Project Number:

COMPUTER AIDED INSTRUCTION AS A TOOL TO TEACH PROGRAMMING

An Interactive Qualifying Project Report

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Sze-Wai Yam

Date: May 24, 2000

Approved:

Professor Glynis Hamel, Advisor

ABSTRACT

This project handles the design, implementation and evaluation of a Computer Aided Instruction (CAI) program. The project presents the background of CAI, and suggests ways to make CAI programs more effective. A survey was conducted to solicit input on the factors that contribute to an effective CAI program. By investigating various theories about learning and incorporating principles of good interface design, an effective CAI program was built using Borland C++Builder.

TABLE OF CONTENTS

ABSTRACT **2**
TABLE OF CONTENTS **3**
TABLE OF FIGURES **5**
TABLE OF TABLES **6**

1 INTRODUCTION **7**

1.1 Project Introduction 8
1.2 Project Goals 8
1.3 Project Progression 9

2 BACKGROUND **11**

2.1 What is CAI? 12
2.1.1 Tutorials 12
2.1.2 Drill and Practice Exercises 12
2.1.3 Simulations 13
2.2 History of CAI 13
2.3 Social Implications of CAI 14
2.3.1 Introduction 14
2.3.2 Advantages 15
2.3.3 Challenges 17
2.3.3.1 CAI in General 17
2.3.3.2 CAI in Classrooms 18
2.3.4 Conclusion 19

3 PROJECT DESCRIPTION **20**

3.1 Introduction 21
3.2 Psychological Basis 21
3.2.1 Introduction 21
3.2.2 Learning Theories 21
3.2.2.1 Experiential Learning 21
3.2.2.2 Behaviorism 23
3.2.2.3 Neuroscience 24
3.2.2.4 Learning Styles 24
3.3 Survey on CAI Program Design 25
3.3.1 Description 25
3.3.2 Results 28
3.3.3 Conclusion 29
3.4 Design Rationale 30
3.4.1 Introduction 30
3.4.2 Choice of Platform 30
3.4.3 Choice of Programming Language 31
3.4.3.1 Background 31
3.4.3.2 Decision 32
3.4.4 Software Module 32

3.4.5 User Interface 33
3.4.5.1 Goals 33
3.4.5.2 Primitive Design 34
3.4.5.3 Improved Design 36
3.4.5.3.1 Advantages 39
3.4.5.4 Further Improvements 40
3.4.6 Quiz Questions 43
3.4.6.1 Goals 43
3.4.6.2 Examples 43
3.4.7 Implementation Details 44
3.4.7.1 Goals 44
3.4.7.2 Object-Oriented Design in RAD 45
3.4.7.3 Object-Oriented Programming 45
3.4.7.4 Object Representation 46
3.4.7.5 Polymorphism and Virtual Functions 50
3.4.7.6 Directory and File Format 51
3.4.7.7 Object-Oriented Design: Before and After 53

4 RESULTS 56

4.1 Evaluating the Program 57
4.1.1 Introduction 57
4.1.2 Results 58
4.1.3 Conclusion 59

5 CONCLUSIONS 61

5.1 Conclusions and Possible Extensions 62

APPENDIX A SURVEY RESULTS 63
APPENDIX B SOURCE CODE (Learning C++) 66
APPENDIX C SOURCE CODE (Survey Form) 106

BIBLIOGRAPHY 110

TABLE OF FIGURES

Figure 3.1	The Experiential Learning Cycle (Glover)	22
Figure 3.2	Primitive Interface – Table of Contents	34
Figure 3.3	Primitive Interface – Tutorials	35
Figure 3.4	Tutorials for the Improved Interface	36
Figure 3.5	Demonstrations for the Improved Interface	37
Figure 3.6	Quiz Selection Menu for the Improved Interface	37
Figure 3.7	Multiple Choice Question	38
Figure 3.8	Fill in the Blanks Question	38
Figure 3.9	Short Answer Question	39
Figure 3.10	Further Improved Interface on Tutorial	41
Figure 3.11	Further Improved Interface on Demo	42
Figure 3.12	Detailed Answers to Quiz	43
Figure 3.13	Program Flow Cycle for RAD Application	45
Figure 3.14	Class Hierarchy for the CAI Program	50

TABLE OF TABLES

Table 1.1	Functions of the Three Components	9
Table 3.1	Computer-Aided Instruction (CAI) Survey	28
Table 3.2	Functions of the Three Components	33
Table 3.3	Example of a Recall Question	44
Table 3.4	Example of a Challenging Question	44
Table 3.5	Object Representation for the CAI software	49
Table 3.6	Tutorial Index File Format	51
Table 3.7	The Contents of “contents.dat”	52
Table 3.8	Example of a Quiz Question Data File (q_XX.dat)	52
Table 3.9	Example of a Tutorial Index File (index.dat)	53
Table 3.10	Procedural Object-Based Programming extracted from part of the Second Version of the code	54
Table 3.11	Object-Oriented Programming Paradigm	54
Table 3.12	Polymorphism	55
Table 4.1	Computer-Aided Instruction (CAI) Survey	58
Table 4.2	Results on Program Evaluation Form	59

1 INTRODUCTION

1.1 Project Introduction

Computer Aided Instruction (CAI) is educational software that provides the delivery of instructional material on a computer. The effectiveness of CAI is questionable. Many CAI programs are no better than reference books, providing little interaction with the user and no accommodation for various learning styles.

As computers become more prevalent, the role of computers in education becomes more important. If CAI programs remain ineffective, a valuable educational resource will go to waste.

This project presents the history of CAI, and suggests ways to make CAI more effective. The project investigates various theories about learning and applies them to the development of a CAI program. The CAI program incorporates principles of good interface design.

1.2 Project Goals

The goals of this project are to study the criteria that contribute to the effectiveness of CAI programs, and to design and implement an effective CAI program that teaches C++ programming based on those criteria.

In order to be effective, the CAI program is aimed at the following areas:

- Content — the content must be accurate.
- Presentation — the presentation of information must be clear. Each lesson should provide a clear objective. Also, effort should be made in reducing the memory load for the user.
- Interaction — the program should provide good interaction.

The CAI system is composed of three components: the tutorials that present the C++ material, the demonstrations that illustrate the concept using animation, and the quizzes that test the material being taught.

Component	Description	Effect
Tutorials	Concise, easy to read material that delivers the C++ information in a simple way	Providing concise and easy to read material helps to reduce the memory load of the user, thus increasing learning ability
Demonstrations	Illustrate concepts from the tutorial using animation	Using graphical visual aid helps the user remember more information
Quizzes	Self-addressing questions for the user to check how much he or she has learned	By rewarding correct answers and correcting incorrect answers, the concept is reinforced

Table 1.1 Functions of the Three Components.

Since it was not feasible to prepare CAI material that covers all aspects of the C++ language, the subject of "looping" was chosen for the content of the tutorials. By limiting the scope of the educational content, the project could focus more on the design principles.

1.3 Project Progression

The project was broken into four parts: research, design, implementation, and evaluation. During the research phase, different types of CAI materials, the history of CAI, and the social implications of CAI including its growing potential, advantages and challenges were studied.

Before designing the CAI program, certain learning theories that are related to the design of a CAI program were studied, including "Experiential Learning", "Behaviorism", "Neuroscience", and "Learning Styles". Based on these theories, some design principles were drawn.

Also, in order to get some ideas on developing an effective CAI program, a survey was given via an educational discussion group on the Internet. The results were useful in indicating which aspects of the design should be of the most concern.

In designing the user interface, human-computer interaction principles were adopted. Designing an effective user interface was a very difficult task; in this project, three

revisions on the interface were done. They will be presented together with a description on why the improved interfaces are better.

The quiz questions were designed to enable the student to review the learned material and to encourage thinking by providing more challenging questions.

The object-oriented design was done within a Rapid Application Development (RAD) environment. In the implementation of this project, polymorphism was applied so that system flexibility was maximized.

Also, the various index and data file formats that were used were explained to allow for future extensions to the tutorials.

Finally, the program was evaluated and some conclusions were made. Possible extensions to the project are suggested.

2 BACKGROUND

2.1 What is CAI?

Computer-Aided Instruction (CAI) is an “educational medium in which instructional content or activities are delivered by a computer.” [Munden] It is also known as computer-assisted instruction.

