(KeyRoot As Long, KeyName As String, _
                              SubKeyRef As String, _
                              ByRef KeyVal As String) As Boolean
    Dim i As Long                           ' Loop Counter
    Dim rc As Long                          ' Return Code
    Dim hKey As Long                        ' Handle To An Open Registry Key
    Dim hDepth As Long                      '
    Dim KeyValType As Long                  ' Data Type Of A Registry Key
    Dim tmpVal As String                    ' Tempory Storage For A Registry Key Value
    Dim KeyValSize As Long                  ' Size Of Registry Key Variable

    ' Open RegKey Under KeyRoot (HKEY_LOCAL_MACHINE...)

    rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey) ' Open Registry Key

    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError       ' Handle Error...

    tmpVal = String$(1024, 0)                            ' Allocate Variable Space
    KeyValSize = 1024                                    ' Mark Variable Size


    ' Retrieve Registry Key Value...
    rc = RegQuerColValueEx(hKey, SubKeyRef, 0, _
                     KeyValType, tmpVal, KeyValSize)  ' Get/Create Key Value

    If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError    ' Handle Errors

    If (Asc(Mid(tmpVal, KeyValSize, 1)) = 0) Then     ' Win95 Adds Null Terminated Str
        tmpVal = Left(tmpVal, KeyValSize - 1)         ' Null Found, Extract From Str
    Else                                              ' WinNT Does NOT Null Terminate
        tmpVal = Left(tmpVal, KeyValSize)             ' Null Not Found, Extract Str
    End If

    ' Determine Key Value Type For Conversion...
    Select Case KeyValType                            ' Search Data Types...
    Case REG_SZ                                       ' String Registry Key Data Type
        KeyVal = tmpVal                               ' Copy String Value
    Case REG_DWORD                                    ' Double Word Reg. Key Data Type
        For i = Len(tmpVal) To 1 Step -1              ' Convert Each Bit
            KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1))) ' Build Value Char. By Char.
        Next
        KeyVal = Format$("&h" + KeyVal)               ' Convert Double Word To String
    End Select

    GetKeColValue = True                                 ' Return Success
    rc = RegCloseKey(hKey)                               ' Close Registry Key
    Exit Function                                        ' Exit

GetKeyError:      ' Cleanup After An Error Has Occured...
    KeyVal = ""                                          ' Set Return Val To Empty String
    GetKeColValue = False                                ' Return Failure
    rc = RegCloseKey(hKey)                               ' Close Registry Key
End Function
```

FrmAddStudent – Add new student to system

```
' ---------------------------------------------------------------------------
' Form Name:   frmAddstudent.frm
' Description: This form is used for registering student name into the system
' ---------------------------------------------------------------------------
Option Explicit

Private Sub cmdCancel_Click()
    Unload Me
End Sub

Private Sub cmdOK_Click()
    With CurrStudent
        .StudentID          = txtStudentID
        .Password           = txtPassword
        .FirstName          = txtFirstName
        .MiddleName         = txtMiddleName
        .LastName           = txtLastName
        .IntelligentLevel    = 5              'Maximum level which student can achieve
        .AccumGrade         = 0
        .Comments           = txtComment
    End With

    stuFileName = App.Path & "\" & txtStudentID & stuFileExtn
    Call PutStudentInfo
    Call InsertIntoCurrNode(txtStudentID)

Unload Me
End Sub

Private Sub Form_Load()
    ' Center the form
    Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height) / 2
```

FrmCGM – CGM main panel

```
'--------------------------------------------------------------
' Form Name:    frmCGM.frm
' Description: This form is the main form of the application.
'              It contains the CGM, video, text, slide … screens.
'--------------------------------------------------------------
Public SveNodeName As String
Public AllSlideCnt As Long
Public SlideShowOn As Boolean

Private Sub cmdCurrNode_Click()

    ' Start current node
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub

Private Sub cmdFAQ_Click()
' -------------------------------
' active for frmFAQ when requested
' -------------------------------

    frmFAQ.Show
    frmCGM.Hide

End Sub
Private Sub cmdHelp_Click()

    frmMessage.rtfMessage = Chr(13) & Chr(10) & Chr(13) & Chr(10) _
                          & "Click a node in the CGM panel" & Chr(13) & Chr(10) _
                          & "to have the to view the lecture" & Chr(13) & Chr(10) _
                          & "Click a button in the control panel" & Chr(13) & Chr(10) _
                          & "to perform an action" & Chr(13) & Chr(10) _
                          & ""

    frmMessage.Show

End Sub

Private Sub cmdMain_Click()

    Unload Me
    frmModule.Show

End Sub

Private Sub cmdNextNode_Click()
' -------------------------------
' Get next available node and then
' process the node
' -------------------------------

    ' re-set the color of current node to what it was
    CurrCntlImg.Picture = imgControlNode.Picture

    Dim tmpNodeName As String
    tmpNodeName = SelectNextNode(CurrStudent.Module, _
                                 CurrCntlImg.Tag, _
                                 CurrStudent.StudentID)

    If tmpNodeName = "NA" Then
        Dim tmpMessage As String
        tmpMessage = "Warning - No next node information was found." & Chr(13) & Chr(10) &
  Chr(13) & Chr(10) _
                    & "Please contact your system administration"
        frmMessage.rtfMessage = tmpMessage
        frmMessage.Show
    Else
        Call ProcessCurrNode(tmpNodeName)
    End If

End Sub


Private Sub cmdPrevNode_Click()
' -------------------------------------------
' get previous node and then process the node
' -------------------------------------------

    CurrCntlImg.Picture = imgControlNode.Picture

    Dim tmpNodeName As String
    tmpNodeName = SelectPrevNode(CurrStudent.Module, _
                                 CurrCntlImg.Tag, _
                                 CurrStudent.StudentID)
    If tmpNodeName = "NA" Then
        Dim tmpMessage As String
        tmpMessage = "Warning - No prev node information was found." & Chr(13) & Chr(10) &
  Chr(13) & Chr(10) _
                    & "Please contact your system administration"
        frmMessage.Tag = "CGM"
        frmMessage.rtfMessage = tmpMessage
        frmMessage.Show
        frmCGM.Hide
    Else
        Call ProcessCurrNode(tmpNodeName)
    End If

End Sub

Private Sub cmdProgress_Click()

    Dim tmpMessage As String
    ' generate heading
    tmpMessage = "The progress of " & Trim(CurrStudent.FirstName) & " " &
  Trim(CurrStudent.LastName) & ":" & Chr(13) & Chr(10) _
              & "Module    Concept     Yours  Class  # Std." & Chr(13) & Chr(10) _
              & "------    -------     -----  -----  ------"

    ' get progress info
    Call SelectProgress(CurrStudent.StudentID)

    ' concatenate progress info
    For i = 1 To ProgressCnt
         tmpMessage = tmpMessage _
                    & Chr(13) & Chr(10) _
                    & StudentProgress(i).ModuleName _
                    & StudentProgress(i).NodeName _
                    & StudentProgress(i).StudentGrade _
                    & StudentProgress(i).GroupGrade _
                    & StudentProgress(i).GroupCount
    Next i

    ' display progress message
    frmMessage.Tag = "CGM"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show

End Sub
Private Sub cmdSibling_Click()
' ---------------------------------------------------
' identify the sibling node and then process the node
' ---------------------------------------------------

    Dim tmpMessage As String

    ' identify the sibling node
    If CGMNodeInfo.CGMSiblingNode = "PROF" Then
        tmpMessage = "Please use the E-Mail sub-system to contact your" & Chr(13) &
  Chr(10) _
                   & "Professor for further explaination or help." & Chr(13) & Chr(10) _
                   & "Thank you"
        frmMessage.Tag = "CGM"
        frmMessage.rtfMessage = tmpMessage
        frmMessage.Show modal
        frmCGM.Hide
    Else
        CurrCntlImg.Picture = imgControlNode.Picture
        Call ProcessCurrNode(CGMNodeInfo.CGMSiblingNode)
    End If

End Sub
Private Sub cmdQuit_Click()

    End

End Sub
Private Sub cmdQuiz_Click()
```

```
' Quiz(zes) of current node is(are) requested by student even though
' student has not finished the node yet
' this command is enabled, which means current node has quiz

    CurNodeName = CurrCntlImg.Tag

    Call ProcessQuizzes(CurrStudent.Module, _
                        CurNodeName)

End Sub

Private Sub cmdPlay_Click()
' --------------------------
' play the current video/audio
' --------------------------

    mciControl.Command = "play"
    Call EnableCommands(mciControl.Mode)

End Sub

Private Sub cmdSendMail_Click()
'-------------------------------------------------------
' Activate Mail subsystem
' Note: Mail system works independently to CGM presenation
' -------------------------------------------------------

    frmVBMail.Show
    frmCGM.Hide

End Sub
Private Sub cmdStop_Click()
' ----------------------
' stop current movie/audio
' ----------------------

    mciControl.Command = "stop"
    Call EnableCommands(mciControl.Mode)

End Sub

Private Sub Form_Load()
' -------------------------------------------------------
' on form load either
' *) process the current node
' *) process initialization
' -------------------------------------------------------

    'center the form
    Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height - 480) / 2

    If InitCGMNode Then
        Call EnableCommands("init")
        Call InitializeAllNodes
        InitCGMNode = False
    Else
        Call ProcessCurrNode(CurNodeName)
    End If

End Sub
Private Sub imgNode0101_Click()


    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0101

    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0102_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0102
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0103_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0103
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0104_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0104
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0105_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0105
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0201_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0201
Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0202_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0202
    Call ProcessCurrNode(CurrCntlImg.Tag)
```

```
End Sub
Private Sub imgNode0203_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0203
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0204_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0204
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0205_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0205
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0301_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0301
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0302_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0302
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0303_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0303
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0304_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0304
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0305_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0305
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0401_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0401
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0402_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0402
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0403_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0403
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0404_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0404
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0405_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0405
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0501_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0501
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0502_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0502
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub

Private Sub imgNode0503_Click()
```

```
    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0503
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0504_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0504
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0505_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0505
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0601_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0601
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0602_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0602
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub

Private Sub imgNode0603_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0603
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0604_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0604
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
Private Sub imgNode0605_Click()

    CurrCntlImg.Picture = imgControlNode.Picture
    Set CurrCntlImg = imgNode0605
    Call ProcessCurrNode(CurrCntlImg.Tag)

End Sub
```

```
Private Sub mnuAbout_Click()
' -----------------
' display about info
' -----------------

    frmMessage.rtfMessage = Chr(13) & Chr(10) & Chr(13) & Chr(10) & Chr(13) & Chr(10) _
                          & "      MULTIMEDIA PRESENTATION SYSTEM" & Chr(13) & Chr(10) _
                          & "      for INTERACTIVE LEARNING" & Chr(13) & Chr(10) _
                          & "      Written by Mai Lan Nguyen" & Chr(13) & Chr(10) _
                          & "      Concordia Unversity, Dec. 1997"

    frmMessage.Show

End Sub

Private Sub mnuExit_Click()

    End

End Sub

Private Sub tmrDrawLine_Timer()
' -----------------------------------------------------
' use timer to draw lines between nodes once on form init
' -----------------------------------------------------

    Dim i As Integer
    ' disable timer so that lines will not be drew again
    tmrDrawLine.Enabled = False

    For i = 1 To LineNumber
        picCGM.Line (CGMLineArray(i).X1, CGMLineArray(i).Y1)-(CGMLineArray(i).X2,
  CGMLineArray(i).Y2)
    Next i


End Sub

Private Sub mciControl_ModeChange(Mode As String)

    ' endable command buttons
    EnableCommands (Mode)

End Sub
Sub ProcessCurrNode(ByVal CurrNodeName As String)
' ----------------------------------------------------------
' process all neccessarily events when this node is clicked.
' ----------------------------------------------------------

    Dim tmpNodeName As String
    Dim TmpNodePosX, TmpNodePosY As Integer

    ' a new conceopt node is requested:
    ' terminate all the the threads that were activated from prev. node
' *) set slide show to off to stop slide show thread if any
' *) stop mciControl
SlideShowOn = False

' make sure movie is not playing
mciControl.Command = "stop"

Call SelectNodeInfo(CurrStudent.Module, _
                    CurrNodeName, _
                    CGMNodeInfo)

' set info in the status bar
sbrCGM.Panels(1).Text = CGMNodeInfo.CGMShortDesc & ": " _
                      & CGMNodeInfo.CGMLongDesc

TmpNodePosX = CGMNodeInfo.CGMNodePosX
TmpNodePosY = CGMNodeInfo.CGMNodePosY

' activate/deactivate commands
Call EnableCommands("playing")

Set CurrCntlImg = IdentifyThisNode(TmpNodePosX, TmpNodePosY)

' reset prev. picture
imgControlNode.Picture = CurrCntlImg.Picture

CurrCntlImg.Picture = frmCGM.imgActiveNode.Picture

lecFileName = CGMNodeInfo.CGMTxtFName
aviFilename = CGMNodeInfo.CGMAviFName
wavFileName = CGMNodeInfo.CGMWavFName

lecFullName = lecFilePath & lecFileName & lecFileExtn
aviFullName = aviFilePath & aviFilename & aviFileExtn
wavFullName = wavFilePath & wavFileName & wavFileExtn

' prepare diaplay area for either video+lecture or audio+slide
' presentation
Call PrepareDisplayArea

If Dir(aviFullName) <> "" Then
    ' play movie clip now
    mciControl.FileName = aviFullName
    mciControl.Command = "play"

    If Dir(lecFullName) <> "" Then
        ' display text box (lecture)
        Dim DisplayText$
        ' process lecture
        Open lecFullName For Binary As #1
        DisplayText$ = Space$(LOF(1))
        Get #1, , DisplayText$
        Close #1
        rtfLecture.Text = DisplayText$  'display file
    End If
    Else
        If Dir(wavFullName) <> "" Then
            ' play audio clip now
            mciControl.FileName = wavFullName
            mciControl.Command = "play"
        End If
        ' *) identify all slides to be displayed
        ' *) identify the time of each slide
        ' *) get start slide time (current)
        ' *) check timer from time to time to identify the lapse time
        ' *) get next slide and displayr
        If CGMNodeInfo.CGMBmpCount > 0 Then
            ' load slide info into array
            SlideShowOn = True
            Call SelectSlideInfo(CurrStudent.Module, _
                                 CurrNodeName)
            If SlideShowOn Then
                fraVideo.Height = fraCGM.Height
                ' process all slides in the array. upon processing,
                ' yield resources to other processes
                Call ProcessSlides
            End If
        End If
    End If

End Sub
Private Sub PrepareDisplayArea()

    If aviFilename <> "NA" Then
        fraVideo.Height = 3275
        mciControl.Visible = True
        fraLecture.Visible = True
        rtfLecture.Visible = True
        rtfLecture.Enabled = True
    Else
        fraLecture.Visible = False
        rtfLecture.Enabled = False
        rtfLecture.Visible = False
        mciControl.Visible = False
    End If

End Sub

Private Sub EnableCommands(ByVal Mode As String)
' -----------------------------------------------------
' Enabled/Disabled command buttons base on the apps mode
' -----------------------------------------------------

    'frmcgm.Caption = cgm
    If CGMNodeInfo.CGMNodeType = "TOP" Or CGMNodeInfo.CGMNodeType = "T&B" Then
        frmCGM.cmdPrevNode.Enabled = False
    Else
        frmCGM.cmdPrevNode.Enabled = True
    End If
    If CGMNodeInfo.CGMNodeType = "BOT" Or CGMNodeInfo.CGMNodeType = "T&B" Then
```

```
        frmCGM.cmdNextNode.Enabled = False
    Else
        If CGMGradeArray(CGMNodeInfo.CGMNodePosX, _
                        CGMNodeInfo.CGMNodePosY) >= 50 Then
            frmCGM.cmdNextNode.Enabled = True
        Else
            frmCGM.cmdNextNode.Enabled = False
        End If
    End If
    If CGMNodeInfo.CGMSiblingNode = "PROF" Then
        frmCGM.cmdSibling.Enabled = False
    Else
        frmCGM.cmdSibling.Enabled = True
    End If