The types of CAI materials most commonly used today are tutorials, drill and practice exercises, and simulations.

2.1.1 Tutorials

Tutorials are self-instructional programs or documents that present lessons on computers in some combination of text and multimedia formats, which include photographs, videos, animation, speech, and music. [Arnold & ETCAI] Examples include self-instructed reading materials, pre-recorded lectures, and the visualization of concepts. During tutorials, new concepts are presented to the student. The interaction between the student and the computer is like that between the student and the teacher. [Munden] The development of tutorials has been a challenge for many years because the tutorial is fundamentally non-interactive in nature; as a result, tutorials need to be designed with more interaction in mind, otherwise students will become bored and quickly lose attention. [ETCAI]

2.1.2 Drill and Practice Exercises

Drill and practice exercises are materials that consolidate the knowledge learned by the student. They require responses from the student and provide feedback. Most drill and practice exercises are multiple-choice questions; however, fill in the blank questions and graph-drawing questions are also possible. Drill and practice exercises can include games such as a crossword puzzle in which the student learns spelling by filling in the puzzle. A drill and practice exercise could also be used as an alternate instructional tool; for example, after using a typing practice program for a while, the student would remember the position of the keystrokes without having to memorizing them explicitly. [Munden]

In drill and practice exercises, the student's performance is often measured. The student is rewarded for good performance. In this way the learning process is reinforced. Also, from the performance statistics the program can decide whether or not additional help is needed for the student.

Students often find drill and practice exercises boring. A more challenging drill and practice exercise typically uses a random number generator to create an unpredictable set of problems that exposes the student to more situations than is possible using a textbook alone. [ETCAI]

2.1.3 Simulations

Simulations are used to predict the outcome or impact of an action without really performing the action. Since the student does not pay real consequences for the action he or she performs, simulations are often used in military and pilot training and other possibly dangerous real-life situations.

A simulation program is often not intuitive, and so extra time is needed to learn its operation. [ETCAI] However, unlike tutorials, the student has more control over the learning process and therefore is more likely to be involved.

2.2 History of CAI

In the mid-1950s and early 1960s, CAI was introduced into some selected elementary schools resulting from the collaboration between educators at Stanford University in California and International Business Machines Corporation (IBM). At the beginning, CAI programs presented information mainly with drill and practice exercises. Meanwhile, the high expense of obtaining, maintaining and using computers limited the early CAI systems. [Arnold]

In the early 1960s, the University of Illinois initiated a CAI system called Programmed Logic for Automatic Teaching Operations (PLATO). It was later

developed by Control Data Corporation and was used in higher education. PLATO consisted of a mainframe computer that supported up to 1000 terminals for use by individual students. In the United States, over 100 PLATO systems were operating by 1985; from 1978 to 1985, there were 40 million hours of usage logged on PLATO systems. A communication system between students was also introduced, which was the forerunner of modern electronic mail. [Arnold]

Another CAI project developed by Mitre Corporation and Brigham Young University in Utah was the Time-shared Interactive Computer Controlled Information Television (TICCIT) system, which was based on personal computer and television technology. In 1970, TICCIT was used to teach freshman-level mathematics and English classes. [Arnold]

In the 1980s, the use of CAI increased dramatically because of the introduction of cheaper and more powerful personal computers. By 1980, CAI systems had only been adopted by 5 percent of elementary schools and 20 percent of secondary schools in the United States. By 1983, both numbers had roughly quadrupled. By the end of the decade, nearly all schools in the United States and in most industrialized countries were equipped with CAI systems. [Arnold]

2.3 Social Implications

2.3.1 Introduction

The time for the widespread use of CAI has come. The term CAI itself implies the importance of using computers as the teaching media. Undoubtedly, CAI can hardly be successful if computer usage itself is not widespread. Given the advent of the personal computer, "there was a time when computers were a luxury item for American schools, but that time has clearly passed." [Bangert-Drowns, 1985] Beginning in the 1970's, many schools started to use computers for instructional purposes; according to Kinnaman, between 1981 and the end of that decade:

- American schools acquired two million computers.

- The number of schools owning computers increased from approximately 25 percent to virtually 100 percent.
- More than half the states began requiring, or at least recommending, pre-service technology programs for all prospective teachers.

Indeed, “the number of computers in American schools has risen from one for every 125 students in 1981 to one for every nine students in 1996. While the United States leads the world in the number of computers per school student, Western European and Japanese schools are also highly computerized.” [Arnold]

In addition, the advance of computer technology has made CAI a more attractive aid in education. For example, the introduction of the graphics display terminal has increased the presentation power of computers and aroused educators' interest in introducing CAI into their classrooms.

Given the facts above, it is clear to see why Kinnaman believes "the educational use of computer technology will surely continue to grow." As computers become increasingly popular throughout the world, it is believed that they will be fully utilized as teaching tools. Hence, the potential of CAI should not be understated.

2.3.2 Advantages

There is no reason to use CAI as a teaching aid if it does not benefit students. But the benefits of CAI are enormous. According to ETCAI, "CAI allows students to practice procedures as long as required to achieve defined competencies." This ensures that students remain focused on the topic and gives them the flexibility to learn at their own pace [Beres, 8]. It provides students with increased instruction without putting extra burdens on the instructor [Frith]. In addition, since students are informed of their misconceptions through immediate feedback from the CAI program, it is unlikely that the students would be learning the wrong concepts [ETCAI].

In order to provide positive reinforcements in the education process, "good CAI material reward[s] students immediately for correct responses and behaviors. This

encourages students to confidently move to more complex concepts." [ETCAI] On the other hand, the negative reinforcements that CAI provides are minimized because "it does not embarrass students who make mistakes." [Cotton] "This reduction in negative reinforcement allows the student to learn through trial and error at his or her own pace. Therefore, positive attitudes can be protected and enhanced." [Robertson, 314]

Previous researchers conclude "the use of CAI leads to more positive student attitudes than the use of conventional instruction." [Cotton] CAI is becoming the preferred teaching medium because students are becoming computer-oriented and they prefer learning in the environment the computer offers [Beres, ii & 8]. "CAI programs insure that students pay attention and understand by constantly testing them on the information they are being taught"; by providing the students with a feeling of control, they acquire a feeling of self-confidence [8].

Cotton found that students who use CAI “have more of an internal locus of control/sense of self-efficacy than conventionally instructed students", "had better attendance", "had higher rates of time-on-task than traditionally instructed controls", and exhibited greater cooperative and pro-social behavior.

CAI also gives students control over their schedules. Since the material in the computer-aided tool is always accessible, CAI accommodates students who learn better at different times of the day [Cergneux, 8].

There are other situations in which CAI proves to be useful. For example, since a CAI program does not experience fatigue, the same material can be presented over and over again without deterioration in teaching quality [Alessi, 5]. For situations in which safety is a concern, CAI can be used to simulate real events [5].

How can it be proven that students really learn better by using CAI programs than by using conventional methods? Past research finds that when CAI is used as a supplement to traditional instruction methods, the “effects are superior to those obtained with traditional instruction alone.” [Cotton] Researchers have found that learning rate can be enhanced by CAI; for the same amount of material, students

using a CAI program learned in less time than the traditionally instructed students [Cotton]. Similar studies found that the performance of students receiving CAI training is often 10 to 30 percent better than the performance of those students receiving conventional instruction [Army].

Another study by Orlansky and String (1979, White House) showed that there was a 30 percent reduction in time for students to achieve criterion performance using CAI in military training. While some people may worry that the faster the learning rate, the shorter the retention, researchers found that "the retention of content learned using CAI is superior to retention following traditional instruction alone." [Cotton]

2.3.3 Challenges

2.3.3.1 CAI in General

Although many researchers insist that teaching using CAI is more powerful than using conventional instruction methods, there are still challenges that must be overcome before CAI can be fully utilized as an instructional tool.

The most common criticism of CAI is that not many CAI programs are well designed. "Many programs are simply computerized textbooks rather than interactive media." [Beres, 15] Vockell and Schwartz concur that CAI "should exploit the full advantages of computerized drill tutorials — otherwise, it may be better to have a teacher, tutor, or textbook deliver the instruction." [Vockell, 63] There has also been a suggestion that CAI should incorporate more intelligence to make it more effective [Lewis, 4]; for example, artificial intelligence could be used to generate sets of drill and practice questions that suit students of different skill levels.

However, an effective CAI program is often expensive to produce, and it requires a large investment of time and expertise [Frith]. A main problem in the development of CAI is that the lifetime of a CAI program is short. As changes occur in subject matter, the CAI programs rapidly become obsolete [9]. Thus, it is doubtful that the reward for CAI developers would compensate for the time invested.

2.3.3.2 CAI in Classrooms

In order for CAI to become an effective tool in the public school system, computers must be an integral part of the classroom, but some schools do not have sufficient funds to afford a computer in every classroom [Beres, 15-16]. If many students are to have frequent access to CAI programs, a large number of computers are needed, and specialized staff are needed to maintain such facilities [Frith]. This induces a great monetary burden on schools that want to use CAI as an instructional tool. However, it is believed that the cost of adding CAI to a training program is adjustable; less expensive CAI materials such as supplemental training software could be used in a tightly budgeted school while at the same time, these CAI materials can reinforce the existing classroom instruction [ETCAI].

There are some other considerations in determining the success of CAI in classroom teaching. First, "the instructor's teaching style and opinions must be taken into consideration." [ETCAI] Teachers who have a strong belief in the value of CAI will favor the use of it in their classroom [ETCAI], whereas there are still many teachers who are reluctant or even unable to integrate CAI into their classrooms [15]. It may be due to the fact that "extra time is necessary for the instructor to successfully combine a computer assisted instruction tool into the course plan." [Cergneux, 8] In addition, CAI may restrict the instructor or user to a few fixed teaching strategies [Kazmierczak, 9]. "Programmed instructions and drills were not a universally accepted form of instruction since they had limited applicability to the teaching process." [9] In order to maximize the effect of using CAI as an instructional aid, classroom teaching must be altered [Cergneux, 8]. There are also some students who do not feel confident about using computers; "they may not be well acquainted with the understanding of the operation of the necessary software/hardware, which must be perfected on their own." [8]

2.3.4 Conclusion

In short, the ubiquitous presence of computers and the technological advances in hardware and software have made CAI a more attractive instructional tool today than it was several years ago. There are many advantages to using CAI in education and

many researchers have proven that students learn effectively when CAI is used. On the other hand, there are still many challenges that must be overcome. If these challenges are addressed correctly, CAI will revolutionize the way people acquire knowledge.

3 PROJECT DESCRIPTION

3.1 Introduction

The goals of this project are to study the criteria that contribute to the effectiveness of CAI programs, and to design and implement an effective CAI program based on those criteria.

3.2 Psychological Basis

3.2.1 Introduction

One of the goals of designing CAI software is to try to enhance learning potential by incorporating learning theories. When these theories are correctly applied, the CAI software will be the most successful. There are basically two questions that must be addressed:

- How do we learn?
- How can we incorporate the answers to the above question into the design of the CAI software?

3.2.2 Learning Theories

Learning is “any increase in knowledge, memorizing information, acquiring knowledge for practical use, abstracting meaning from what we do, and a process that allows us to understand.” (WAVE)

There have been many theories about how people learn:

3.2.2.1 Experiential Learning

According to Glover, we learn by continuously experiencing knowledge. The main idea is that learning is a continuous process consisting of four elements: Experiencing, Reviewing, Concluding, and Planning. Higher levels of learning are achieved when we iterate the cycle:

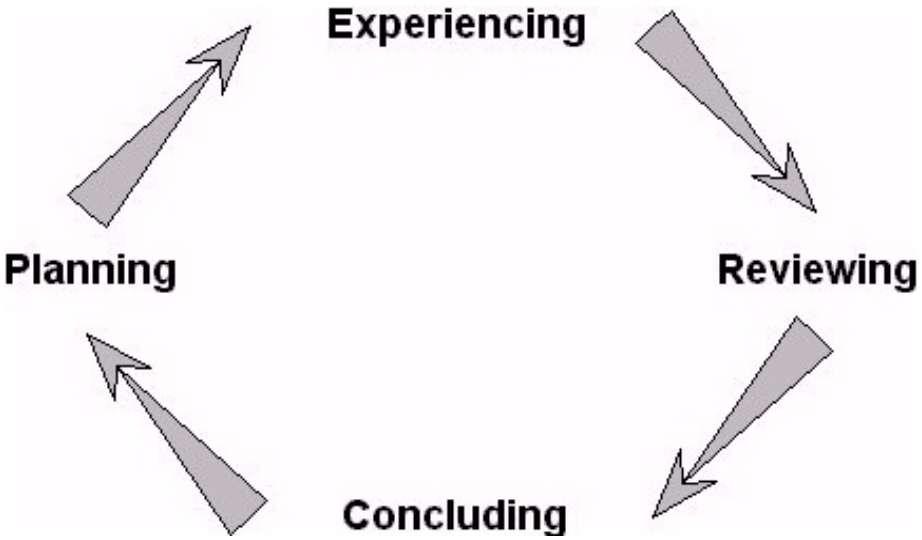

Figure 3.1 The Experiential Learning Cycle (Glover).

Experiencing refers to applying learned knowledge to a life situation. By experiencing, you know much more about what you have already learned and eliminate some of the misconceptions. While you are experiencing, you look at the outcome and figure out the degree of success.

Reviewing refers to the recall of previous experience, if there is any. When you are reviewing, you try to improve the way of doing something by preventing past mistakes from happening again.

Concluding refers to interpreting the outcome of an action. By interpreting the outcome you understand more about what you have learned.

Planning refers to a plan of action. Based on your experience or the reflective review of others, you develop an implicit and sub-conscious plan on what you are going to do.

In conclusion, more practice results in better learning. The improvement in each cycle is large in early stages but is dampened after each cycle.

One of the main advantages of CAI is that its drill and practice questions let the student practice what he or she has learned during the learning process. However, we should ensure that the system contains a reasonable number of questions so that the student can have ample opportunity to practice. We should also endeavor to design exercises that are related to real life situations as much as possible.

3.2.2.2 Behaviorism

Behaviorism is “a movement in psychology that advocates the use of strict experimental procedures to study observable behavior (or responses) in relation to the environment (or stimuli).” (Bijou)

In Behaviorism, conditioning is regarded as a universal learning process. There are two kinds of conditioning (On):

1. *Classical Conditioning*

   ”Classical conditioning occurs when a natural reflex responds to a stimulus.” Humans and animals are biologically “wired” so they produce a specific response to a certain stimulus.

2. *Behavioral or Operant Conditioning*

   ”Behavioral or operant conditioning occurs when a response to a stimulus is reinforced… If a reward or reinforcement follows the response to a stimulus, then the response becomes more probable in the future.”

Behaviorism relies basically on positive (reward) and negative (punishment) reinforcements that encourage and discourage behaviors. In a CAI system this technique could be used in the drill and practice exercises. For example, a smiling

face can be given to reward the student for a correct answer, whereas a sad face can be given to the student as punishment.

3.2.2.3 Neuroscience

Neuroscience is “the study of the human nervous system, the brain, and the biological basis of consciousness, perception, memory, and learning.” (On) In neuroscience, the human nervous system and the brain form the basis of learning.

Neuroscience basically explains how memory is developed. Our brains are always changing when we use them, and they strengthen certain patterns of connections, which make the connections easier to create the next time. (On)

To apply neuroscience to the design of CAI, the system should encourage the use of the brain. For example, the drill and practice exercises could be designed in such a way that the answers are not intuitive to the student. In addition, the exercises should be challenging. The more challenging the questions, the more the student’s brain connections will be reinforced.

3.2.2.4 Learning Styles

As we perceive and process information differently, we cannot expect everyone to learn the same way. Findings show that the amount that a person learns is closely related to his or her learning style. In an article from On Purpose Associates, the different learning styles are classified as follows:

1. *Concrete and abstract perceivers*

   ”Concrete perceivers absorb information through direct experience, by doing, acting, sensing, and feeling. Abstract perceivers, however, take in information through analysis, observation, and thinking.”

2. *Active and reflective processors*

> ”Active processors make sense of an experience by immediately using the new information. Reflective processors make sense of an experience by reflecting on and thinking about it.”

Different learning styles should be incorporated into the CAI program. For example, demonstrations could be used to illustrate concepts in a sensible way. In addition, special visual and audio representations like colors, animation, sound and music could be used to enhance the learning experience.