    Select Case Mode
        Case Is = "init"
            frmCGM.cmdPrevNode.Enabled = False
            frmCGM.cmdCurrNode.Enabled = True
            frmCGM.cmdNextNode.Enabled = False
            frmCGM.cmdPlay.Enabled = False
            frmCGM.cmdStop.Enabled = False
        Case Is = "playing"
            frmCGM.cmdCurrNode.Enabled = False
            frmCGM.cmdPlay.Enabled = False
            frmCGM.cmdStop.Enabled = True
        Case Is = "stopped"
            cmdCurrNode.Enabled = True
            cmdPlay.Enabled = True
            cmdStop.Enabled = False
    End Select

End Sub
Private Sub SelectSlideInfo(ByVal CurrModuleName As String, _
                            ByVal CurrNodeName As String)
' ---------------------------------------------------------------
' get information to slides that associate with current module/node
' ---------------------------------------------------------------


    Dim AllSlideDbs As Database, AllSlideRst As Recordset
    Dim tmpSQL As String
    Dim tmpMessage As String

    Dim i As Integer

    On Error GoTo ErrorHandler1

    Set AllSlideDbs = OpenDatabase(CGMDatabase)

    tmpSQL = " SELECT CGM_Slide_Name," _
           & " Slide_Seq_Number," _
           & " Slide_Begin_Time," _
           & " Slide_End_Time," _
           & " CGM_Slide_FName " _
```

```
            & " FROM CGM_Slide_Info" _
            & " WHERE CGM_Module_Name = '" & CurrModuleName & "'" _
            & "   AND CGM_Node_Name = '" & CurrNodeName & "'" _
            & " ORDER BY Slide_Seq_Number;"

    ' get all the info of slides of the current node
    Set AllSlideRst = AllSlideDbs.OpenRecordset(tmpSQL)

    On Error GoTo ErrorHandler2
    ' go to last record to count
    AllSlideRst.MoveLast
    AllSlideCnt = AllSlideRst.RecordCount

    ReDim CGMSlideInfo(AllSlideCnt)
    ' go to 1st row in recordset
    AllSlideRst.MoveFirst

    On Error GoTo ErrorHandler3

    ' Loop thru recordset to get Slide names and attributes etc...
    For i = 1 To AllSlideCnt
        With CGMSlideInfo(i)
            .CGMSlideName = AllSlideRst.Fields(0)
            .SlideSeqNumber = AllSlideRst.Fields(1)
            .SlideBeginTime = AllSlideRst.Fields(2)
            .SlideEndTime = AllSlideRst.Fields(3)
            .CGMSlideFName = AllSlideRst.Fields(4)
        End With
        ' verifying slide info
        bmpFullName = bmpFilePath & CGMSlideInfo(i).CGMSlideFName & bmpFileExtn
        If Dir(bmpFullName) = "" Then
            SlideShowOn = False
            tmpMessage = "Warning - Not able to find slide" & Chr(13) & Chr(10) _
                       & bmpFullName & Chr(13) & Chr(10) & Chr(13) & Chr(10) _
                       & "Please contact your system administration"
            frmMessage.rtfMessage = tmpMessage
            frmMessage.Show

            AllSlideDbs.Close

            Exit Sub
        End If

        ' advance to next row
        AllSlideRst.MoveNext
    Next i

    AllSlideDbs.Close

    Exit Sub

' Handle fatal error => stop apps
ErrorHandler1:
    tmpMessage = "Fatal Error - Authoring Database was not found." & Chr(13) & Chr(10) &
  Chr(13) & Chr(10) _
                & "Please contact your system administration"
    frmMessage.Tag = "END"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmCGM.Hide

    Exit Sub

ErrorHandler2:

    tmpMessage = "Warning - No SLIDE information was found." & Chr(13) & Chr(10) & Chr(13)
  & Chr(10) _
                & "Please contact your system administration"

    frmMessage.Tag = "CGM"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmCGM.Hide

    Exit Sub

ErrorHandler3:

    tmpMessage = "Fatal Error - Problem upon loading SLIDE." & Chr(13) & Chr(10) & Chr(13)
  & Chr(10) _
                & "Please contact your system administration"

    frmMessage.Tag = "END"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmCGM.Hide

    Exit Sub

End Sub
Private Sub ProcessSlides()

' perform slide show when SlideShowOn = true otherwise terminate the thread

    Dim CurSlideNdx As Integer
    Dim CurShowTimer As Long
    Dim BegSlideTime As Long
    Dim EndSlideTime As Long

    CurShowTimer = Timer
    ' try to display all slides of the current node
    For CurSlideNdx = 1 To AllSlideCnt

        BegSlideTime = CurShowTimer + CGMSlideInfo(CurSlideNdx).SlideBeginTime
        EndSlideTime = CurShowTimer + CGMSlideInfo(CurSlideNdx).SlideEndTime

        ' wait until time to display curr slide request
        Do While Timer < BegSlideTime And SlideShowOn
```

```
            DoEvents
        Loop

        ' time to display current slide
        ' to be on the safe side
        ' we check both end time and slide show flag
        If Timer < EndSlideTime And SlideShowOn Then
            imgSlide.Visible = True
            Call LoadOneSlide(CurSlideNdx)
        End If

        ' keep displaying slide
        Do While Timer < EndSlideTime And SlideShowOn
            DoEvents
        Loop

        ' display time is over
        Set imgSlide.Picture = LoadPicture()
        imgSlide.Visible = False

        If Not SlideShowOn Then
            Exit Sub
        End If

     Next CurSlideNdx

End Sub

Private Sub LoadOneSlide(SlideCntlNbr As Integer)
' ----------------
' load slide  wmf
' ----------------

    bmpFullName = bmpFilePath & CGMSlideInfo(SlideCntlNbr).CGMSlideFName & bmpFileExtn
    If Dir(bmpFullName) <> "" Then
        On Error Resume Next
        Set imgSlide.Picture = LoadPicture(bmpFullName)
        If Err Then
            Set imgSlide.Picture = LoadPicture()
            SlideShowOn = False
        End If
    Else
        SlideShowOn = False
        Dim tmpMessage As String
        tmpMessage = "Warning - Not able to find slide" & Chr(13) & Chr(10) _
                   & bmpFullName & Chr(13) & Chr(10) & Chr(13) & Chr(10) _
                   & "Please contact your system administration"
        frmMessage.Tag = "CGM"
        frmMessage.rtfMessage = tmpMessage
        frmMessage.Show
        frmCGM.Hide
    End If

End Sub
Private Sub ProcessQuizzes(ByVal CurrModuleName As String, _
                                ByVal CurrNodeName As String)
' ----------------------------------------------
' process quizzes that associate with current node
' ----------------------------------------------

    Dim AllQuizCnt As Long
    Dim AllQuizDbs As Database, AllQuizRst As Recordset
    Dim tmpSQL As String
    Dim i As Integer

    On Error GoTo ErrorHandler1
    Set AllQuizDbs = OpenDatabase(CGMDatabase)

    tmpSQL = " SELECT CGM_Quiz_Name ," _
           & " CGM_Quiz_Grade ," _
           & " CGM_Quiz_FName ," _
           & " CGM_Ans1_FName ," _
           & " CGM_Ans2_FName ," _
           & " CGM_Ans3_FName ," _
           & " CGM_Ans4_FName ," _
           & " CGM_Correct_Answer ," _
           & " CGM_Ans_Desc" _
           & " FROM CGM_Quiz_Info" _
           & " WHERE CGM_Module_Name = '" & CurrModuleName & "'" _
           & "   AND CGM_Node_Name = '" & CurrNodeName & "';"

    ' get all the info of quizzes of the current node
    Set AllQuizRst = AllQuizDbs.OpenRecordset(tmpSQL)

    ' no data found
    On Error GoTo ErrorHandler2

    ' go to last record to count
    AllQuizRst.MoveLast
    AllQuizCnt = AllQuizRst.RecordCount

    CurrStudent.AccumGrade = 0
    ReDim CGMQuizInfo(AllQuizCnt)
    ' go to 1st row in recordset
    AllQuizRst.MoveFirst
    ' Loop thru recordset to get quiz names and attributes etc...
    For i = 1 To AllQuizCnt
        With CGMQuizInfo(i)
            .CGMQuizName = AllQuizRst.Fields(0)
            .CGMQuizGrade = AllQuizRst.Fields(1)
            .CGMQuizFName = AllQuizRst.Fields(2)
            .CGMAns1FName = AllQuizRst.Fields(3)
            .CGMAns2FName = AllQuizRst.Fields(4)
            .CGMAns3FName = AllQuizRst.Fields(5)
            .CGMAns4FName = AllQuizRst.Fields(6)
            .CGMCorrectAnswer = AllQuizRst.Fields(7)
            .CGMAnsDesc = AllQuizRst.Fields(8)
        End With
        ' advance to next row
        AllQuizRst.MoveNext
    Next i

    AllQuizDbs.Close

    QuizCntlNbr = 1
    QuizCntlMax = AllQuizCnt
    Unload Me
    frmQuiz.Show

Exit Sub

' Handle fatal error => stop apps
ErrorHandler1:

    Dim tmpMessage As String
    tmpMessage = "Fatal Error - Authoring Database was not found." & Chr(13) & Chr(10) &
  Chr(13) & Chr(10) _
              & "Please contact your system administration"

    frmMessage.Tag = "END"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmCGM.Hide

    Exit Sub

' no quiz was found in quiz table but the number of quizzes was
' not equal to zero in the node info table => could be mistake
ErrorHandler2:

    tmpMessage = "Warning - No quiz information was found." & Chr(13) & Chr(10) & Chr(13)
  & Chr(10) _
              & "Please contact your system administration"

    frmMessage.Tag = "CGM"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmCGM.Hide

    Exit Sub

End Sub
Sub InitializeAllNodes()
' ---------------------------------------------------
' *) Get all available nodes from database and store them in
'    recordset array ALLNODERST
' *) Loop through ALLNODERST to intialize each node
' ---------------------------------------------------

    Dim AllNodeCnt As Long
    Dim tmpNodeName, TmpNodeDesc, TmpStudentID As String
    Dim TmpNodePosX, TmpNodePosY, TmpNodeGrad As Integer
    Dim AllNodeDbs As Database, AllNodeRst As Recordset
    Dim tm2NodeName As String
```

```
 Dim tmpSQL As String
 Dim i As Integer

 On Error GoTo ErrorHandler1
 Set AllNodeDbs = OpenDatabase(CGMDatabase)

 tmpSQL = " SELECT C.Curr_Node_Name ," _
        & " C.Curr_Node_Grade" _
        & " FROM CGM_Curr_Node AS C " _
        & " WHERE C.Curr_Student_ID = '" & CurrStudent.StudentID & "'" _
        & "   AND C.Curr_Module_Name = '" & CurrStudent.Module & "';"

 Set AllNodeRst = AllNodeDbs.OpenRecordset(tmpSQL)

 On Error GoTo ErrorHandler2
 ' go to last record to count
 AllNodeRst.MoveLast
 AllNodeCnt = AllNodeRst.RecordCount

 ' go to 1st row in recordset
 AllNodeRst.MoveFirst
 ' save starting node as the 1st node, this might be changed
 ' in sub IntializeThisNode
 tm2NodeName = AllNodeRst.Fields(0)
 SveNodeName = ""
 ' Loop thru recordset to get node names and attributes etc...
 LineNumber = 0
 For i = 0 To AllNodeCnt - 1
     tmpNodeName = AllNodeRst.Fields(0)
     TmpNodeGrad = AllNodeRst.Fields(1)
     Call SelectNodeInfo(CurrStudent.Module, _
                         tmpNodeName, _
                         CGMNodeInfo)
     TmpNodePosX = CGMNodeInfo.CGMNodePosX
     TmpNodePosY = CGMNodeInfo.CGMNodePosY
     TmpNodeDesc = CGMNodeInfo.CGMShortDesc
     CGMGradeArray(TmpNodePosX, TmpNodePosY) = TmpNodeGrad
     Call InitializeThisNode(TmpNodePosX, _
                             TmpNodePosY, _
                             tmpNodeName, _
                             TmpNodeDesc, _
                             TmpNodeGrad)
      ' advance to next row
     AllNodeRst.MoveNext
 Next i

 If SveNodeName = "" Then
     tmpNodeName = tm2NodeName
 Else
     tmpNodeName = SelectNextNode(CurrStudent.Module, _
                                  SveNodeName, _
                                  CurrStudent.StudentID)
 End If
```

```
    ' identify current active node
    Call SelectNodeInfo(CurrStudent.Module, _
                        tmpNodeName, _
                        CGMNodeInfo)

    Set CurrCntlImg = IdentifyThisNode(CGMNodeInfo.CGMNodePosX, _
                                       CGMNodeInfo.CGMNodePosY)

    imgControlNode.Picture = CurrCntlImg.Picture
    ' change the color of the node after the save
    CurrCntlImg.Picture = frmCGM.imgActiveNode.Picture

    tmrDrawLine.Enabled = True
    AllNodeDbs.Close

    Exit Sub

' Handle fatal error => stop apps
ErrorHandler1:

    Dim tmpMessage As String
    tmpMessage = "Fatal Error - Authoring Database was not found." & Chr(13) & Chr(10) &
  Chr(13) & Chr(10) _
             & "Please contact your system administration"

    frmMessage.Tag = "END"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmCGM.Hide
    Unload Me

    Exit Sub

ErrorHandler2:

    tmpMessage = "Fatal Error - No NODE information was found." & Chr(13) & Chr(10) &
  Chr(13) & Chr(10) _
             & "Please contact your system administration"

    frmMessage.Tag = "END"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmCGM.Hide
    Unload Me

    Exit Sub

End Sub

Sub InitializeThisNode(ByVal RowValue As Integer, _
                       ByVal ColValue As Integer, _
                       ByVal NodeName As String, _
                       ByVal NodeDesc As String, _
                       ByVal NodeGrade As Integer)
' -----------------------------------------------------------------
' Initialize node (RowValue, ColValue) using NodeName, NodeDesc, Enabled
' ------------------------------------------------------------------
    Dim OnOrOff As Boolean
    Dim tmpNodeName As String

    Set CurrCntlImg = IdentifyThisNode(RowValue, ColValue)
    ' quiz must be passed to be considered as visited
    If NodeGrade >= 50 Then
        CurrCntlImg.Picture = frmCGM.imgVisitedNode.Picture
    Else
        CurrCntlImg.Picture = frmCGM.imgUnvisitedNode.Picture
    End If

    If CGMNodeInfo.CGMNodeType = "TOP" Or _
       CGMNodeInfo.CGMNodeType = "T&B" Or _
       NodeGrade >= 50 Then
        ' enable the node if it's first node or it's a visited node
        OnOrOff = True
        If NodeGrade >= 50 Then
            ' save the node that the student passed. This will be used to determine next
  active node
            SveNodeName = NodeName
        End If
    Else
        ' a node should also be enabled if the prev. node is passed
        tmpNodeName = SelectPrevNode(CurrStudent.Module, _
                                     NodeName, _
                                     CurrStudent.StudentID)
        Call SelectNodeInfo(CurrStudent.Module, _
                            tmpNodeName, _
                            CGMNodeInfo)
        If CGMGradeArray(CGMNodeInfo.CGMNodePosX, _
                         CGMNodeInfo.CGMNodePosY) >= 50 Then
            OnOrOff = True
        Else
            OnOrOff = False
        End If
    End If