3.3 Survey on CAI Program Design

3.3.1 Description

A survey was developed as a means of soliciting input on the factors that contribute to an effective CAI program. The survey request was sent electronically over the Internet through 30 random ICQ messages, and was posted on educational discussion groups. The reason for choosing an educational discussion group was that the participants in this group would probably be more knowledgeable in education, and would therefore provide the most useful ideas.

The survey was an HTML document prepared using Microsoft’s FrontPage 2000. A CGI program was written in Perl [Appendix C] so that the survey could be sent back directly on the participants’ web browsers. This would hopefully encourage more people to fill out the survey because of its ease of use.

The survey consisted of 10 questions [Table 3.1]. The quantity was intentionally limited to keep the survey from being overwhelming. The first three questions focused on those people who had used a CAI program before. The questions asked what they thought about the CAI programs they had used, in order to provide feedback on the effectiveness of current CAI packages.

The next question asked about the kinds of instructional methods people prefer. It was included to help gauge people's receptiveness to CAI.

Question 9 asked the participants their opinions on what constitutes a reasonable price for a CAI program. As mentioned earlier, it is doubtful that the reward for CAI developers would compensate for the time invested. If the reward is too little, this could be an obstacle to the development of CAI programs.

The rest of the questions had to do with the content of CAI programs, for example, whether the users prefer to have sound more than graphics. Also, some of the questions asked for their opinions on how to make the programs more effective and interesting.

Computer-Aided Instructions (CAI) Survey

Background
Computer-Aided Instruction (CAI) is an educational medium in which "instructional content or activities are delivered by a computer." [Munden] This survey focuses on ideas about the design of educational software (on a CD-ROM), and tries to find ways that make such software packages more effective.

Source: Munden, C. Dianne. "What is Computer Assisted Instruction?" 21 Aug. 1996. http://www.auburn.edu/~mundecd/cai.html

The Survey

1. Have you ever used a CAI program?

   Yes (please specify the name of a package you have used: _____)
   No (skip to question 4)

2. On a scale of 1 to 10, where 1 means "nothing" and 10 means "I fully mastered the subject", how much do you think you learned from the CAI program?

   1 2 3 4 5 6 7 8 9 10

3. What features did you like/dislike most about those CAI programs?

4. Rank the following learning materials in the order of your preference (1: most preferred, 2: next preferred, etc):

   Book _____ Course _____ Computer _____
   Other _____ (please specify: _____)

5. Rank each of the following according to its order of importance for the overall efficacy of a CAI program (please make separate rankings for each category with 1 being the most important and 2 being the second most important, etc. Put an 'X' if the item is not applicable):

System

Ease of use _____ Performance _____
Other _____ (please specify: _____)

Presentation

Text Layout _____ Color _____ Graphics _____
Sound _____ Animation _____
Other _____ (please specify: _____)

Tutorial Content

Quantity _____ Easy to understand _____
Meaningfully structured _____ Accuracy _____
Other _____ (please specify: _____)

Drill and Practice Exercises

Quantity _____ Review learned materials _____
Encourage thinking _____ Feedback _____
Other _____ (please specify: _____)

6. What types of interaction could a CAI program provide so that it consistently maintains the user's interest?

7. How do you think the tutorial materials could be organized so that they are more interesting than the material in a book?

8. Some experts think that the drill and practice questions in CAI software are ineffective. Do you agree? Why? How would you suggest the questions be designed so that they become more effective?

9. What do you think is a reasonable price for a CAI program? (in US dollars)

$0, should be free
>$0 to $10
>$10 to $20
>$20 to $30
>$30 to $50
>$50 to $100
$100 or more

```
10. What would you like to see in future CAI programs?

  Your name (optional):
  Your e-mail (optional):
  Thank you very much for your time!
```

Table 3.1 Computer-Aided Instruction (CAI) Survey.

3.3.2 Results

Nineteen surveys were returned with the following results [Appendix A]:

- Only 21% of the respondents had actually tried out a CAI program in the past.
- In general, people felt they did not learn a lot from using CAI programs.
- People liked CAI programs because they are user friendly and interesting. The users respond favorably to a visually appealing interface. They liked the activities because they were interesting. Example given was a game testing the materials being learned.
- Respondents preferred most to learn from a course (47%), then from books (32%), and finally from computers (21%).
- 53% of the respondents regarded computers to be the least preferred learning tool.
- A majority of respondents (79%) rated ease of use over system performance.
- In regard to presentation, most people rated text layout as the most important element for a CAI program (53%). Graphics were also listed as being important, while animation and sound were regarded as relatively less important.
- Most people thought it was most important for the tutorial to be easily understandable (47%). It is also considered important that those contents be meaningfully structured (26%). Quantity and accuracy were regarded as relatively less important.
- Most respondents thought the drill and practice exercises that a CAI program provides should focus on reviewing learned materials and encouraging thinking; the quantity of exercises and feedback were considered less important.

- To consistently maintain the user's interest, respondents suggested providing feedback on the user's performance, including a user-friendly interface with more graphics and animation, and providing supplementary information.
- People suggested the tutorial materials should be brief and concise, and should include a glossary.
- The most reasonable price for a CAI program was suggested to be between $10 and $20.
- In the future, people would like to see that using a computer to learn would be no different from attending classes. Also, they are looking for CAI programs on more advanced topics.

3.3.3 Conclusion

The survey results showed that CAI programs do not have a lot of public support. When people want to learn about something, the first image that comes to their minds is to take a course or to read a book. CAI programs are nearly ignored. As a result, more work needs to be done in advertising CAI programs so that people know they can learn from their computers as well as from traditional methods.

The survey also showed that people learn little using CAI programs. Again, it is the effectiveness problem that the CAI program developers need to address. However, a user-friendly and interesting CAI environment could compensate for this weakness to a certain extent. The psychological basis discussed previously in this project is strongly recommended in developing any CAI programs.

Generally speaking, an effective CAI program should be measured by how much the student learns from it, rather than by how many features it provides. This suggests that it is more valuable to make a brief and clear presentation than it is to add various extra features like animation and sound.

Drill and practice exercises should provide a review of the tutorial materials. After that, some challenging questions could be asked in order to encourage thinking.

The survey also showed that people do not wish to spend much money on buying CAI programs.

3.4 Design Rationale

3.4.1 Introduction

The major goal of this project is to design and create an effective CAI program that teaches anyone with a little computer experience to learn programming in C++. In order to be effective, the CAI program is aimed at the following areas:

- Content — the content must be accurate.
- Presentation — the presentation of information must be clear. Each lesson should provide a clear objective. Also, effort should be made in reducing the memory load for the user.
- Interaction — the program should provide good interaction.

3.4.2 Choice of Platform

The program could be developed under any platform. For example, it could be developed under DOS, Windows, Linux, or even on the more recent World Wide Web (WWW) technology. Developing a CAI program on the WWW is a good idea, but there are some drawbacks. First, accessing CAI materials from the WWW pages is comparatively slower than accessing them from the hard drive. If the CAI materials contain rich multimedia presentations, the difference in speed would become obvious. This could affect learning because people would become frustrated or annoyed with slow display rates, and this would lead to frequent errors. (Shneiderman) On the other hand, accessing the CAI materials on a hard drive requires less access time, so the interaction is more realistic. The program could be put online so it could be downloaded.

The Windows environment was chosen for the development of the CAI program for this project. It was chosen because the interface of Windows is more suitable for

developing interactive media, and it is the most common operating system in the world. However, Windows programming is quite challenging because of its interface. For each component, the design of the user interface is equally as important as its actual implementation.

3.4.3 Choice of Programming Language

3.4.3.1 Background

There are basically three methodologies available for writing a Windows program:

1. *Application Programming Interface* (API)

   In early years, the only way to do Windows programming was by calling the Windows internal API functions using C/C++. The programmer needed to be very familiar with the internal structures of Windows before he or she could write a Windows program. All the user interfaces had to be done by coding, which is extremely time consuming.

2. *Windows Class Libraries*

   Writing Windows programs using its primitive API functions is difficult because it deals with the internal structure of Windows. As a result, people tried to find ways that could minimize the effort of coding by abstracting some of the internal structure. This led to various object components like Microsoft Foundation Classes (MFC) and Borland's Object Windows Libraries (OWL). User interfaces like windows, buttons, menus, etc., are classified as objects. The programmer takes care of the operations on a particular object rather than how the operations are implemented. A strong background in object-oriented programming is required for this method of development.