    Select Case RowValue
    Case Is = 1
        Select Case ColValue
        Case Is = 1
            frmCGM.imgNode0101.Visible = True
            frmCGM.imgNode0101.Tag = NodeName
            frmCGM.imgNode0101.Enabled = OnOrOff
            frmCGM.lblNode0101.Caption = NodeDesc
        Case Is = 2
            frmCGM.imgNode0102.Visible = True
            frmCGM.imgNode0102.Tag = NodeName
            frmCGM.imgNode0102.Enabled = OnOrOff
            frmCGM.lblNode0102.Caption = NodeDesc
        Case Is = 3
            frmCGM.imgNode0103.Visible = True
            frmCGM.imgNode0103.Tag = NodeName
            frmCGM.imgNode0103.Enabled = OnOrOff
            frmCGM.lblNode0103.Caption = NodeDesc
        Case Is = 4
            frmCGM.imgNode0104.Visible = True
            frmCGM.imgNode0104.Tag = NodeName
            frmCGM.imgNode0104.Enabled = OnOrOff
            frmCGM.lblNode0104.Caption = NodeDesc
        Case Is = 5
            frmCGM.imgNode0105.Visible = True
            frmCGM.imgNode0105.Tag = NodeName
            frmCGM.imgNode0105.Enabled = OnOrOff
            frmCGM.lblNode0105.Caption = NodeDesc
        End Select
    Case Is = 2
        Select Case ColValue
        Case Is = 1
            frmCGM.imgNode0201.Visible = True
            frmCGM.imgNode0201.Tag = NodeName
            frmCGM.imgNode0201.Enabled = OnOrOff
            frmCGM.lblNode0201.Caption = NodeDesc
        Case Is = 2
            frmCGM.imgNode0202.Visible = True
            frmCGM.imgNode0202.Tag = NodeName
            frmCGM.imgNode0202.Enabled = OnOrOff
            frmCGM.lblNode0202.Caption = NodeDesc
        Case Is = 3
            frmCGM.imgNode0203.Visible = True
            frmCGM.imgNode0203.Tag = NodeName
            frmCGM.imgNode0203.Enabled = OnOrOff
            frmCGM.lblNode0203.Caption = NodeDesc
        Case Is = 4
            frmCGM.imgNode0204.Visible = True
            frmCGM.imgNode0204.Tag = NodeName
            frmCGM.imgNode0204.Enabled = OnOrOff
            frmCGM.lblNode0204.Caption = NodeDesc
        Case Is = 5
            frmCGM.imgNode0205.Visible = True
            frmCGM.imgNode0205.Tag = NodeName
            frmCGM.imgNode0205.Enabled = OnOrOff
            frmCGM.lblNode0205.Caption = NodeDesc
        End Select
        If frmCGM.imgNode0101.Visible Then
           Call LogLineCordinates(1, 1, RowValue, ColValue)
        End If
        If frmCGM.imgNode0102.Visible Then
           Call LogLineCordinates(1, 2, RowValue, ColValue)
        End If
        If frmCGM.imgNode0103.Visible Then
           Call LogLineCordinates(1, 3, RowValue, ColValue)
        End If
        If frmCGM.imgNode0104.Visible Then
           Call LogLineCordinates(1, 4, RowValue, ColValue)
        End If
        If frmCGM.imgNode0105.Visible Then
           Call LogLineCordinates(1, 5, RowValue, ColValue)
        End If
    Case Is = 3
        Select Case ColValue
        Case Is = 1
            frmCGM.imgNode0301.Visible = True
            frmCGM.imgNode0301.Tag = NodeName
            frmCGM.imgNode0301.Enabled = OnOrOff
            frmCGM.lblNode0301.Caption = NodeDesc
        Case Is = 2
            frmCGM.imgNode0302.Visible = True
            frmCGM.imgNode0302.Tag = NodeName
            frmCGM.imgNode0302.Enabled = OnOrOff
            frmCGM.lblNode0302.Caption = NodeDesc
        Case Is = 3
            frmCGM.imgNode0303.Visible = True
            frmCGM.imgNode0303.Tag = NodeName
            frmCGM.imgNode0303.Enabled = OnOrOff
            frmCGM.lblNode0303.Caption = NodeDesc
        Case Is = 4
            frmCGM.imgNode0304.Visible = True
            frmCGM.imgNode0304.Tag = NodeName
            frmCGM.imgNode0304.Enabled = OnOrOff
            frmCGM.lblNode0304.Caption = NodeDesc
        Case Is = 5
            frmCGM.imgNode0305.Visible = True
            frmCGM.imgNode0305.Tag = NodeName
            frmCGM.imgNode0305.Enabled = OnOrOff
            frmCGM.lblNode0305.Caption = NodeDesc
        End Select
        If frmCGM.imgNode0201.Visible Then
           Call LogLineCordinates(2, 1, RowValue, ColValue)
        End If
        If frmCGM.imgNode0202.Visible Then
           Call LogLineCordinates(2, 2, RowValue, ColValue)
        End If
        If frmCGM.imgNode0203.Visible Then
           Call LogLineCordinates(2, 3, RowValue, ColValue)
        End If
        If frmCGM.imgNode0204.Visible Then
           Call LogLineCordinates(2, 4, RowValue, ColValue)
        End If
        If frmCGM.imgNode0205.Visible Then
           Call LogLineCordinates(2, 5, RowValue, ColValue)
        End If
    Case Is = 4
        Select Case ColValue
        Case Is = 1
            frmCGM.imgNode0401.Visible = True
            frmCGM.imgNode0401.Tag = NodeName
            frmCGM.imgNode0401.Enabled = OnOrOff
            frmCGM.lblNode0401.Caption = NodeDesc
        Case Is = 2
            frmCGM.imgNode0402.Visible = True
            frmCGM.imgNode0402.Tag = NodeName
            frmCGM.imgNode0402.Enabled = OnOrOff
            frmCGM.lblNode0402.Caption = NodeDesc
        Case Is = 3
            frmCGM.imgNode0403.Visible = True
            frmCGM.imgNode0403.Tag = NodeName
            frmCGM.imgNode0403.Enabled = OnOrOff
            frmCGM.lblNode0403.Caption = NodeDesc
        Case Is = 4
            frmCGM.imgNode0404.Visible = True
            frmCGM.imgNode0404.Tag = NodeName
            frmCGM.imgNode0404.Enabled = OnOrOff
            frmCGM.lblNode0404.Caption = NodeDesc
        Case Is = 5
            frmCGM.imgNode0405.Visible = True
            frmCGM.imgNode0405.Tag = NodeName
            frmCGM.imgNode0405.Enabled = OnOrOff
            frmCGM.lblNode0405.Caption = NodeDesc
        End Select
        If frmCGM.imgNode0301.Visible Then
           Call LogLineCordinates(3, 1, RowValue, ColValue)
        End If
        If frmCGM.imgNode0302.Visible Then
           Call LogLineCordinates(3, 2, RowValue, ColValue)
        End If
        If frmCGM.imgNode0303.Visible Then
           Call LogLineCordinates(3, 3, RowValue, ColValue)
        End If
        If frmCGM.imgNode0304.Visible Then
           Call LogLineCordinates(3, 4, RowValue, ColValue)
        End If
        If frmCGM.imgNode0305.Visible Then
           Call LogLineCordinates(3, 5, RowValue, ColValue)
        End If
    Case Is = 5
        Select Case ColValue
        Case Is = 1
            frmCGM.imgNode0501.Visible = True
            frmCGM.imgNode0501.Tag = NodeName
            frmCGM.imgNode0501.Enabled = OnOrOff
            frmCGM.lblNode0501.Caption = NodeDesc
        Case Is = 2
            frmCGM.imgNode0502.Visible = True
            frmCGM.imgNode0502.Tag = NodeName
            frmCGM.imgNode0502.Enabled = OnOrOff
            frmCGM.lblNode0502.Caption = NodeDesc
        Case Is = 3
            frmCGM.imgNode0503.Visible = True
            frmCGM.imgNode0503.Tag = NodeName
            frmCGM.imgNode0503.Enabled = OnOrOff
            frmCGM.lblNode0503.Caption = NodeDesc
        Case Is = 4
            frmCGM.imgNode0504.Visible = True
            frmCGM.imgNode0504.Tag = NodeName
            frmCGM.imgNode0504.Enabled = OnOrOff
            frmCGM.lblNode0504.Caption = NodeDesc
        Case Is = 5
```

```
            frmCGM.imgNode0505.Visible = True
            frmCGM.imgNode0505.Tag = NodeName
            frmCGM.imgNode0505.Enabled = OnOrOff
            frmCGM.lblNode0505.Caption = NodeDesc
        End Select
        If frmCGM.imgNode0401.Visible Then
           Call LogLineCordinates(4, 1, RowValue, ColValue)
        End If
        If frmCGM.imgNode0402.Visible Then
           Call LogLineCordinates(4, 2, RowValue, ColValue)
        End If
        If frmCGM.imgNode0403.Visible Then
           Call LogLineCordinates(4, 3, RowValue, ColValue)
        End If
        If frmCGM.imgNode0404.Visible Then
           Call LogLineCordinates(4, 4, RowValue, ColValue)
        End If
        If frmCGM.imgNode0405.Visible Then
           Call LogLineCordinates(4, 5, RowValue, ColValue)
        End If
    Case Is = 6
        Select Case ColValue
        Case Is = 1
            frmCGM.imgNode0601.Visible = True
            frmCGM.imgNode0601.Tag = NodeName
            frmCGM.imgNode0601.Enabled = OnOrOff
            frmCGM.lblNode0601.Caption = NodeDesc
        Case Is = 2
            frmCGM.imgNode0602.Visible = True
            frmCGM.imgNode0602.Tag = NodeName
            frmCGM.imgNode0602.Enabled = OnOrOff
            frmCGM.lblNode0602.Caption = NodeDesc
        Case Is = 3
            frmCGM.imgNode0603.Visible = True
            frmCGM.imgNode0603.Tag = NodeName
            frmCGM.imgNode0603.Enabled = OnOrOff
            frmCGM.lblNode0603.Caption = NodeDesc
        Case Is = 4
            frmCGM.imgNode0604.Visible = True
            frmCGM.imgNode0604.Tag = NodeName
            frmCGM.imgNode0604.Enabled = OnOrOff
            frmCGM.lblNode0604.Caption = NodeDesc
        Case Is = 5
            frmCGM.imgNode0605.Visible = True
            frmCGM.imgNode0605.Tag = NodeName
            frmCGM.imgNode0605.Enabled = OnOrOff
            frmCGM.lblNode0605.Caption = NodeDesc
        End Select
        If frmCGM.imgNode0501.Visible Then
           Call LogLineCordinates(5, 1, RowValue, ColValue)
        End If
        If frmCGM.imgNode0502.Visible Then
           Call LogLineCordinates(5, 2, RowValue, ColValue)
        End If
        If frmCGM.imgNode0503.Visible Then
```

```
          Call LogLineCordinates(5, 3, RowValue, ColValue)
        End If
        If frmCGM.imgNode0504.Visible Then
          Call LogLineCordinates(5, 4, RowValue, ColValue)
        End If
        If frmCGM.imgNode0505.Visible Then
          Call LogLineCordinates(5, 5, RowValue, ColValue)
        End If
    End Select
End Sub
Function IdentifyThisNode(ByVal RowValue As Integer, _
                          ByVal ColValue As Integer) _
                          As Object
' ---------------------------------------------------------------
' having the X-Y cordinate, return the corresponding image node
' ---------------------------------------------------------------
    Select Case RowValue
    Case Is = 1
        Select Case ColValue
        Case Is = 1
           Set IdentifyThisNode = frmCGM.imgNode0101
        Case Is = 2
           Set IdentifyThisNode = frmCGM.imgNode0102
        Case Is = 3
           Set IdentifyThisNode = frmCGM.imgNode0103
        Case Is = 4
           Set IdentifyThisNode = frmCGM.imgNode0104
        Case Is = 5
           Set IdentifyThisNode = frmCGM.imgNode0105
        End Select
    Case Is = 2
        Select Case ColValue
        Case Is = 1
           Set IdentifyThisNode = frmCGM.imgNode0201
        Case Is = 2
           Set IdentifyThisNode = frmCGM.imgNode0202
        Case Is = 3
           Set IdentifyThisNode = frmCGM.imgNode0203
        Case Is = 4
           Set IdentifyThisNode = frmCGM.imgNode0204
        Case Is = 5
           Set IdentifyThisNode = frmCGM.imgNode0205
        End Select
    Case Is = 3
        Select Case ColValue
        Case Is = 1
           Set IdentifyThisNode = frmCGM.imgNode0301
        Case Is = 2
           Set IdentifyThisNode = frmCGM.imgNode0302
        Case Is = 3
           Set IdentifyThisNode = frmCGM.imgNode0303
        Case Is = 4
           Set IdentifyThisNode = frmCGM.imgNode0304
        Case Is = 5
           Set IdentifyThisNode = frmCGM.imgNode0305
        End Select
    Case Is = 4
        Select Case ColValue
        Case Is = 1
           Set IdentifyThisNode = frmCGM.imgNode0401
        Case Is = 2
           Set IdentifyThisNode = frmCGM.imgNode0402
        Case Is = 3
           Set IdentifyThisNode = frmCGM.imgNode0403
        Case Is = 4
           Set IdentifyThisNode = frmCGM.imgNode0404
        Case Is = 5
           Set IdentifyThisNode = frmCGM.imgNode0405
        End Select
    Case Is = 5
        Select Case ColValue
        Case Is = 1
           Set IdentifyThisNode = frmCGM.imgNode0501
        Case Is = 2
           Set IdentifyThisNode = frmCGM.imgNode0502
        Case Is = 3
           Set IdentifyThisNode = frmCGM.imgNode0503
        Case Is = 4
           Set IdentifyThisNode = frmCGM.imgNode0504
        Case Is = 5
           Set IdentifyThisNode = frmCGM.imgNode0505
        End Select
    Case Is = 6
        Select Case ColValue
        Case Is = 1
           Set IdentifyThisNode = frmCGM.imgNode0601
        Case Is = 2
           Set IdentifyThisNode = frmCGM.imgNode0602
        Case Is = 3
           Set IdentifyThisNode = frmCGM.imgNode0603
        Case Is = 4
           Set IdentifyThisNode = frmCGM.imgNode0604
        Case Is = 5
           Set IdentifyThisNode = frmCGM.imgNode0605
        End Select
    End Select
End Function
Private Sub LogLineCordinates(ByVal FromRow As Integer, _
                              ByVal FromCol As Integer, _
                              ByVal ToRow As Integer, _
                              ByVal ToCol As Integer)

' log line cordinates in an array so that lines could be drewn later

    Dim tmpX2, tmpY2 As Integer

    Select Case FromCol
        Case Is = ToCol
            tmpX2 = (ToCol - 1) * 800 + 240
            tmpY2 = (ToRow - 1) * 800
        Case Is < ToCol
            tmpX2 = (ToCol - 1) * 800 + 50
            tmpY2 = (ToRow - 1) * 800 + 50
        Case Is > ToCol
            tmpX2 = (ToCol - 1) * 800 + 480 - 50
            tmpY2 = (ToRow - 1) * 800 + 50
    End Select

    LineNumber = LineNumber + 1
    With CGMLineArray(LineNumber)
        Select Case FromCol
            Case Is = ToCol
                .X1 = (FromCol - 1) * 800 + 240
                .Y1 = (FromRow - 1) * 800 + 480
            Case Is < ToCol
                .X1 = (FromCol - 1) * 800 + 480 - 50
                .Y1 = (FromRow - 1) * 800 + 480 - 50
            Case Is > ToCol
                .X1 = (FromCol - 1) * 800 + 50
                .Y1 = (FromRow - 1) * 800 + 480 - 50
        End Select
        .X2 = tmpX2
        .Y2 = tmpY2
    End With

    ' log the line left arrow
    LineNumber = LineNumber + 1
    With CGMLineArray(LineNumber)
        .X2 = tmpX2
        .Y2 = tmpY2
        Select Case FromCol
            Case Is = ToCol
                .X1 = .X2 + 100
                .Y1 = .Y2 - 100
            Case Is < ToCol
                .X1 = .X2
                .Y1 = .Y2 - 144
            Case Is > ToCol
                .X1 = .X2 + 144
                .Y1 = .Y2
        End Select
    End With

    ' log the line right arrow
    LineNumber = LineNumber + 1
    With CGMLineArray(LineNumber)
        .X2 = tmpX2
        .Y2 = tmpY2
        Select Case FromCol
            Case Is = ToCol
                .X1 = .X2 - 100
                .Y1 = .Y2 - 100
            Case Is < ToCol
                .X1 = .X2 - 144
                .Y1 = .Y2
            Case Is > ToCol
                .X1 = .X2
                .Y1 = .Y2 - 144
        End Select
    End With

End Sub
```

FrmFAQ - F.A.Q. panel

```
'----------------------------------------------------------------------
' Form Name:   frmFAQ.doc
' Description: This form is used to display the Frequency Asked Questions
'----------------------------------------------------------------------

Private Sub cmdFAQHelp_Click()

    Dim tmpMessage As String

    tmpMessage = Chr(13) & Chr(10) & Chr(13) & Chr(10) _
               & "These are common questions" & Chr(13) & Chr(10) _
               & "If you could not find the answers to your questions," _
               & Chr(13) & Chr(10) _
               & "please contact your Professor"

    frmMessage.Tag = "FAQ"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmFAQ.Hide

End Sub


Private Sub cmdQuit_Click()

    End

End Sub


Private Sub cmdReturn_Click()

    frmCGM.Show
    frmFAQ.Hide

End Sub


Private Sub Form_Load()

    ' Center the form
    Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height - 480) / 2

    Dim DisplayFAQ$
    faqFullName = faqFilePath & faqFileName & faqFileExtn
    Open faqFullName For Binary As #1
    DisplayFAQ$ = Space$(LOF(1))
    Get #1, , DisplayFAQ$
    Close #1
    rtfFAQ.Text = DisplayFAQ$

End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmCGM.Show
End Sub
```

FrmLogin - Login the application

```
'-----------------------------------------------------------------
' Form Name:   frmLogin.frm
' Description: This form is used for checking the student's access
'-----------------------------------------------------------------

Option Explicit

' Define type student to store student information
Private Sub cmdCancel_Click()
    LoginSucceeded = False
    Unload Me
End Sub

Private Sub cmdOK_Click()
    ' Check for correct password
    ' See workshop chapter 13, page 268 for more detail
    Dim tmpFileName As String

    txtStudentID = Format(txtStudentID, "<")    ' convert to lower case
    txtPassword = Format(txtPassword, "<")      ' convert to lower case

    Call Initialize_Path

    If Trim(txtStudentID) = "" Then
        MsgBox "Invalid StudentID, try again!", , "Login"
        txtStudentID.SetFocus
        SendKeys "{Home}+{End}"
    Else
        If (txtStudentID = "self") Then
            ' this is a new student
            ' call another form to handle input
            frmAddStudent.Show
        Else
            stuFileName = App.Path & "\" & txtStudentID & stuFileExtn
            ' try to find out if file exist
            tmpFileName = Dir(stuFileName)
            If tmpFileName = "" Then
                MsgBox "Invalid StudentID, try again!", , "Login"
                txtStudentID.SetFocus
                SendKeys "{Home}+{End}"
            Else
                Call GetStudentInfo
                If Trim(txtPassword) = Trim(CurrStudent.Password) Then
                    LoginSucceeded = True
                    'Call Main
                    Unload frmLogin
                    frmModule.Show
                Else
                    MsgBox "Invalid Password, try again!", , "Login"
                    txtPassword.SetFocus
                    SendKeys "{Home}+{End}"
                End If
            End If
        End If
    End If

End Sub

Private Sub Form_Load()
    ' Center the form
    Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height) / 2
End Sub
```

FrmMailList – List out the mails

```
'------------------------------------------------------------
' Form Name:   frmMailList.frm
' Description: This form is about the Mail list functions
'------------------------------------------------------------

' Module variable to hold MouseDown position information.
Dim ListX, ListY

Private Sub Form_Load()

    ' Resize the form.
    Height = 3945
    Call Tools_Resize

     ' Set list box headings.
     a$ = Mid$(Format$("From", "!" + String$(25, "@")), 1, 25)
     b$ = Mid$(Format$("Subject", "!" + String$(35, "@")), 1, 35)
     c$ = "Date"
     Headings = a$ + b$ + c$
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)
    ' If the user is closing the application, let this form unload.
    If UnloadMode = 4 Then
        ' Unloading is permitted.
    Else
        ' If the user is still logged on, minimize the form rather than closing it.
        If frmVBMail.MapiMess.SessionID <> 0 Then
            Me.WindowState = 1
            Cancel = True
        End If
    End If
End Sub

Private Sub Form_Resize()
    ' If the form isn't minimized, resize the list box to fit the form.
    If WindowState <> 1 Then

        If frmVBMail.DispTools.Checked Then
            xHeight% = Tools.Height
        Else
            xHeight% = 0
        End If

        ' Check for the minimum form height.
        If Height < 2500 - xHeight% Then
            Height = 2500
            Exit Sub
        End If
        MList.Width = ScaleWidth - MList.Left - 90
        MList.Height = ScaleHeight - 90 - MList.Top - xHeight%
    End If
End Sub

Private Sub MList_Click()
' Set the message index and enable the
' Previous and Next buttons as needed.
    Select Case MList.ListIndex
        Case 0
            frmVBMail.Previous.Enabled = False
        Case MList.ListCount - 1
            frmVBMail.Next.Enabled = False
        Case Else
            frmVBMail.Previous.Enabled = True
            frmVBMail.Next.Enabled = True
    End Select
    frmVBMail.MapiMess.MsgIndex = MList.ListIndex
End Sub

Private Sub MList_DBLClick()
' Check to see if the message is currently viewed,
' and if it isn't, load it into a new form.
    If Not frmMailList.MList.ItemData(frmMailList.MList.ListIndex) Then
       Dim Msg As New frmMailMessage
       Call LoadMessage(frmMailList.MList.ListIndex, Msg)
       frmMailList.MList.ItemData(frmMailList.MList.ListIndex) = True
    Else
        ' Search through the active windows to
        ' find the window with the correct message to view.
        For i = 0 To Forms.Count - 1
            If TypeOf Forms(i) Is frmMailMessage Then
                If Val(Forms(i).Tag) = frmMailList.MList.ListIndex Then
                    Forms(i).Show
                    Exit Sub
                End If
            End If
        Next i
     End If
End Sub

Private Sub MList_KeyPress(KeyAscii As Integer)
    ' If the user presses ENTER, process the action as a DblClick event.
    If KeyAscii = 13 Then
        Call MList_DBLClick
    End If
End Sub

Private Sub MList_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Save the X and Y positions to determine the start of the drag-and-drop action.
    ListX = X
    ListY = Y
End Sub
```

```
Private Sub MList_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' If the mouse button is down and the X,Y position has changed, start dragging.
    If Button = 1 And ((X <> ListX) Or (Y <> ListY)) Then
        MList.Drag 1
    End If
End Sub

Private Sub PrtImage_DragDrop(Source As Control, X As Single, Y As Single)
    ' Same as File.PrintMessage on the frmVBMail File menu.
    Call PrintMail
End Sub

Private Sub Tools_Resize()
    ' Adjust the width of the lines on the top of the toolbar.
    Line1(0).X2 = Tools.Width
    Line1(1).X2 = Tools.Width
    Tools.Refresh
End Sub

Private Sub Trash_DragDrop(Source As Control, X As Single, Y As Single)
    ' Delete a message (Delete Button or Edit.Delete).
   Call DeleteMessage
End Sub
```

FrmMailMessage - Mail messages

```
'-----------------------------------------------------------------
' Form Name:   frmMailMessage.frm
' Description: This form is used for mail messages information
'-----------------------------------------------------------------

Private Sub aList_DblClick()
    ' ListIndex is the index into the attachment list.
    frmVBMail.MapiMess.AttachmentIndex = aList.ListIndex
    If frmVBMail.MapiMess.AttachmentType = vbAttachTypeData Then
        Call DisplayAttachedFile(frmVBMail.MapiMess.AttachmentPathName)
    Else
        MsgBox " This sample application doesn't view OLE-type attachments"
    End If
End Sub

Private Sub AttachWin_Resize()
    '  Update the widths of the fields and adjust the line
    '  controls as needed.
    aList.Width = AttachWin.Width - aList.Left - 315
End Sub

Private Sub Form_Activate()
    ' When the form is activated, update frmMailList.MList
    ' to reflect the current item. The Tag property contains
    ' the index of the currently viewed message.
    frmMailList.MList.ListIndex = Val(Me.Tag)
    frmMailList.MList.ItemData(Val(Me.Tag)) = True
    frmVBMail.MapiMess.MsgIndex = Val(Me.Tag)
End Sub

Private Sub Form_Load()
    ' Ensure all resizing is done on startup.
    Call Picture1_Resize
    Call AttachWin_Resize
    Call Form_Resize
End Sub

Private Sub Form_Resize()
    ' Adjust the window size if the form isn't minimized.
    Call SizeMessageWindow(Me)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    ' Tag is set to -1 after the currently viewed message
    ' is deleted.
    If Val(Me.Tag) > 0 Then
        frmMailList.MList.ItemData(Val(Me.Tag)) = False
    End If
End Sub
```

```
Private Sub Picture1_Resize()
    '  Update the widths of the fields and adjust the line
    '  controls as needed.
    TopLine.X2 = Picture1.Width - 90
    BottomLine.X2 = Picture1.Width - 90
    RightLine.X1 = Picture1.Width - 90
    RightLine.X2 = Picture1.Width - 90
    lf% = txtTo.Left
    txtTo.Width = Picture1.Width - 120 - lf%
    txtDate.Width = Picture1.Width - 120 - lf%
    txtcc.Width = Picture1.Width - 120 - lf%
    txtsubject.Width = Picture1.Width - 120 - lf%
    txtFrom.Width = Picture1.Width - 120 - lf%
    Picture1.Refresh
End Sub

Private Sub txtcc_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtDate_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtFrom_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtNoteText_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtsubject_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtTo_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub
```

FrmMailOption – Mail available option

```
'-------------------------------------------------------------------
' Form Name:   frmMailOption.frm
' Description: This form is used for choosing the mail options
'-------------------------------------------------------------------

Private Sub aList_DblClick()
    ' ListIndex is the index into the attachment list.
    frmVBMail.MapiMess.AttachmentIndex = aList.ListIndex
    If frmVBMail.MapiMess.AttachmentType = vbAttachTypeData Then
        Call DisplayAttachedFile(frmVBMail.MapiMess.AttachmentPathName)
    Else
        MsgBox " This sample application doesn't view OLE-type attachments"
    End If
End Sub

Private Sub AttachWin_Resize()
    '  Update the widths of the fields and adjust the line
    '  controls as needed.
    aList.Width = AttachWin.Width - aList.Left - 315
End Sub

Private Sub Form_Activate()
    ' When the form is activated, update frmMailList.MList
    ' to reflect the current item. The Tag property contains
    ' the index of the currently viewed message.
    frmMailList.MList.ListIndex = Val(Me.Tag)
    frmMailList.MList.ItemData(Val(Me.Tag)) = True
    frmVBMail.MapiMess.MsgIndex = Val(Me.Tag)
End Sub

Private Sub Form_Load()
    ' Ensure all resizing is done on startup.
    Call Picture1_Resize
    Call AttachWin_Resize
    Call Form_Resize
End Sub

Private Sub Form_Resize()
    ' Adjust the window size if the form isn't minimized.
    Call SizeMessageWindow(Me)
End Sub

Private Sub Form_Unload(Cancel As Integer)
    ' Tag is set to -1 after the currently viewed message
    ' is deleted.
    If Val(Me.Tag) > 0 Then
        frmMailList.MList.ItemData(Val(Me.Tag)) = False
    End If
End Sub
```

```
Private Sub Picture1_Resize()
    '  Update the widths of the fields and adjust the line
    '  controls as needed.
    TopLine.X2 = Picture1.Width - 90
    BottomLine.X2 = Picture1.Width - 90
    RightLine.X1 = Picture1.Width - 90
    RightLine.X2 = Picture1.Width - 90
    lf% = txtTo.Left
    txtTo.Width = Picture1.Width - 120 - lf%
    txtDate.Width = Picture1.Width - 120 - lf%
    txtcc.Width = Picture1.Width - 120 - lf%
    txtsubject.Width = Picture1.Width - 120 - lf%
    txtFrom.Width = Picture1.Width - 120 - lf%
    Picture1.Refresh
End Sub

Private Sub txtcc_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtDate_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtFrom_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtNoteText_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtsubject_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub

Private Sub txtTo_KeyPress(KeyAscii As Integer)
    ' Ignore all keystrokes.
    KeyAscii = 0
End Sub
```

FrmMessage - Error/Warning message

```
' -----------------------------------------------
' Form Name..: frmMessage
' Description: Being called by many other forms
'              to display prompt message to the
'       student
' -----------------------------------------------

Option Explicit
Private Sub cmdOK_Click()

    frmMessage.Hide

    Select Case frmMessage.Tag
    Case Is = "CGM"
        frmCGM.Show
    Case Is = "FAQ"
        frmFAQ.Show
    Case Is = "MOD"
        frmModule.Show
    Case Is = "MAIL"
        frmVBMail.Show
    Case Is = "QUIZ"
        frmQuiz.Show
    Case Is = "END"
        End
    End Select

End Sub
Private Sub Form_Load()
    Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height) / 2
End Sub
```

FrmModule – Module control panel

```
' -----------------------------------------------------------
' Form Name: frmModule.frm
' Description: Main module allowing student select a module
' -----------------------------------------------------------

Option Explicit

Public sveModuleNdx As Integer
Public allModuleMax As Integer

Private Sub cmdHelp_Click()

    Dim tmpMessage As String
    Dim tmpTit As String

    tmpMessage = Chr(13) & Chr(10) & Chr(13) & Chr(10) _
               & "Select a module you wish to learn" & Chr(13) & Chr(10) _
               & "Click <start> to start the module" & Chr(13) & Chr(10) _
               & "      <intro> to have a brief introduction"

    frmMessage.Tag = "   "
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show

End Sub
Private Sub cmdModule_Click(Index As Integer)

' -----------------------------------------------------------
' A module is selected -> pop up start and intro buttons
' -----------------------------------------------------------

    lblStart.Visible = True

    cmdStart(sveModuleNdx).Visible = False
    cmdIntroduction(sveModuleNdx).Visible = False
    sveModuleNdx = Index

    cmdStart(Index).Visible = True
    cmdIntroduction(Index).Visible = True

    sbrModule.Panels(1).Text = CGMModuleInfo(Index).CGMModuleSDesc _
                         & " - " _
                         & CGMModuleInfo(Index).CGMModuleLDesc

End Sub

Private Sub cmdProgress_Click()
```

```
    Dim tmpMessage As String
    Dim i As Integer

    ' loop through the database to get the progress
    tmpMessage = "The progress of " & Trim(CurrStudent.FirstName) & " " &
  Trim(CurrStudent.LastName) & ":" & Chr(13) & Chr(10) _
               & "Module    Concept     Yours  Class  # Std." & Chr(13) & Chr(10) _
               & "------    -------     -----  -----  ------"

    ' get progress info
    Call SelectProgress(CurrStudent.StudentID)

    For i = 1 To ProgressCnt
         tmpMessage = tmpMessage _
                    & Chr(13) & Chr(10) _
                    & StudentProgress(i).ModuleName _
                    & StudentProgress(i).NodeName _
                    & StudentProgress(i).StudentGrade _
                    & StudentProgress(i).GroupGrade _
                    & StudentProgress(i).GroupCount