3. *Rapid Application Development* (RAD)

   RAD has brought us the easiest method for programming under Windows. In an

RAD environment, the development tools provide the interface components that can be dragged to wherever you like on a window. Since the interface appears visually during the development process, the programmer can focus more on the actual task rather than the interface. Examples of RAD development tools are Microsoft Visual Basic, Borland Delphi, and Borland C++Builder.

3.4.3.2 Decision

For the aforementioned reasons, RAD was chosen as the development tool for the CAI program. The initial development was done using Visual Basic; because the language is simple and familiar, the hope was that the development effort could focus more on CAI issues than syntax issues.

However, after writing a few tutorials using Visual Basic, it became apparent that Visual Basic is not a good computer language for developing large software applications. While the object-based feature is adequate for developing small-to-medium-sized applications, the lack of a true object-oriented programming paradigm makes it hard to develop large programs. As the program grew, the code started to get messy and hard to debug.

As a result, a switch was made from Visual Basic to Borland C++Builder. It is a more suitable language for the project because the C++ language supports object-oriented programming intuitively, thus maintaining a good coding structure even for a very large program.

3.4.4 Software Module

The CAI system is composed of three components: the tutorials that present the C++ material, the demonstrations that illustrate the concept using animation, and the quizzes that test the material being taught.

Component	Description	Effect
Tutorials	Concise, easy to read material that deliver the C++	Providing concise and easy to read material helps to reduce the

	information in a simple way	memory load of the user, thus increasing learning ability
Demonstrations	Illustrate concepts from the tutorial using animation	Using graphical visual aid helps the user remember more information
Quizzes	Self-addressing questions for the user to check how much he or she has learned	By rewarding correct answers and correcting incorrect answers, the concept is reinforced

Table 3.2 Functions of the Three Components.

3.4.5 User Interface

3.4.5.1 Goals

The design of the user interface is very challenging because a good interface arouses the user's interest whereas a bad interface deters the user from using the program. To determine the criteria of a good interface, Shneiderman proposed the Eight Golden Rules of Interface Design:

1. *Strive for consistency* — the interface should adopt consistent sequences of actions, identical terminology, and consistent color, layout, capitalization and font, etc.
2. *Enable frequent users to use shortcuts* — the interface should provide shortcuts for frequent users so as to decrease the number of interactions and increase the pace of interaction.
3. *Offer informative feedback* — there should be system feedback for every user action.
4. *Design dialog yield to closure* — sequences of actions should be organized into groups with a beginning, middle, and end.
5. *Offer error prevention and simple error handling* — design the system in such a way that it minimizes the chance of errors made by the user. If the user makes an error, the system should detect it and offer simple, constructive, and specific instructions for recovery.

6. *Permit easy reversal of actions* — actions should be reversible so as to encourage exploration of unfamiliar options.
7. *Support internal locus of control* — the user should be in charge of the system. The system responds to the user's actions rather than the user responding to the system actions.
8. *Reduce short-term memory load* — the interface should be simple enough to fit the short-term memory of the user.

3.4.5.2 Primitive Design

For the first prototype, the interface was designed using Visual Basic. Here are some screen shots for the interface in the earliest stage:

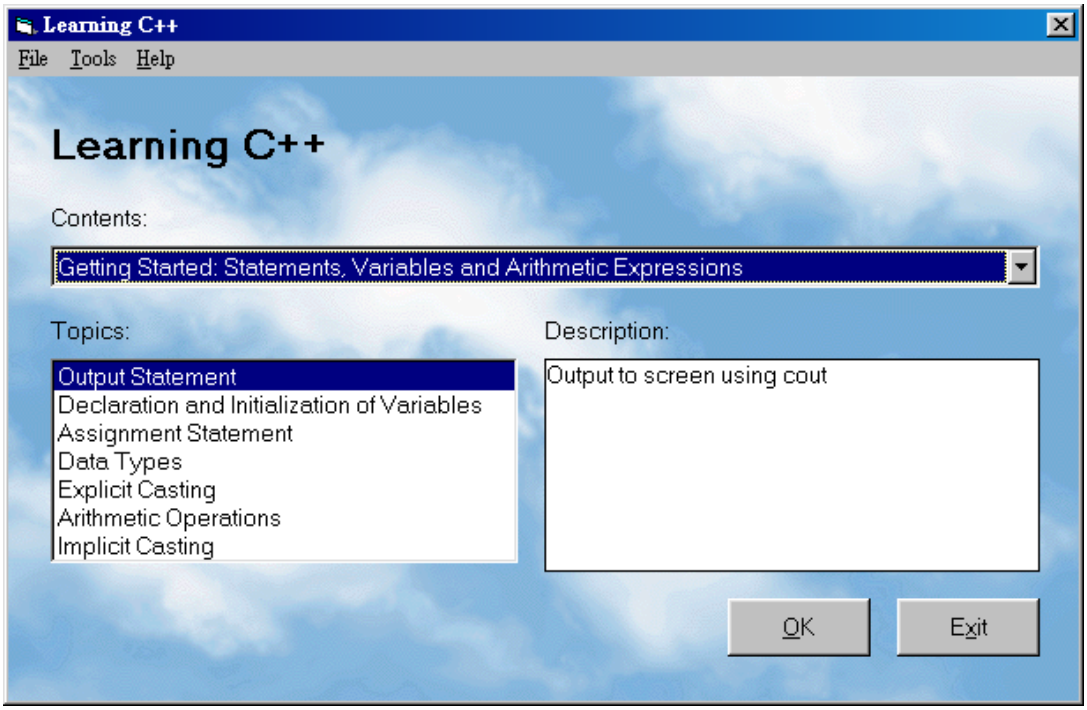

Figure 3.2 Primitive Interface – Table of Contents.

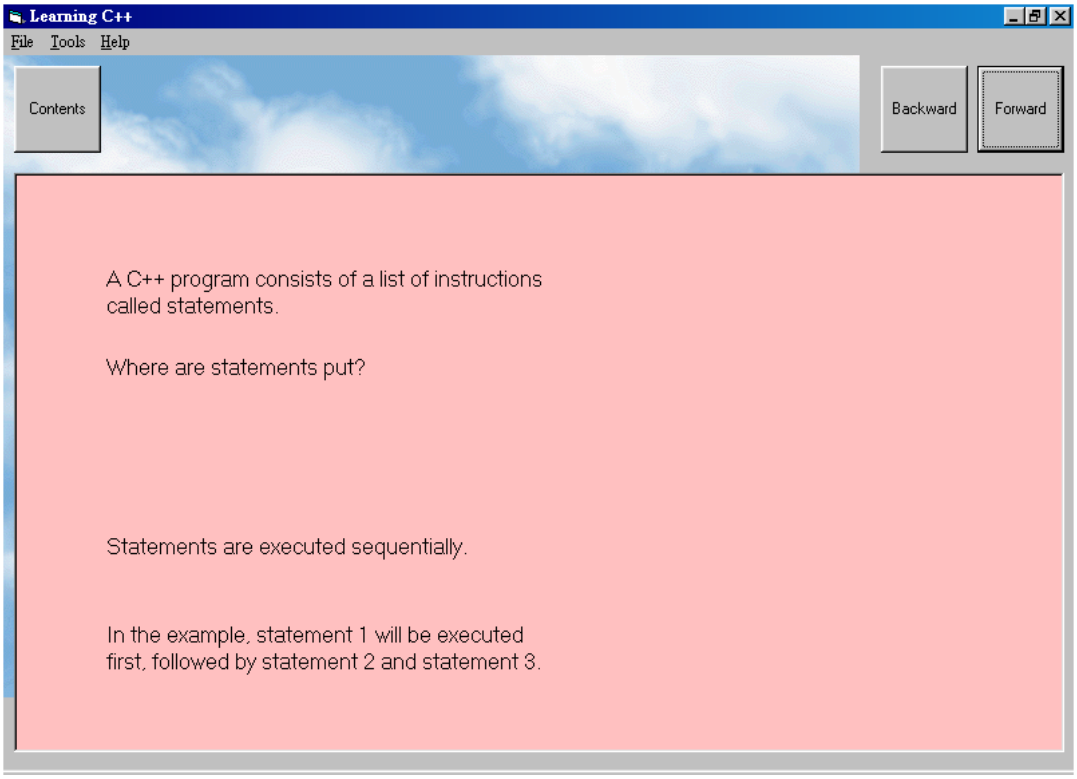

Figure 3.3 Primitive Interface – Tutorials.