    Next i

    frmMessage.Tag = "   "
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show

End Sub
Private Sub cmdQuit_Click()
    End
End Sub
Private Sub cmdStart_Click(Index As Integer)

        ' for the time being, actiavte only if it's module 2
        mciControl.Command = "stop"
        InitCGMNode = True
        CurrStudent.Module = CGMModuleInfo(Index).CGMModuleName
        frmCGM.Caption = "Concept Graph Model " _
                       & " - " _
                       & CGMModuleInfo(Index).CGMModuleName _
                       & " (" _
                       & CGMModuleInfo(Index).CGMModuleLDesc _
                       & ")"

        frmCGM.Show
        Unload Me

End Sub

Private Sub cmdIntroduction_Click(Index As Integer)

    ' play video
    aviFilename = CGMModuleInfo(Index).ModuleAviFName
    wavFileName = CGMModuleInfo(Index).ModuleWavFName
    If aviFilename <> "NA" And aviFilename <> " " Then
        ' if avi is available, play avi file 1st
        aviFullName = aviFilePath & aviFilename & aviFileExtn
        mciControl.FileName = aviFullName
        fraVideo.Visible = True
        mciControl.Visible = True
        mciControl.Command = "play"
    Else
        If wavFileName <> "NA" And wavFileName <> " " Then
            wavFullName = wavFilePath & wavFileName & wavFileExtn
            mciControl.FileName = wavFullName
            mciControl.Command = "play"
        End If
    End If

End Sub

Private Sub Form_Load()

    Dim i As Integer

    'center the form
    Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height) / 2

    ' get info from database
    Call Select_Module_Info

    For i = 1 To allModuleMax
        cmdModule(i).Caption = CGMModuleInfo(i).CGMModuleSDesc
        cmdStart(i).Visible = False
        cmdIntroduction(i).Caption = "Intro to " & CGMModuleInfo(i).CGMModuleSDesc
        cmdIntroduction(i).Visible = False
    Next i

    sveModuleNdx = 2

    lblModule.Visible = True
    lblStart.Visible = False
    fraVideo.Visible = False
    mciControl.Visible = False

End Sub
Private Sub mciControl_ModeChange(Mode As String)

    If Mode = "stopped" Then
        ' if there are any quizzes, popup quiz pane
        Unload Me
        InitCGMNode = True
        frmCGM.Show
    End If

End Sub
Sub Select_Module_Info()
```

```
' ----------------------------------------------------
' *) Get all available modules from database and store
'    them in recordset array ALLMODULERST
' *) Loop through ALLMODULERST and load module info into array
' ----------------------------------------------------

    Dim AllModuleDbs As Database, AllModuleRst As Recordset

    Dim tmpSQL As String
    Dim i As Integer

    On Error GoTo ErrorHandler1
    Set AllModuleDbs = OpenDatabase(CGMDatabase)

    tmpSQL = " SELECT C.CGM_Module_Name," _
           & " C.CGM_Module_SDesc," _
           & " C.CGM_Module_LDesc," _
           & " C.Module_Avi_FName," _
           & " C.Module_Wav_FName" _
           & " FROM CGM_Module_Info AS C;" _

    Set AllModuleRst = AllModuleDbs.OpenRecordset(tmpSQL)

    On Error GoTo ErrorHandler2
    ' go to last record to count
    AllModuleRst.MoveLast
    allModuleMax = AllModuleRst.RecordCount

    If allModuleMax > 8 Then
        allModuleMax = 8
    End If

    ReDim CGMModuleInfo(allModuleMax)

    ' go to 1st row in recordset
    AllModuleRst.MoveFirst
    ' Loop thru recordset to get module names and attributes etc...
    For i = 1 To allModuleMax
        With CGMModuleInfo(i)
            .CGMModuleName = AllModuleRst.Fields(0)
            .CGMModuleSDesc = AllModuleRst.Fields(1)
            .CGMModuleLDesc = AllModuleRst.Fields(2)
            .ModuleAviFName = AllModuleRst.Fields(3)
            .ModuleWavFName = AllModuleRst.Fields(4)
        End With
         ' advance to next row
        AllModuleRst.MoveNext
    Next i

    AllModuleDbs.Close

    Exit Sub

' Handle fatal error => stop apps
ErrorHandler1:
    Dim tmpMessage As String
    tmpMessage = "Fatal Error - Authoring Database was not found. " _
              & "Please contact your system administration"

    frmMessage.Tag = "END"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmModule.Hide

    Exit Sub

ErrorHandler2:

    tmpMessage = "Fatal Error - No information was found. " _
              & "Please contact your system administration"

    frmMessage.Tag = "END"
    frmMessage.rtfMessage = tmpMessage
    frmMessage.Show
    frmModule.Hide

    Exit Sub

End Sub
```

FrmQuiz – Display quiz

```
' -----------------------------------------------
' Form Name: frmQuiz.frm
' Description: Handle the quiz by:
'      display question in the text box
'      display answers in radio buttons
'      accept input from student
'      evaluate the answer
'      accunmulate the mark
'      update grade table
' -----------------------------------------------

Private QuizResponse As Integer

Private Sub cmdCancel_Click()

    InitCGMNode = True
    Unload Me
    frmCGM.Show

End Sub

Private Sub cmdOK_Click()

    Dim tmpMessage As String
    frmMessage.Tag = "QUIZ"
    ' verify if an answer is check if NOT generate error
    If QuizResponse = 0 Then
        ' generate error
        Beep
        tmpMessage = Chr(13) & Chr(10) & Chr(13) & Chr(10) _
                   & "Please enter an answer, try again!" _
                   & Chr(13) & Chr(10)
        frmMessage.rtfMessage = tmpMessage
        frmMessage.Show
    Else
        ' calculate mark
        If QuizResponse = CGMQuizInfo(QuizCntlNbr).CGMCorrectAnswer Then
            CurrStudent.AccumGrade = CurrStudent.AccumGrade _
                                   + CGMQuizInfo(QuizCntlNbr).CGMQuizGrade
            tmpMessage = Chr(13) & Chr(10) & Chr(13) & Chr(10) _
                       & "Congratulation! Your answer is correct"
        Else
            tmpMessage = Chr(13) & Chr(10) & Chr(13) & Chr(10) _
                       & "Sorry! Your answer is not correct" & Chr(13) & Chr(10) _
                       & "Reason: " & Chr(13) & Chr(10) _
                       & CGMQuizInfo(QuizCntlNbr).CGMAnsDesc
        End If
        frmMessage.Tag = "QUIZ"
        frmMessage.rtfMessage = tmpMessage
        frmMessage.Show
        If QuizCntlNbr < QuizCntlMax Then
            QuizCntlNbr = QuizCntlNbr + 1
            Call LoadOneQuiz(QuizCntlNbr)
        Else
            Call UpdateCurrNode(CurrStudent.Module, _
                                CurNodeName, _
                                CurrStudent.StudentID, _
                                CurrStudent.AccumGrade)
            tmpMessage = "You have finished your quiz and your grade is " _
                       & CurrStudent.AccumGrade
            ' prompt student indicating quiz has been finished
            InitCGMNode = True
            frmMessage.Tag = "CGM"
            frmMessage.rtfMessage = tmpMessage
            frmMessage.Show
            Unload Me
        End If
   End If

End Sub

Private Sub Form_Load()

' Quiz is started
' load question base on quiz name

    'center the form
    Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height) / 2

    Call LoadOneQuiz(QuizCntlNbr)

End Sub

Private Sub LoadOneQuiz(ByVal QuizCntlNbr As Integer)

    qizFileName = CGMQuizInfo(QuizCntlNbr).CGMQuizFName

    Dim DisplayQuestion$, DisplayAnswer$

    ' get question
    qizFullName = qizFilePath & qizFileName & qizFileExtn
    Open qizFullName For Binary As #1
    DisplayQuestion$ = Space$(LOF(1))
    Get #1, , DisplayQuestion$
    Close #1
    rtfQuestion.Text = DisplayQuestion$  'display file
    rtfQuestion.Enabled = True
    ' get proposed answer 1
    If CGMQuizInfo(QuizCntlNbr).CGMAns1FName <> "NA" Then
        qizFileName = CGMQuizInfo(QuizCntlNbr).CGMAns1FName
        qizFullName = qizFilePath & qizFileName & qizFileExtn
        Open qizFullName For Binary As #1
        DisplayAnswer$ = Space$(LOF(1))
        Get #1, , DisplayAnswer$
        Close #1
        optAnswer(1).Caption = DisplayAnswer$
        optAnswer(1).Visible = True
    End If
    ' get proposed answer 2
    If CGMQuizInfo(QuizCntlNbr).CGMAns2FName <> "NA" Then
        qizFileName = CGMQuizInfo(QuizCntlNbr).CGMAns2FName
        qizFullName = qizFilePath & qizFileName & qizFileExtn
        Open qizFullName For Binary As #1
        DisplayAnswer$ = Space$(LOF(1))
        Get #1, , DisplayAnswer$
        Close #1
        optAnswer(2).Caption = DisplayAnswer$
        optAnswer(2).Visible = True
    End If
    ' get proposed answer 3
    If CGMQuizInfo(QuizCntlNbr).CGMAns3FName <> "NA" Then
        qizFileName = CGMQuizInfo(QuizCntlNbr).CGMAns3FName
        qizFullName = qizFilePath & qizFileName & qizFileExtn
        Open qizFullName For Binary As #1
        DisplayAnswer$ = Space$(LOF(1))
        Get #1, , DisplayAnswer$
        Close #1
        optAnswer(3).Caption = DisplayAnswer$
        optAnswer(3).Visible = True
    End If
    ' get proposed answer 4
    If CGMQuizInfo(QuizCntlNbr).CGMAns4FName <> "NA" Then
        qizFileName = CGMQuizInfo(QuizCntlNbr).CGMAns4FName
        qizFullName = qizFilePath & qizFileName & qizFileExtn
        Open qizFullName For Binary As #1
        DisplayAnswer$ = Space$(LOF(1))
        Get #1, , DisplayAnswer$
        Close #1
        optAnswer(4).Caption = DisplayAnswer$
        optAnswer(4).Visible = True
    End If

End Sub

Private Sub optAnswer_Click(Index As Integer)
    QuizResponse = Index
End Sub
```

FrmSendNote - Send out a message

```
' -------------------------------------------------------
' Form Name..: frmSendNote
' Description: Handle the send message to the professor
' -------------------------------------------------------

Private Sub Attach_Click()
' Handle attachments.
On Error Resume Next
   frmVBMail.CMDialog1.DialogTitle = "Attach"
   frmVBMail.CMDialog1.Filter = "All Files(*.*)|*.*|Text Files(*.txt)|*.txt"
   frmVBMail.CMDialog1.ShowOpen
   If Err = 0 Then
        On Error GoTo 0
        frmVBMail.MapiMess.AttachmentIndex = frmVBMail.MapiMess.AttachmentCount
        frmVBMail.MapiMess.AttachmentName = frmVBMail.CMDialog1.FileTitle
        frmVBMail.MapiMess.AttachmentPathName = frmVBMail.CMDialog1.FileName
        frmVBMail.MapiMess.AttachmentPosition = frmVBMail.MapiMess.AttachmentIndex
        frmVBMail.MapiMess.AttachmentType = vbAttachTypeData
   End If
End Sub

Private Sub ChkNames_Click()
    ' Resolve the names.
    Call CopyNamestoMsgBuffer(Me, True)
    Call UpdateRecips(Me)
End Sub

Private Sub CompAdd_Click()
    ' Display the address book and update upon return.
    Call CopyNamestoMsgBuffer(Me, False)
    frmVBMail.MapiMess.Action = vbMessageShowAdBook
    Call UpdateRecips(Me)
End Sub

Private Sub CompOpt_Click()
    ' Display the Message Option form.
    OptionType = conOptionMessage
    MailOptFrm.Show 1
End Sub

Private Sub Form_Activate()
    ' Set the MessageIndex to -1 (Compose Buffer) when this window is activated.
    frmVBMail.MapiMess.MsgIndex = -1
End Sub

Private Sub Form_Load()
    ' Ensure the windows are sized as needed.
    Call Picture1_Resize
    Call Picture2_Resize
    Call Form_Resize
End Sub

Private Sub Form_Resize()
    ' Adjust the window sizes if the form isn't minimized.
    If WindowState <> 1 Then
        If ScaleHeight > txtNoteText.Top Then
            txtNoteText.Height = ScaleHeight - txtNoteText.Top
            txtNoteText.Width = ScaleWidth
        End If
    End If
End Sub

Private Sub Picture1_Resize()
    ' Update the widths of the fields and adjust the line
    ' controls as needed.
    TopLine(0).X2 = Picture1.Width
    TopLine(1).X2 = Picture1.Width
    Picture1.Refresh
End Sub

Private Sub Picture2_Resize()
    ' Update the widths of the fields and adjust the line
    ' controls as needed.
    TopLine2.X2 = Picture2.Width
    Picture2.Refresh
End Sub

Private Sub Send_Click()
    ' Place the Subject and Note text into the buffer.
    ' Add room in the beginning for attachment files.
    If frmVBMail.MapiMess.AttachmentCount > 0 Then
        txtNoteText = String$(frmVBMail.MapiMess.AttachmentCount, "*") + txtNoteText
    End If
    frmVBMail.MapiMess.MsgSubject = txtsubject
    frmVBMail.MapiMess.MsgNoteText = txtNoteText
    frmVBMail.MapiMess.MsgReceiptRequested = ReturnRequest
    Call CopyNamestoMsgBuffer(Me, True)

    On Error Resume Next
    frmVBMail.MapiMess.Action = vbMessageSend
    If Err Then
        MsgBox "An error occurred during a send: " + Str$(Err)
    Else
        Unload Me
    End If
End Sub
```

FrmSplash – Splash screen

```
' ---------------------------------------------------
' Form Name. .: frmSplah
' Description: Splash the screen when the load is slow
' ---------------------------------------------------

Option Explicit

Private Sub Form_KeyPress(KeyAscii As Integer)
    Unload Me
End Sub

Private Sub Form_Load()
    'center the form
    Me.Move (Screen.Width - Me.Width) / 2, _
            (Screen.Height - Me.Height) / 2

    lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "." & App.Revision
    lblProductName.Caption = App.Title
End Sub

Private Sub Frame1_Click()
    Unload Me
End Sub
```

FrmVBMail – Mail main control panel

```
' --------------------------------------------
' FormName. .: frmVBMail
' Description: Handle the mail sub-system
' --------------------------------------------

Private Sub About_Click()
    Dim tmpMsg As String
    Dim tmpTit As String

    frmMessage.rtfMessage = Chr(13) & Chr(10) & Chr(13) & Chr(10) _
                          & "Make sure you already logged on your" & Chr(13) & Chr(10) _
                          & "mail box prior to compose a message"

    frmMessage.Show
End Sub

Private Sub cmdExit_Click()
    ' log user off
    Call Exit_Click

End Sub

Private Sub Delete_Click()
' Delete a mail message.