3.4.5.3 Improved Design

Because the primitive interface looked intimidating, a new interface was proposed. The new interface, built with C++Builder, provided a great improvement over the primitive interface.

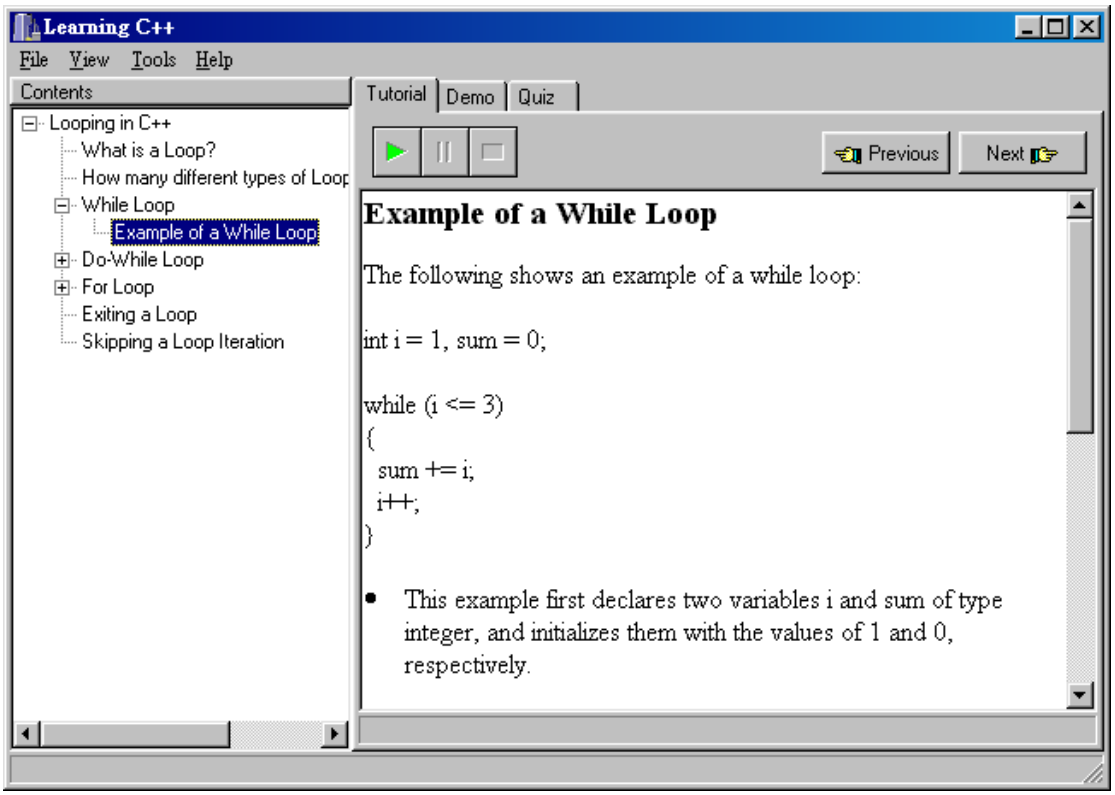

Figure 3.4 Tutorials for the Improved Interface. By providing concise and easy to read material, the interface helps to reduce the memory load of the user, thus increasing the user's learning ability.

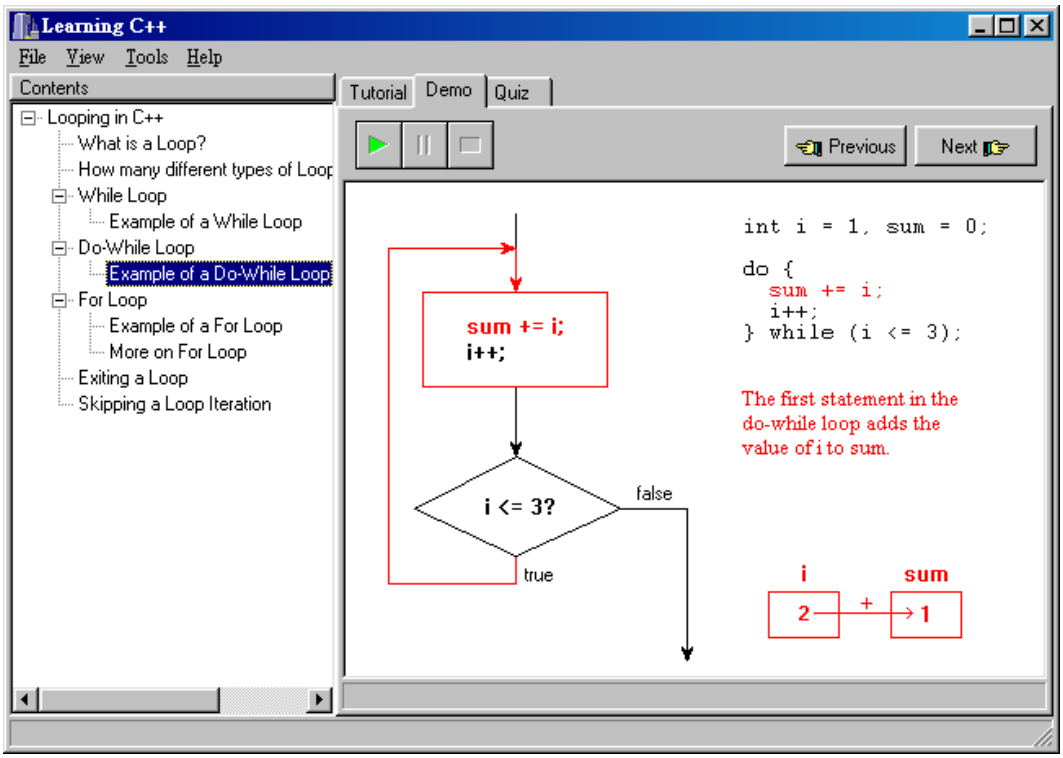

Figure 3.5 Demonstrations for the Improved Interface. By illustrating the concepts using graphical visual aids, the interface helps the user remember more than would be possible with text alone.

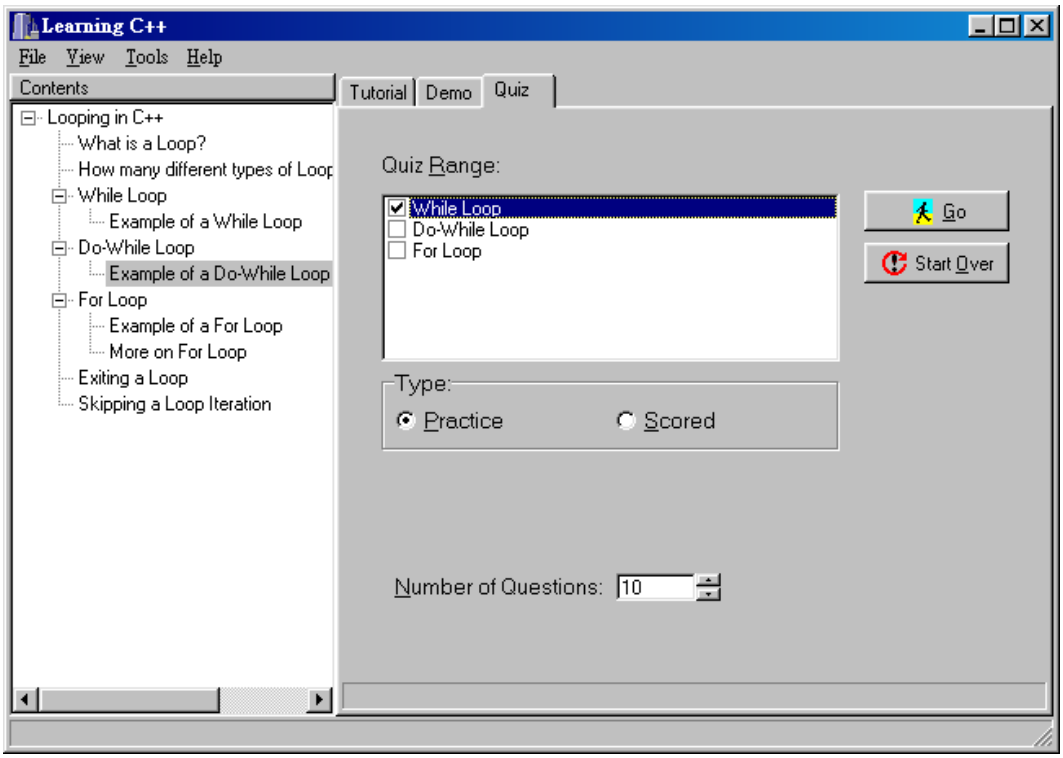

Figure 3.6 Quiz Selection Menu for the Improved Interface. The user can specify the topic to be tested on, and the type of the quiz (whether it is a practice quiz or a scored quiz). The number of questions is also specified here.

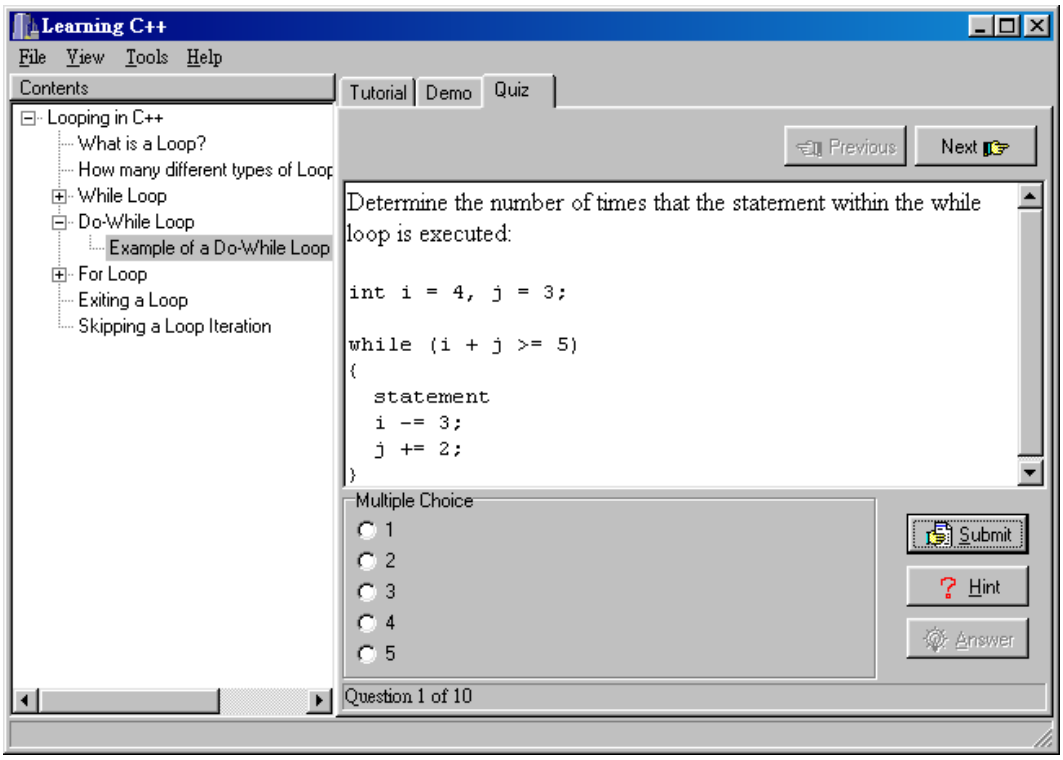

Figure 3.7 Multiple Choice Question. The user chooses from a list of answers.

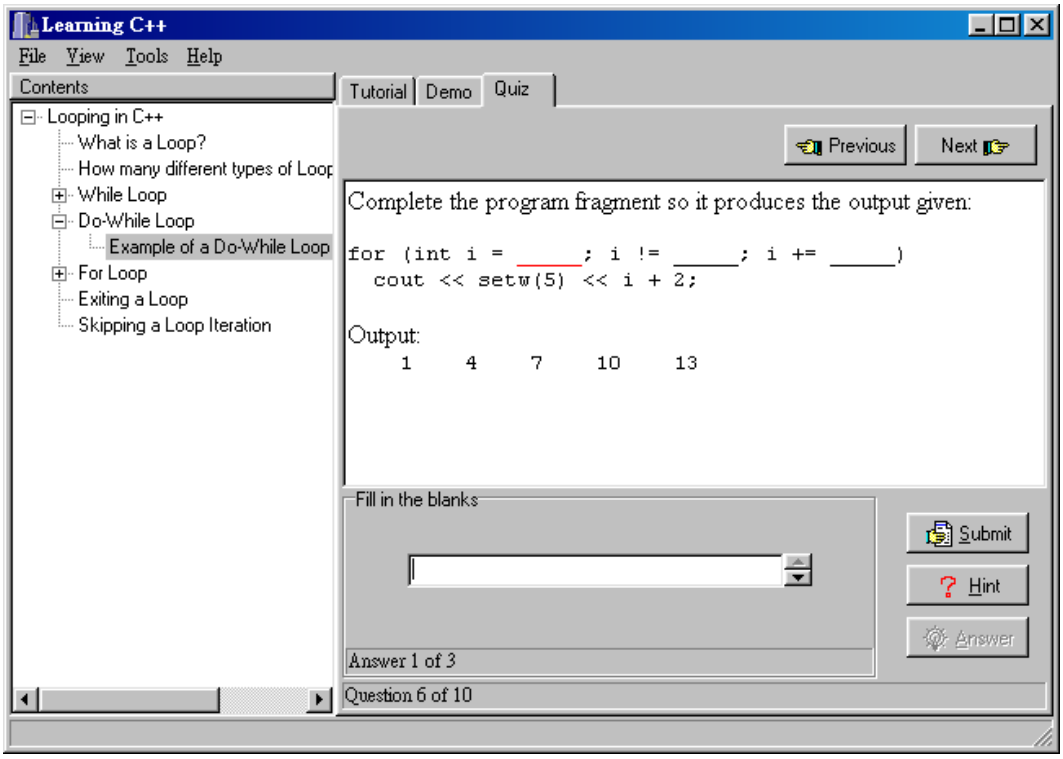

Figure 3.8 Fill in the Blanks Question. This kind of question requires one or more answers.

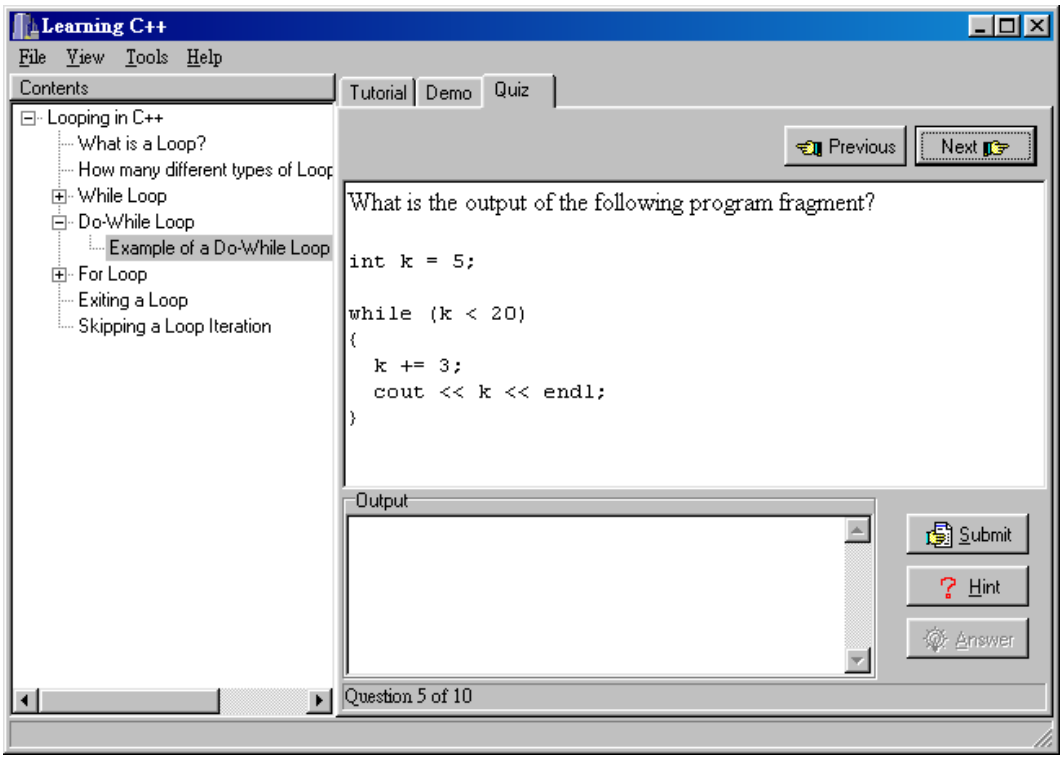

Figure 3.9 Short Answer Question. The required answer is a prediction on the program fragment output.