    ' View all selected messages that are deleted.
    If TypeOf frmVBMail.ActiveForm Is frmMailMessage Then
        Call DeleteMessage
    ElseIf TypeOf frmVBMail.ActiveForm Is frmMailList Then
        ' Delete multiple selection.
        frmVBMail.MapiMess.MsgIndex = frmMailList.MList.ListIndex
        Call DeleteMessage
    End If

End Sub

Private Sub DispTools_Click()

    DispTools.Checked = Not DispTools.Checked
    frmMailList.Tools.Visible = DispTools.Checked


    If frmMailList.Tools.Visible Then
        Factor = 1
        ToolsSize% = -frmMailList.Tools.Height
    Else
        Factor = -1
        ToolsSize% = 0
    End If
    Select Case frmMailList.WindowState
        Case 0 ' Change the size of the form to reflect the addition/deletion of atoolbar.
            frmMailList.Height = frmMailList.Height + (Factor * frmMailList.Tools.Height)
        Case 2 ' If maximized, adjust the size of the list box.
            frmMailList.MList.Height = ScaleHeight - 90 - frmMailList.MList.Top _
                                     + ToolsSize%
    End Select
End Sub

Private Sub EditDelete_Click()
' Delete the items in the list.
On Error GoTo Trap
    If TypeOf frmVBMail.ActiveForm Is frmMailList Then
        Call Delete_Click
    End If
    Exit Sub

Trap:
    ' If an error occurs, there is probably no active form.
    ' Exit the Sub procedure.
    Exit Sub
End Sub

Private Sub Exit_Click()

    Unload Me

End Sub


Private Sub FontPrt_Click()
    ' Set the printer fonts.
    On Error Resume Next
    CMDialog1.Flags = 2
    CMDialog1.FontName = Printer.FontName
    CMDialog1.FontSize = Printer.FontSize
    CMDialog1.FontBold = Printer.FontBold
    CMDialog1.FontItalic = Printer.FontItalic
    CMDialog1.ShowFont
    If Err = 0 Then
        Printer.FontName = CMDialog1.FontName
        Printer.FontSize = CMDialog1.FontSize
        Printer.FontBold = CMDialog1.FontBold
        Printer.FontItalic = CMDialog1.FontItalic
    End If

End Sub

Private Sub FontScreen_Click()
    ' Set the screen fonts for the active control.
    On Error Resume Next
    CMDialog1.Flags = 1
    CMDialog1.FontName = frmVBMail.ActiveForm.ActiveControl.FontName
    CMDialog1.FontSize = frmVBMail.ActiveForm.ActiveControl.FontSize
    CMDialog1.FontBold = frmVBMail.ActiveForm.ActiveControl.FontBold
    CMDialog1.FontItalic = frmVBMail.ActiveForm.ActiveControl.FontItalic
    CMDialog1.ShowFont
    If Err = 0 Then
        frmVBMail.ActiveForm.ActiveControl.FontName = CMDialog1.FontName
        frmVBMail.ActiveForm.ActiveControl.FontSize = CMDialog1.FontSize
        frmVBMail.ActiveForm.ActiveControl.FontBold = CMDialog1.FontBold
        frmVBMail.ActiveForm.ActiveControl.FontItalic = CMDialog1.FontItalic
    End If
End Sub

Private Sub logoff_Click()
    ' Log off from the mail system.
    Call LogOffUser
End Sub

Private Sub Logon_Click()
    ' Log onto the mail system.
    On Error Resume Next
    MapiSess.Action = 1
    If Err <> 0 Then
        MsgBox "Logon Failure: " + Error$
    Else
        Screen.MousePointer = 11
        MapiMess.SessionID = MapiSess.SessionID
        ' Get the message count.
        GetMessageCount
        ' Load the mail list with envelope information.
        Screen.MousePointer = 11
        Call LoadList(MapiMess)
        Screen.MousePointer = 0
        ' Adjust the buttons as needed.
        Logon.Enabled = False
        LogOff.Enabled = True
        frmVBMail.SendCtl(vbMessageCompose).Enabled = True
        frmVBMail.SendCtl(vbMessageReplyAll).Enabled = True
        frmVBMail.SendCtl(vbMessageReply).Enabled = True
        frmVBMail.SendCtl(vbMessageForward).Enabled = True
        frmVBMail.PrintMessage.Enabled = True
        frmVBMail.DispTools.Enabled = True
        frmVBMail.rMsgList.Enabled = True
        frmVBMail.EditDelete.Enabled = True
      End If
End Sub

Private Sub MailOpts_Click()
    ' Display the Mail Options form.
    OptionType = conOptionGeneral
    frmMailOption.Show 1
End Sub

Private Sub MDIForm_Load()

    ' center the form
    Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height - 480) / 2
    ' Ensure all the controls are sized as needed.

    SendWithMapi = True
    frmVBMail.sbrVBMail.Panels(1).Text = "Off Line"
End Sub

Private Sub MDIForm_Unload(Cancel As Integer)

     ' Close the application and log off.
    If MapiSess.SessionID <> 0 Then
        Call logoff_Click
    End If

    frmCGM.Show

End Sub

Private Sub Next_Click()

    ' View the next message in the list.
    If frmMailList.MList.ListIndex <> frmMailList.MList.ListCount - 1 Then
        frmMailList.MList.ItemData(frmMailList.MList.ListIndex) = False
        frmMailList.MList.ListIndex = frmMailList.MList.ListIndex + 1
    End If

    Call ViewNextMsg

End Sub

Private Sub Previous_Click()

    ' View the previous message in the list.
    If frmMailList.MList.ListIndex <> 0 Then
        frmMailList.MList.ItemData(frmMailList.MList.ListIndex) = False
        frmMailList.MList.ListIndex = frmMailList.MList.ListIndex - 1
    End If

    Call ViewNextMsg

End Sub

Private Sub PrintMessage_Click()

    ' Print mail.
    Call PrintMail

End Sub

Private Sub PrSetup_Click()

' Call the printer setup procedure in the common dialog control.
On Error Resume Next
    CMDialog1.Flags = &H40  ' Printer setup dialog box only.
    CMDialog1.ShowPrinter
```

```
End Sub

Private Sub rMsgList_Click()

        Screen.MousePointer = 11
        GetMessageCount
        Call LoadList(MapiMess)
        Screen.MousePointer = 0

End Sub

Private Sub SendCtl_Click(Index As Integer)

    Dim NewMessage As New frmSendNote
    On Error Resume Next

    ' Index = 6: Compose New Message
    '       = 7: Reply
    '       = 8: Reply All
    '       = 9: Forward

    ' Save the header information and current note text.
    If Index > 6 Then
        ' SVNote = GetHeader(frmVBMail.MapiMess) + frmVBMail.MapiMess.MsgNoteText
        SVNote = frmVBMail.MapiMess.MsgNoteText
        SVNote = GetHeader(frmVBMail.MapiMess) + SVNote
    End If

    frmVBMail.MapiMess.Action = Index

    ' Set the new message text.
    If Index > 6 Then
        frmVBMail.MapiMess.MsgNoteText = SVNote
    End If

    If SendWithMapi Then
        frmVBMail.MapiMess.Action = vbMessageSendDlg
    Else
        Call LoadMessage(-1, NewMessage) ' Load message into frmVBMail frmSendNote window.
    End If
End Sub

Private Sub ShowAB_Click()

On Error Resume Next
    ' Show the address for the current message.
    frmVBMail.MapiMess.Action = vbMessageShowAdBook

    If Err Then
        If Err <> 32001 Then        ' User chose Cancel.
            MsgBox "Error: " + Error$ + " occurred trying to show the Address Book"
        End If
    Else
        If TypeOf frmVBMail.ActiveForm Is frmSendNote Then
            Call UpdateRecips(frmVBMail.ActiveForm)
        End If
    End If

End Sub

Private Sub wa_Click(Index As Integer)

    ' Arrange the windows as selected.
    frmVBMail.Arrange Index

End Sub
```

ModAuthoring – Authoring modules

```
' ------------------------------------------------------------------
' Module Name: modAuthoring
' Description: Common subroutine module to handle different tasks
'              For example, get module information from database or
'       find next node
' ------------------------------------------------------------------

' faq = Frequently Asked Question
' lec = Lecture
' qiz = Quiz
' avi = movie avi file
' wmf = window meta file slide type
' wav = audio wav to go with slide
'
' path of files
Public faqFilePath, lecFilePath, qizFilePath, aviFilePath, wavFilePath, bmpFilePath,
  stuFilePath, datFilePath
' file name
Public lecFileName, qizFileName, aviFilename, wavFileName, bmpFileName, stuFileName
' full file name = file path & file name
Public faqFullName, lecFullName, qizFullName, aviFullName, wavFullName, bmpFullName,
  stuFullName, datFullName
' location of database
Public CGMDatabase As String
' Node name to control the flow
Public CurNodeName, NxtNodeName, PrvNodeName
' Quiz control number to identify the current processing quiz
Public QuizCntlNbr, QuizCntlMax As Integer
Public ProgressCnt As Integer

' Current & Save image control
Public CurrCntlImg As Object
Public imgControlNode As Object

Public InitCGMNode As Boolean

' define student type to hold student info
Public Type Student
    StudentID As String * 8
    Password As String * 8
    FirstName As String * 20
    MiddleName As String * 20
    LastName As String * 20
    Module As String
    IntelligentLevel As Integer
    AccumGrade As Integer
    Comments As String * 60
End Type
```

```
Public CurrStudent As Student

Public Type CGMModule
    CGMModuleName As String
    CGMModuleSDesc As String
    CGMModuleLDesc As String
    ModuleAviFName As String
    ModuleWavFName As String
End Type

Public CGMModuleInfo() As CGMModule

Public Type CGMNode
    CGMModuleName As String
    CGMNodeName As String
    CGMNodePosX As Integer
    CGMNodePosY As Integer
    CGMNodeType As String
    CGMSiblingNode As String
    CGMShortDesc As String
    CGMLongDesc As String
    CGMAviFName As String
    CGMTxtFName As String
    CGMWavFName As String
    CGMQizCount As Integer
    CGMBmpCount As Integer
End Type

Public CGMNodeInfo As CGMNode

Public Type CGMQuiz
    CGMQuizName As String
    CGMQuizGrade As Integer
    CGMQuizFName As String
    CGMAns1FName As String
    CGMAns2FName As String
    CGMAns3FName As String
    CGMAns4FName As String
    CGMCorrectAnswer As Integer
    CGMAnsDesc As String
End Type

Public CGMQuizInfo() As CGMQuiz

Public Type CGMSlide
    CGMSlideName As String
    SlideSeqNumber As Integer
    SlideBeginTime As Integer
    SlideEndTime As Integer
    CGMSlideFName As String
End Type

Public CGMSlideInfo() As CGMSlide

Public Type StudProgress
```

```
    ModuleName As String * 10
    NodeName As String * 10
    StudentGrade As String * 7
    GroupGrade As String * 7
    GroupCount As String * 7
End Type

Public StudentProgress() As StudProgress

Public CGMGradeArray(6, 5) As Integer

Public Type CGMLine
    X1 As Integer
    Y1 As Integer
    X2 As Integer
    Y2 As Integer
End Type

Public CGMLineArray(200) As CGMLine

Public LineNumber As Integer

Public LoginSucceeded As Boolean

Public Const faqFileName = "FAQ"

Public Const faqFileExtn = ".txt"
Public Const lecFileExtn = ".txt"
Public Const qizFileExtn = ".txt"
Public Const aviFileExtn = ".avi"
Public Const wavFileExtn = ".AVI"
Public Const bmpFileExtn = ".wmf"
Public Const icoFileExtn = ".ico"
Public Const stuFileExtn = ".id"

'GetDriveType return values
Const DRIVE_REMOVABLE = 2
Const DRIVE_FIXED = 3
Const DRIVE_REMOTE = 4
Const DRIVE_CDROM = 5
Const DRIVE_RAMDISK = 6

' Identify drive type (to find CD Drive)
Declare Function GetDriveType _
Lib "kernel32" Alias "GetDriveTypeA" ( _
    ByVal nDrive As String _
) As Long
Sub Main()

    ' show the splah screen
    frmSplash.Show

    Initialize_Path

    frmCGM.Show
    Unload frmSplash

End Sub
Sub Initialize_Path()

    ' verify if the application is run on hard drive or not
    faqFullName = App.Path & "\FAQ\FAQ.txt"
    If Dir(faqFullName) <> "" Then
        ' run app on harddrive
        faqFilePath = App.Path & "\FAQ\"
        lecFilePath = App.Path & "\Lecture\"
        qizFilePath = App.Path & "\Quiz\"
        aviFilePath = App.Path & "\Video\"
        wavFilePath = App.Path & "\Audio\"
        bmpFilePath = App.Path & "\Slide\"
    Else
        ' run app on CD
        ' identify CD drive
        CDDriveName = GetCDDrive()

        If CDDriveName = "NA" Then
            ' CD Drive not found
            ' Prompt user for alternative location
            GoTo ErrorHandler1
        Else
            faqFilePath = CDDriveName & "Authoring\FAQ\"
            lecFilePath = CDDriveName & "Authoring\Lecture\"
            qizFilePath = CDDriveName & "Authoring\Quiz\"
            aviFilePath = CDDriveName & "Authoring\Video\"
            wavFilePath = CDDriveName & "Authoring\Audio\"
            bmpFilePath = CDDriveName & "Authoring\Slide\"
        End If

    End If

    faqFullName = faqFilePath & "FAQ.txt"

    On Error GoTo ErrorHandler1

    If Dir(faqFullName) = "" Then
       GoTo ErrorHandler1
    End If

    ' database must be where the app is (required update access)
    datFilePath = App.Path & "\Database\"
    CGMDatabase = App.Path & "\Database\Authoring.mdb"

    On Error GoTo ErrorHandler2

    If Dir(CGMDatabase) = "" Then
       GoTo ErrorHandler2
    End If

    Exit Sub
```

```
' Handle error => stop apps
ErrorHandler1:

    Dim ErrorMessage As String
    ErrorMessage = "Fatal Error - Application components were missing. " & _
                   "(multimedia files not found on local nor CD ROM) " & _
                   "Please contact your system administration"
    MsgBox ErrorMessage

    End
    Exit Sub

ErrorHandler2:

    ErrorMessage = "Fatal Error - Application database were not found. " & _
                   "(MS ACCESS Authoring database not found on local) " & _
                    "Please contact your system administration"
    MsgBox ErrorMessage

    End
    Exit Sub


End Sub
Function GetCDDrive()

' -----------------------------------------------------------
' Loop through all available characters to identify CD Drive
' If CD Drive is not found - Return NA
' Otherwise                - Return CD Drive (R:\)
' -----------------------------------------------------------

    Dim i, Drv, D$
    GetCDDrive = "NA"  ' intialize return value as not found
    For i = 0 To 25    ' Verify all possible drives A to Z
        D$ = Chr$(i + 65) & ":\"
        Drv = GetDriveType(D$)
        If Drv = DRIVE_CDROM Then
            GetCDDrive = D$
        End If
    Next i
End Function
Sub GetStudentInfo()

    Open stuFileName For Binary As #1
    ' read student record
    Get #1, , CurrStudent
    Close #1

End Sub
Sub PutStudentInfo()

    Open stuFileName For Binary As #1
    ' write student record
    Put #1, , CurrStudent
    Close #1

End Sub
Sub InsertIntoCurrNode(ByVal CurrStudentID As String)

' Create a new set of nodes for current student.
' logic:
'    insert into table CGM_Curr_Node all the node names available having
'    the student ID be intialized to current student ID

    Dim dbs As Database

    On Error GoTo ErrorHandler1

    Set dbs = OpenDatabase(CGMDatabase)

    ' Create a new records in the CGM_Curr_Node table
    dbs.Execute " INSERT INTO CGM_Curr_Node " _
              & " (Curr_Module_name," _
              & " Curr_Node_Name," _
              & " Curr_Student_ID," _
              & " Curr_Node_Grade) " _
              & " SELECT CGM_Module_Name," _
              & " CGM_Node_Name," _
              & " '" & CurrStudentID & "'," _
              & " 0 " _
              & " FROM CGM_Node_Info" _
              & " ORDER BY CGM_Module_Name, CGM_Node_Name;"
    dbs.Close

    Exit Sub

' Handle fatal error => stop apps
ErrorHandler1:

    Dim ErrorMessage As String
    ErrorMessage = "Fatal Error - Authoring Database was not found. " & _
                   "Please contact your system administration"
    MsgBox ErrorMessage

    End
    Exit Sub

End Sub
Sub UpdateCurrNode(ByVal CurrModuleName As String, _
                   ByVal CurrNodeName As String, _
                   ByVal CurrStudentID As String, _
                   ByVal CurrNodeGrade As Integer)

' update the grade of curr node

    On Error GoTo ErrorHandler1
    Dim dbs As Database

    Set dbs = OpenDatabase(CGMDatabase)

    ' update grade for the quiz that just completed
    dbs.Execute " UPDATE CGM_Curr_Node" _
              & " SET Curr_Node_Grade = " & CurrNodeGrade _
              & " WHERE Curr_Student_ID = '" & CurrStudentID & "'" _
              & "   AND Curr_Node_Name = '" & CurrNodeName & "'" _
              & "   AND Curr_Module_Name = '" & CurrModuleName & "';"

    dbs.Close

    Exit Sub

' Handle fatal error => stop apps
ErrorHandler1:

    Dim ErrorMessage As String
    ErrorMessage = "Fatal Error - Authoring Database was not found. " & _
                   "Please contact your system administration"
    MsgBox ErrorMessage

    End
    Exit Sub

End Sub

Function SelectNextNode(ByVal CurrModuleName As String, _
                        ByVal CurrNodeName As String, _
                        ByVal CurrStudentID As String) _
        As String

    On Error GoTo ErrorHandler1

    Dim dbs As Database, rst As Recordset
    Dim tmpSQL As String

    Set dbs = OpenDatabase(CGMDatabase)

    tmpSQL = " SELECT N.Next_Node_Name " _
           & " FROM CGM_Curr_Node AS C, " _
           & "      CGM_Next_Node AS N " _
           & " WHERE C.Curr_Student_ID = '" & CurrStudentID & "'" _
           & "   AND C.Curr_Node_Name = '" & CurrNodeName & "'" _
           & "   AND C.Curr_Module_Name = '" & CurrModuleName & "'" _
           & "   AND N.Curr_Node_Name =  C.Curr_Node_Name " _
           & "   AND N.Curr_Module_Name =  C.Curr_Module_Name " _
           & "   AND N.Curr_Node_Min <= C.Curr_Node_Grade " _
           & "   AND N.Curr_Node_Max >= C.Curr_Node_Grade;"

    Set rst = dbs.OpenRecordset(tmpSQL)

    On Error GoTo ErrorHandler2
    ' Populate the Recordset.
    rst.MoveLast

    SelectNextNode = rst.Fields(0)

    dbs.Close

    Exit Function

ErrorHandler1:

    Dim ErrorMessage As String
    ErrorMessage = "Fatal Error - Authoring Database was not found. " & _
                   "Please contact your system administration"
    MsgBox ErrorMessage

    End
    Exit Function

ErrorHandler2:

    SelectNextNode = "NA"

    dbs.Close

    Exit Function

End Function
Function SelectPrevNode(ByVal CurrModuleName As String, _
                        ByVal CurrNodeName As String, _
                        ByVal CurrStudentID As String) _
        As String

    Dim dbs As Database, rst As Recordset
    Dim tmpSQL As String

    On Error GoTo ErrorHandler1

    Set dbs = OpenDatabase(CGMDatabase)

    tmpSQL = " SELECT P.Prev_Node_Name " _
           & "  FROM CGM_Curr_Node AS C, " _
           & "       CGM_Prev_Node AS P " _
           & " WHERE C.Curr_Student_ID = '" & CurrStudentID & "'" _
           & "   AND C.Curr_Module_Name = '" & CurrModuleName & "'" _
           & "   AND P.Curr_Module_Name = '" & CurrModuleName & "'" _
           & "   AND P.Curr_Node_Name = '" & CurrNodeName & "'" _
           & "   AND P.Prev_Node_Name =  C.Curr_Node_Name " _
           & "   AND C.Curr_Node_Grade = (SELECT MAX (C2.Curr_Node_Grade) " _
                                       & "  FROM CGM_Curr_Node AS C2, " _
                                       & "       CGM_Prev_Node AS P2 " _
                                       & " WHERE C2.Curr_Student_ID = '" & CurrStudentID &
 "'"
 _
                                       & "   AND C2.Curr_Module_Name = '" & CurrModuleName
 & "'" _
```

```
                                        & "   AND P2.Curr_Module_Name = '" & CurrModuleName
  & "'" _
                                        & "   AND P2.Curr_Node_Name = '" & CurrNodeName &
  "'" _
                                        & "   AND P2.Prev_Node_Name =  C2.Curr_Node_Name);"

    Set rst = dbs.OpenRecordset(tmpSQL)

    On Error GoTo ErrorHandler2

    ' Populate the Recordset.
    rst.MoveLast

    SelectPrevNode = rst.Fields(0)

    dbs.Close

    Exit Function

ErrorHandler1:

    Dim ErrorMessage As String
    ErrorMessage = "Fatal Error - Authoring Database was not found. " & _
                    "Please contact your system administration"
    MsgBox ErrorMessage

    End
    Exit Furction

ErrorHandler2:

    SelectPrevNode = "NA"

    dbs.Close

    Exit Function

End Function

Sub SelectNodeInfo(ByVal CurrModuleName As String, _
                   ByVal CurrNodeName As String, _
                   ByRef CurrNodeInfo As CGMNode)

    Dim dbs As Database, rst As Recordset
    Dim tmpSQL As String

    On Error GoTo ErrorHandler1

    Set dbs = OpenDatabase(CGMDatabase)

    tmpSQL = " SELECT C.CGM_Module_Name ," _
           & " C.CGM_Node_Name," _
           & " C.CGM_Node_Row," _
           & " C.CGM_Node_Col," _
           & " C.CGM_Node_Type," _
```

```
            & " C.CGM_Sibling_Node," _
            & " C.CGM_Short_Desc," _
            & " C.CGM_Long_Desc," _
            & " C.CGM_Avi_FName," _
            & " C.CGM_Txt_FName," _
            & " C.CGM_Wav_FName," _
            & " C.CGM_Qiz_Count," _
            & " C.CGM_Bmp_Count" _
            & " FROM CGM_Node_Info AS C " _
            & " WHERE C.CGM_Node_Name = '" & CurrNodeName & "'" _
            & "   AND C.CGM_Module_Name = '" & CurrModuleName & "';"

    Set rst = dbs.OpenRecordset(tmpSQL)

    On Error GoTo ErrorHandler2

    ' Populate the Recordset.
    rst.MoveLast

    ' fill up returned info to calling procedure
    With CurrNodeInfo
        .CGMModuleName = rst.Fields(0)
        .CGMNodeName = rst.Fields(1)
        .CGMNodePosX = rst.Fields(2)
        .CGMNodePosY = rst.Fields(3)
        .CGMNodeType = rst.Fields(4)
        .CGMSiblingNode = rst.Fields(5)
        .CGMShortDesc = rst.Fields(6)
        .CGMLongDesc = rst.Fields(7)
        .CGMAviFName = rst.Fields(8)
        .CGMTxtFName = rst.Fields(9)
        .CGMWavFName = rst.Fields(10)
        .CGMQizCount = rst.Fields(11)
        .CGMBmpCount = rst.Fields(12)
    End With

    dbs.Close

    Exit Sub

ErrorHandler1:

    Dim ErrorMessage As String
    ErrorMessage = "Fatal Error - Authoring Database was not found. " & _
                   "Please contact your system administration"
    MsgBox ErrorMessage

    End
    Exit Sub

ErrorHandler2:

    CurrNodeInfo.CGMNodeName = "NA"

    dbs.Close
    Exit Sub

End Sub
Sub SelectProgress(ByVal CurrStudentID As String)

    Dim dbs As Database, rst As Recordset
    Dim tmpSQL As String

    On Error GoTo ErrorHandler1

    Set dbs = OpenDatabase(CGMDatabase)

    tmpSQL = " SELECT Curr_Module_Name ," _
           & " Curr_Node_Name," _
           & " Curr_Node_Grade" _
           & " FROM CGM_Curr_Node " _
           & " WHERE Curr_Student_ID = '" & CurrStudentID & "'" _
           & " ORDER BY Curr_Module_Name," _
           & "          Curr_Node_Name;"

    Set rst = dbs.OpenRecordset(tmpSQL)

    On Error GoTo ErrorHandler2

    ' Populate the Recordset.
    rst.MoveLast

    ProgressCnt = rst.RecordCount

    ReDim StudentProgress(ProgressCnt)
    ' go to 1st row in recordset
    rst.MoveFirst

    ' Loop thru recordset to get the student progress info.
    For i = 1 To ProgressCnt
        With StudentProgress(i)
            .ModuleName = rst.Fields(0)
            .NodeName = rst.Fields(1)
            RSet .StudentGrade = rst.Fields(2)
        End With
        rst.MoveNext
    Next i

    ' Get group average
    tmpSQL = " SELECT Curr_Module_Name ," _
           & "        Curr_Node_Name," _
           & " AVG   (Curr_Node_Grade)," _
           & " COUNT (Curr_Node_Grade)" _
           & " FROM CGM_Curr_Node" _
           & " GROUP BY Curr_Module_Name," _
           & "          Curr_Node_Name;"

    Set rst = dbs.OpenRecordset(tmpSQL)
    On Error GoTo ErrorHandler2

    ' Populate the Recordset.
    rst.MoveLast

    ProgressCnt = rst.RecordCount

    Dim TmpGrade As Integer

    ' go to 1st row in recordset
    rst.MoveFirst

    ' Loop thru recordset to get the student progress info.
    For i = 1 To ProgressCnt
        With StudentProgress(i)
            TmpGrade = rst.Fields(2)
            RSet .GroupGrade = TmpGrade
            RSet .GroupCount = rst.Fields(3)
        End With
        rst.MoveNext
    Next i

    dbs.Close

    Exit Sub

ErrorHandler1:

    Dim ErrorMessage As String
    ErrorMessage = "Fatal Error - Authoring Database was not found. " & _
                   "Please contact your system administration"
    MsgBox ErrorMessage

    End
    Exit Sub

ErrorHandler2:

    ErrorMessage = "Fatal Error - Student progress was not found. " & _
                   "Please contact your system administration"
    MsgBox ErrorMessage

    End
    Exit Sub

End Sub
```

ModVBMail - Mail modules

```
' ----------------------------------------------------------------
' Module Name: modVBMail
' Description: Common subroutine module to handle different tasks
'              of the mail sub-system. For example, log user off
'              the E-mail
' ----------------------------------------------------------------


Public Const conMailLongDate = 0
Public Const conMailListView = 1

Public Const conOptionGeneral = 1      ' Constant for Option Dialog Type - General Options
Public Const conOptionMessage = 2      ' Constant for Option Dialog Type - Message Options

Public Const conUnreadMessage = "*"    ' Constant for string to indicate unread message

Public Const vbRecipTypeTo = 1
Public Const vbRecipTypeCc = 2

Public Const vbMessageFetch = 1
Public Const vbMessageSendDlg = 2
Public Const vbMessageSend = 3
Public Const vbMessageSaveMsg = 4
Public Const vbMessageCopy = 5
Public Const vbMessageCompose = 6
Public Const vbMessageReply = 7
Public Const vbMessageReplyAll = 8
Public Const vbMessageForward = 9
Public Const vbMessageDelete = 10
Public Const vbMessageShowAdBook = 11
Public Const vbMessageShowDetails = 12
Public Const vbMessageResolveName = 13
Public Const vbRecipientDelete = 14
Public Const vbAttachmentDelete = 15

Public Const vbAttachTypeData = 0
Public Const vbAttachTypeEOLE = 1
Public Const vbAttachTypeSOLE = 2

Type ListDisplay
    Name As String * 20
    Subject As String * 40
    Date As String * 20
End Type

Public currentRCIndex As Integer
Public UnRead As Integer
Public SendWithMapi As Integer
Public ReturnRequest As Integer
Public OptionType As Integer

Declare Function GetProfileString _
        Lib "kernel32" _
       (ByVal lpAppName As String, _
        lpKeyName As Any, _
        ByVal lpDefault As String, _
        ByVal lpReturnedString As String, _
        ByVal nSize As Long) As Long

Sub Attachments(Msg As Form)
    ' Clear the current attachment list.
    Msg.aList.Clear

    ' If there are attachments, load them into the list box.
    If frmVBMail.MapiMess.AttachmentCount Then
        Msg.NumAtt = frmVBMail.MapiMess.AttachmentCount & " Files"
        For i% = 0 To frmVBMail.MapiMess.AttachmentCount - 1
            frmVBMail.MapiMess.AttachmentIndex = i%
            a$ = frmVBMail.MapiMess.AttachmentName
            Select Case frmVBMail.MapiMess.AttachmentType
                Case vbAttachTypeData
                    a$ = a$ + " (Data File)"
                Case vbAttachTypeEOLE
                    a$ = a$ + " (Embedded OLE Object)"
                Case vbAttachTypeSOLE
                    a$ = a$ + " (Static OLE Object)"
                Case Else
                    a$ = a$ + " (Unknown attachment type)"
            End Select
            Msg.aList.AddItem a$
        Next i%

        If Not Msg.AttachWin.Visible Then
            Msg.AttachWin.Visible = True
            Call SizeMessageWindow(Msg)
            ' If Msg.WindowState = 0 Then
            '    Msg.Height = Msg.Height + Msg.AttachWin.Height
            ' End If
        End If

    Else
        If Msg.AttachWin.Visible Then
            Msg.AttachWin.Visible = False
            Call SizeMessageWindow(Msg)
            ' If Msg.WindowState = 0 Then
            '    Msg.Height = Msg.Height - Msg.AttachWin.Height
            ' End If
        End If
    End If
    Msg.Refresh
End Sub

Sub CopyNamestoMsgBuffer(Msg As Form, fResolveNames As Integer)
```

```
    Call KillRecips(frmVBMail.MapiMess)
    Call SetRCList(Msg.txtTo, frmVBMail.MapiMess, vbRecipTypeTo, fResolveNames)
    Call SetRCList(Msg.txtcc, frmVBMail.MapiMess, vbRecipTypeCc, fResolveNames)

End Sub

Function DateFromMapiDate$(ByVal S$, wFormat%)
' -----------------------------------------------
' This procedure formats a MAPI date in one of
' two formats for viewing the message.
' -----------------------------------------------

    Y$ = Left$(S$, 4)
    M$ = Mid$(S$, 6, 2)
    D$ = Mid$(S$, 9, 2)
    T$ = Mid$(S$, 12)
    Ds# = DateValue(M$ + "/" + D$ + "/" + Y$) + TimeValue(T$)
    Select Case wFormat
        Case conMailLongDate
            f$ = "dddd, mmmm d, yyyy, h:mmAM/PM"
        Case conMailListView
            f$ = "mm/dd/yy hh:mm"
    End Select
    DateFromMapiDate = Format$(Ds#, f$)

End Function

Sub DeleteMessage()

    ' If the currently active form is a message, set MListIndex to
    ' the correct value.
    If TypeOf Screen.ActiveForm Is frmMailMessage Then
        frmMailList.MList.ListIndex = Val(Screen.ActiveForm.Tag)
        ViewingMsg = True
    End If

   ' Delete the mail message.
    If frmMailList.MList.ListIndex <> -1 Then
        frmVBMail.MapiMess.MsgIndex = frmMailList.MList.ListIndex
        frmVBMail.MapiMess.Action = vbMessageDelete
        X% = frmMailList.MList.ListIndex
        frmMailList.MList.RemoveItem X%
        If X% < frmMailList.MList.ListCount - 1 Then
            frmMailList.MList.ListIndex = X%
        Else
            frmMailList.MList.ListIndex = frmMailList.MList.ListCount - 1
        End If
        frmVBMail.sbrVBMail.Panels(1).Text = Format$(frmVBMail.MapiMess.MsgCount) _
                                            + " Messages"

        ' Adjust the index values for currently viewed messages.
        If ViewingMsg Then
            Screen.ActiveForm.Tag = Str$(-1)
        End If
        For i = 0 To Forms.Count - 1
            If TypeOf Forms(i) Is frmMailMessage Then
                If Val(Forms(i).Tag) > X% Then
                    Forms(i).Tag = Val(Forms(i).Tag) - 1
                End If
            End If
        Next i

        ' If the user is viewing a message,
        ' load the next message into the frmMailMessage form
        ' if the message isn't currently displayed.
        If ViewingMsg Then
            ' First check to see if the message is currently being viewed.
            WindowNum% = FindMsgWindow((frmMailList.MList.ListIndex))
            If WindowNum% > 0 Then
                If Forms(WindowNum%).Caption <> Screen.ActiveForm.Caption Then
                    Unload Screen.ActiveForm
                     ' Find the correct window again and display it.
                     ' The index isn't valid after the unload.
                     Forms(FindMsgWindow((frmMailList.MList.ListIndex))).Show
                Else
                     Forms(WindowNum%).Show
                End If
            Else
                Call LoadMessage(frmMailList.MList.ListIndex, Screen.ActiveForm)
            End If
        Else
            ' Check to see if there was a window viewing the message,
            ' and unload the window.
            WindowNum% = FindMsgWindow(X%)
            If WindowNum% > 0 Then
                Unload Forms(X%)
            End If
        End If
    End If
End Sub

Sub DisplayAttachedFile(ByVal FileName As String)

On Error Resume Next
    ' Determine the filename extension.
    ext$ = FileName
    junk$ = Token$(ext$, ".")
    ' Get the application from the WIN.INI file.
    Buffer$ = String$(256, " ")
    errCode% = GetProfileString("Extensions", ext$, "NOTFOUND", _
               Buffer$, Len(Left(Buffer$, Chr(0)) - 1))
    If errCode% Then
        Buffer$ = Mid$(Buffer$, 1, InStr(Buffer$, Chr(0)) - 1)
        If Buffer$ <> "NOTFOUND" Then
            ' Strip off the ^.EXT information from the string.
            EXEName$ = Token$(Buffer$, " ")
            errCode% = Shell(EXEName$ + " " + FileName, 1)
            If Err Then
                MsgBox "Error occurred during the shell: " + Error$
            End If
        Else
            MsgBox "Application that uses: <" + ext$ + "> not found in WIN.INI"
        End If
    End If
End Sub

Function FindMsgWindow(Index As Integer) As Integer
' This function searches through the active windows
' and locates those with the frmMailMessage type and then
' checks to see if the tag contains the index the user
' is searching for.

    For i = 0 To Forms.Count - 1
        If TypeOf Forms(i) Is frmMailMessage Then
            If Val(Forms(i).Tag) = Index Then
                FindMsgWindow = i
                Exit Function
            End If
        End If
    Next i

    FindMsgWindow = -1

End Function

Function GetHeader(Msg As Control) As String

Dim CR As String

    CR = Chr$(13) + Chr$(10)

    Header$ = String$(25, "-") + CR
    Header$ = Header$ + "Form: " + Msg.MsgOrigDisplayName + CR
    Header$ = Header$ + "To: " + GetRCList(Msg, vbRecipTypeTo) + CR
    Header$ = Header$ + "Cc: " + GetRCList(Msg, vbRecipTypeCc) + CR
    Header$ = Header$ + "Subject: " + Msg.MsgSubject + CR
    Header$ = Header$ + "Date: " _
                + DateFromMapiDate$(Msg.MsgDateReceived, conMailLongDate) + CR + CR
    GetHeader = Header$

End Function

Sub GetMessageCount()

    '  Reads all mail messages and displays the count.
    Screen.MousePointer = 11
    frmVBMail.MapiMess.FetchUnreadOnly = 0
    frmVBMail.MapiMess.Action = vbMessageFetch
    frmVBMail.sbrVBMail.Panels(1).Text = Format$(frmVBMail.MapiMess.MsgCount) _
                                    + " Messages"
    Screen.MousePointer = 0

End Sub
```

```
Function GetRCList(Msg As Control, RCType As Integer) As String
' Given a list of recipients, this function returns
' a list of recipients of the specified type in the
' following format:
'
'       Person 1;Person 2;Person 3

    For i = 0 To Msg.RecipCount - 1
        Msg.RecipIndex = i
        If RCType = Msg.RecipType Then
                a$ = a$ + ";" + Msg.RecipDisplayName
        End If
    Next i

    If a$ <> "" Then
       a$ = Mid$(a$, 2)  ' Strip off the leading ";".
    End If

    GetRCList = a$

End Function

Sub KillRecips(MsgControl As Control)

    ' Delete each recipient.  Loop until no recipients exist.
    While MsgControl.RecipCount
        MsgControl.Action = vbRecipientDelete
    Wend

End Sub

Sub LoadList(mailctl As Control)
' This procedure loads the mail message headers
' into the frmMailList.MList.  Unread messages have
' conUnreadMessage placed at the beginning of the string.

    frmMailList.MList.Clear
    UnRead = 0
    StartIndex = 0
    For i = 0 To mailctl.MsgCount - 1
        mailctl.MsgIndex = i
        If Not mailctl.MsgRead Then
            a$ = conUnreadMessage + " "
            If UnRead = 0 Then
                StartIndex = i  ' Start position in the mail list.
            End If
            UnRead = UnRead + 1
        Else
            a$ = "  "
        End If
        a$ = a$ + Mid$(Format$(mailctl.MsgOrigDisplayName, "!" + String$(10, "@")), 1, 10)
        If mailctl.MsgSubject <> "" Then
            b$ = Mid$(Format$(mailctl.MsgSubject, "!" + String$(35, "@")), 1, 35)
        Else
            b$ = String$(30, " ")
        End If
        c$ = Mid$(Format$(DateFromMapiDate(mailctl.MsgDateReceived, conMailListView), _
            "!" + String$(15, "@")), 1, 15)
        frmMailList.MList.AddItem a$ + Chr$(9) + b$ + Chr$(9) + c$
        frmMailList.MList.Refresh
    Next i

    frmMailList.MList.ListIndex = StartIndex

    ' Enable the correct buttons.
    frmVBMail.Next.Enabled = True
    frmVBMail.Previous.Enabled = True
    frmVBMail![Delete].Enabled = True

    ' Adjust the value of the labels displaying message counts.
    If UnRead Then
        frmVBMail.sbrVBMail.Panels(2).Text = " - " + Format$(UnRead) + " Unread"
        frmMailList.Icon = frmMailList.NewMail.Picture
    Else
        frmVBMail.sbrVBMail.Panels(2).Text = ""
        frmMailList.Icon = frmMailList.nonew.Picture
    End If

End Sub


Sub LoadMessage(ByVal Index As Integer, Msg As Form)
' This procedure loads the specified mail message into
' a form to either view or edit a message.

    If TypeOf Msg Is frmMailMessage Then
        a$ = frmMailList.MList.List(Index)
        ' Message is unread; reset the text.
        If Mid$(a$, 1, 1) = conUnreadMessage Then
            Mid$(a$, 1, 1) = " "
            frmMailList.MList.List(Index) = a$
            UnRead = UnRead - 1
            If UnRead Then
                frmVBMail.sbrVBMail.Panels(2).Text = Format$(UnRead) + " Unread"
            Else
                frmVBMail.sbrVBMail.Panels(2).Text = ""
                ' Change the icon on the list window.
                frmMailList.Icon = frmMailList.nonew.Picture
            End If
        End If
    End If

    ' These fields only apply to viewing.
    If TypeOf Msg Is frmMailMessage Then
        frmVBMail.MapiMess.MsgIndex = Index
        Msg.txtDate = DateFromMapiDate$(frmVBMail.MapiMess.MsgDateReceived, _
                      conMailLongDate)
        Msg.txtFrom = frmVBMail.MapiMess.MsgOrigDisplayName
        frmMailList.MList.ItemData(Index) = True
    End If
    ' These fields apply to both form types.
    Call Attachments(Msg)

    Msg.txtNoteText = frmVBMail.MapiMess.MsgNoteText
    Msg.txtsubject = frmVBMail.MapiMess.MsgSubject
    Msg.Caption = frmVBMail.MapiMess.MsgSubject
    Msg.Tag = Index

    Call UpdateRecips(Msg)

    Msg.Refresh
    Msg.Show

End Sub

Sub LogOffUser()
    On Error Resume Next
    frmVBMail.MapiSess.Action = 2
    If Err <> 0 Then
        MsgBox "Logoff Failure: " + ErrorR
    Else
        frmVBMail.MapiMess.SessionID = 0
        ' Adjust the menu items.
        frmVBMail.LogOff.Enabled = 0
        frmVBMail.Logon.Enabled = -1
        ' Unload all forms except the MDI form.
        i = Forms.Count - 1
        Do Until i = 1
            i = i - 1
            If TypeOf Forms(i) Is MDIForm Then
                ' Do nothing.
            Else
                Unload Forms(i)
            End If
        Loop
        ' Disable the toolbar buttons.
        frmVBMail.Next.Enabled = False
        frmVBMail.Previous.Enabled = False
        frmVBMail![Delete].Enabled = False
        frmVBMail.SendCtl(vbMessageCompose).Enabled = False
        frmVBMail.SendCtl(vbMessageReplyAll).Enabled = False
        frmVBMail.SendCtl(vbMessageReply).Enabled = False
        frmVBMail.SendCtl(vbMessageForward).Enabled = False
        frmVBMail.rMsgList.Enabled = False
        frmVBMail.PrintMessage.Enabled = False
        frmVBMail.DispTools.Enabled = False
        frmVBMail.EditDelete.Enabled = False

        ' Reset the caption for the status bar labels.
        frmVBMail.sbrVBMail.Panels(1).Text = "Off Line"
        frmVBMail.sbrVBMail.Panels(2).Text = ""
    End If

End Sub
```

```
Sub PrintLongText(ByVal LongText As String)
' This procedure prints a text stream to a printer and
' ensures that words are not split between lines and
' that they wrap as needed.

    Do Until LongText = ""
        Word$ = Token$(LongText, " ")
        If Printer.TextWidth(Word$) + Printer.CurrentX > Printer.Width _
                                 - Printer.TextWidth("ZZZZZZZZ") Then
            Printer.Print
        End If

        Printer.Print " " + Word$;
    Loop

End Sub

Sub PrintMail()
    ' In List view, all selected messages are printed.
    ' In Message view, the selected message is printed.

    If TypeOf Screen.ActiveForm Is frmMailMessage Then
        Call PrintMessage(frmVBMail.MapiMess, False)
        Printer.EndDoc
    ElseIf TypeOf Screen.ActiveForm Is frmMailList Then
        For i = 0 To frmMailList.MList.ListCount - 1
            If frmMailList.MList.Selected(i) Then
                frmVBMail.MapiMess.MsgIndex = i
                Call PrintMessage(frmVBMail.MapiMess, False)
            End If
        Next i
        Printer.EndDoc
    End If

End Sub

Sub PrintMessage(Msg As Control, fNewPage As Integer)
'   This procedure prints a mail message.

    Screen.MousePointer = 11

    ' Start a new page if needed.
    If fNewPage Then
        Printer.NewPage
    End If

    Printer.FontName = "Arial"
    Printer.FontBold = True
    Printer.DrawWidth = 10
    Printer.Line (0, Printer.CurrentY)-(Printer.Width, Printer.CurrentY)
    Printer.Print
    Printer.FontSize = 9.75
    Printer.Print "From:";
    Printer.CurrentX = Printer.TextWidth(String$(30, " "))
    Printer.Print Msg.MsgOrigDisplayName
    Printer.Print "To:";
    Printer.CurrentX = Printer.TextWidth(String$(30, " "))
    Printer.Print GetRCList(Msg, vbRecipTypeTo)
    Printer.Print "Cc:";
    Printer.CurrentX = Printer.TextWidth(String$(30, " "))
    Printer.Print GetRCList(Msg, vbRecipTypeCc)
    Printer.Print "Subject:";
    Printer.CurrentX = Printer.TextWidth(String$(30, " "))
    Printer.Print Msg.MsgSubject
    Printer.Print "Date:";
    Printer.CurrentX = Printer.TextWidth(String$(30, " "))
    Printer.Print DateFromMapiDate$(Msg.MsgDateReceived, conMailLongDate)
    Printer.Print
    Printer.DrawWidth = 5
    Printer.Line (0, Printer.CurrentY)-(Printer.Width, Printer.CurrentY)
    Printer.FontSize = 9.75
    Printer.FontBold = False
    Call PrintLongText(Msg.MsgNoteText)
    Printer.Print
    Screen.MousePointer = 0
End Sub

Sub SaveMessage(Msg As Form)

    ' Save the current subject and note text.
    ' Copy the message to the compose buffer.
    ' Reset the subject and message text.
    ' Save the message.
    svSub = Msg.txtsubject
    SVNote = Msg.txtNoteText
    frmVBMail.MapiMess.Action = vbMessageCopy
    frmVBMail.MapiMess.MsgSubject = svSub
    frmVBMail.MapiMess.MsgNoteText = SVNote
    frmVBMail.MapiMess.Action = vbMessageSaveMsg

End Sub

Sub SetRCList(ByVal NameList As String, _
                    Msg As Control, _
                    RCType As Integer, _
                    fResolveNames As Integer)
' Given a list of recipients:
'
'       Person 1;Person 2;Person 3
'
' this procedure places the names into the Msg.Recip
' structures.

    If NameList = "" Then
        Exit Sub
    End If

    i = Msg.RecipCount
    Do
```

```
        Msg.RecipIndex = i
        Msg.RecipDisplayName = Trim$(Token(NameList, ";"))
        If fResolveNames Then
            Msg.Action = vbMessageResolveName
        End If
        Msg.RecipType = RCType
        i = i + 1
    Loop Until (NameList = "")

End Sub

Sub SizeMessageWindow(MsgWindow As Form)

    If MsgWindow.WindowState <> 1 Then
        ' Determine the minimum window size based
        ' on the visiblity of AttachWin (Attachment window).
        If MsgWindow.AttachWin.Visible Then    ' Attachment window.
            MinSize = 3700
        Else
            MinSize = 3700 - MsgWindow.AttachWin.Height
        End If

        ' Maintain the minimum form size.
        If MsgWindow.Height < MinSize And (MsgWindow.WindowState = 0) Then
            MsgWindow.Height = MinSize
            Exit Sub

        End If
        ' Adjust the size of the text box.
        If MsgWindow.ScaleHeight > MsgWindow.txtNoteText.Top Then
            If MsgWindow.AttachWin.Visible Then
                X% = MsgWindow.AttachWin.Height
            Else
                X% = 0
            End If
            MsgWindow.txtNoteText.Height = MsgWindow.ScaleHeight _
                                  - MsgWindow.txtNoteText.Top - X%
            MsgWindow.txtNoteText.Width = MsgWindow.ScaleWidth
        End If
    End If

End Sub

Function Token$(tmp$, search$)

    X = InStr(1, tmp$, search$)
    If X Then
       Token$ = Mid$(tmp$, 1, X - 1)
       tmp$ = Mid$(tmp$, X + 1)
    Else
       Token$ = tmp$
       tmp$ = ""
    End If

End Function
```

```
Sub UpdateRecips(Msg As Form)
' This procedure updates the correct edit fields and the
' recipient information.

    Msg.txtTo.Text = GetRCList(frmVBMail.MapiMess, vbRecipTypeTo)
    Msg.txtcc.Text = GetRCList(frmVBMail.MapiMess, vbRecipTypeCc)

End Sub

Sub ViewNextMsg()

    ' Check to see if the message is currently loaded.
    ' If it is loaded, show that form.
    ' If it is not loaded, load the message.
    WindowNum% = FindMsgWindow((frmMailList.MList.ListIndex))
    If WindowNum% > 0 Then
        Forms(WindowNum%).Show
    Else
        If TypeOf Screen.ActiveForm Is frmMailMessage Then
            Call LoadMessage(frmMailList.MList.ListIndex, Screen.ActiveForm)
        Else
            Dim Msg As New frmMailMessage
            Call LoadMessage(frmMailList.MList.ListIndex, Msg)
        End If
    End If
End Sub
```