3.4.5.3.1 Advantages

The improved interface has a number of advantages:

- Consistency — instead of separating the table of contents from the tutorials, the improved interface integrated all the parts into one single window.
- Shortcuts — shortcuts are provided for the menus and the buttons.
- Feedback — for every user actions, the flow of the interface is natural and easy to follow.
- Closure — each component is grouped into separate windows with the previous and next buttons that indicates sequences of actions.
- Error prevention and handling — the interface limits the number of choices so as to prevent the user from making errors. Also, it pops up error messages with a brief and specific instruction for recovery.

- Action reversal — most of the actions can be reversed, except for the quiz section, which is intentionally designed not to be reversible. However, for actions in which a reversal is not possible, the system will inform the user before performing the action.
- Internal locus of control — instead of the system directing the user where to go, the user decides where he or she wants to go.
- Reduce short-term memory — the interface provides a limited in a number of choices so that it will not overwhelm the user.

3.4.5.4 Further Improvements

Further improvements were made to make the interface more natural. For example, in the second version the demo tab popped up suddenly. Also, the table of contents was visible while a quiz was in progress.

In the third version of the interface, the tab sheets were changed to buttons. While the “Demo” tab popped up suddenly in the second interface, the “Demo” button in this final interface is always visible. If there is no demonstration for a tutorial, the “Demo” button will be grayed out rather than disappearing altogether.

Colors were added to indicate important keywords and phrases. This can help minimize the memory load of the user by separating chunks of information in their minds.

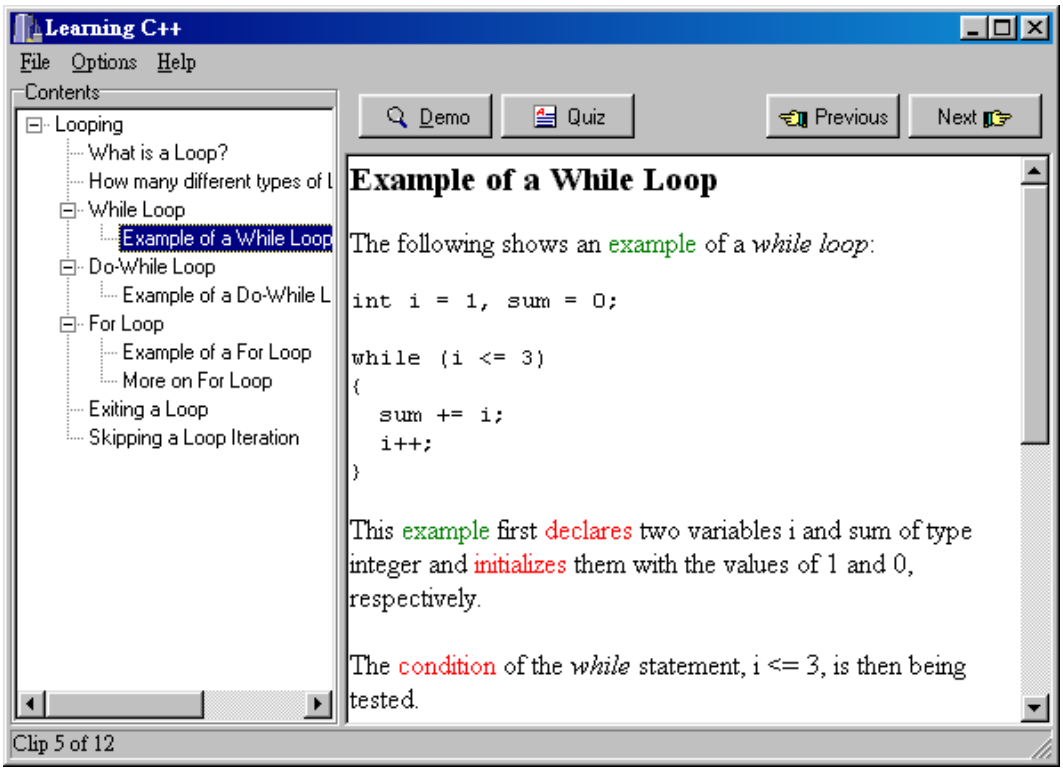

Figure 3.10 Further Improved Interface on Tutorial. The “Demo” button is always there, and it is grayed out when it is not applicable. Also, colors are used to indicate important keywords and phrases.

Instead of leaving the table of contents visible during a quiz session, in the third version of the interface it has been removed. Hence, the interface is less distracting.

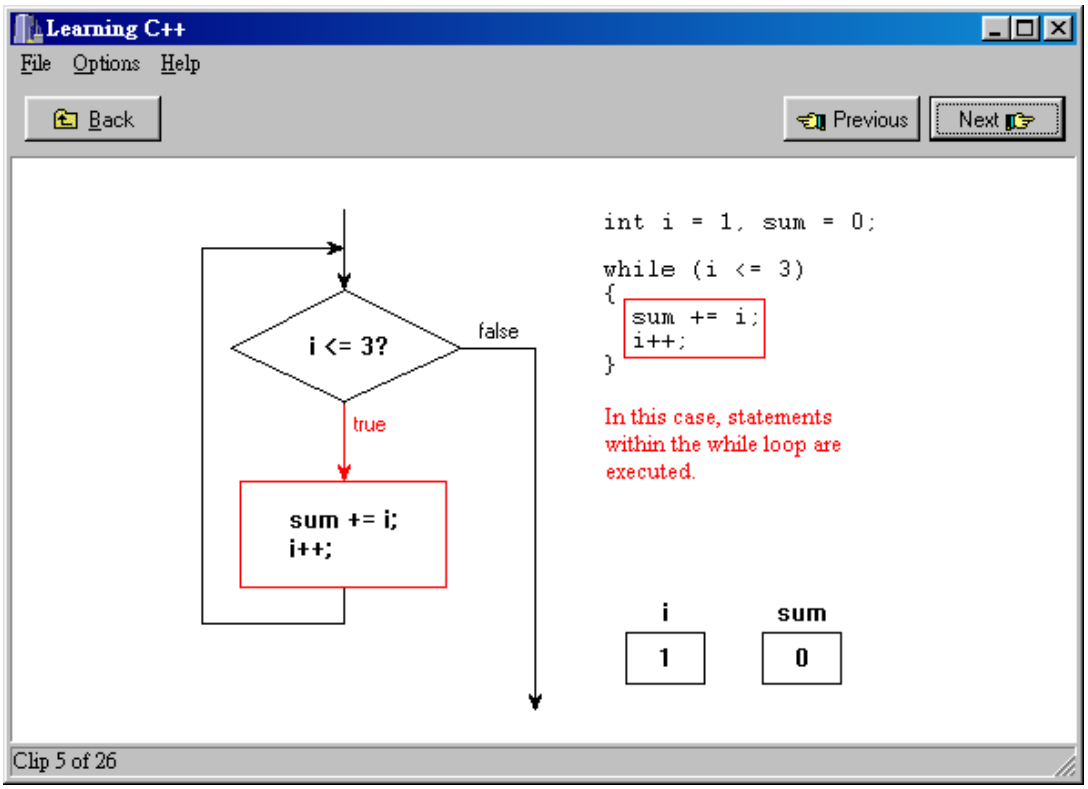

Figure 3.11 Further Improved Interface on Demo.

Also, there is a dramatic increase in efficiency in loading the demonstration images in the improved interface. While the old interface required a few seconds to initialize the graphics, the new interface displays the graphics immediately.

A further improvement is the addition of the explanation of the answers to the quizzes. By providing more detailed answers the student can learn more from taking the quizzes.

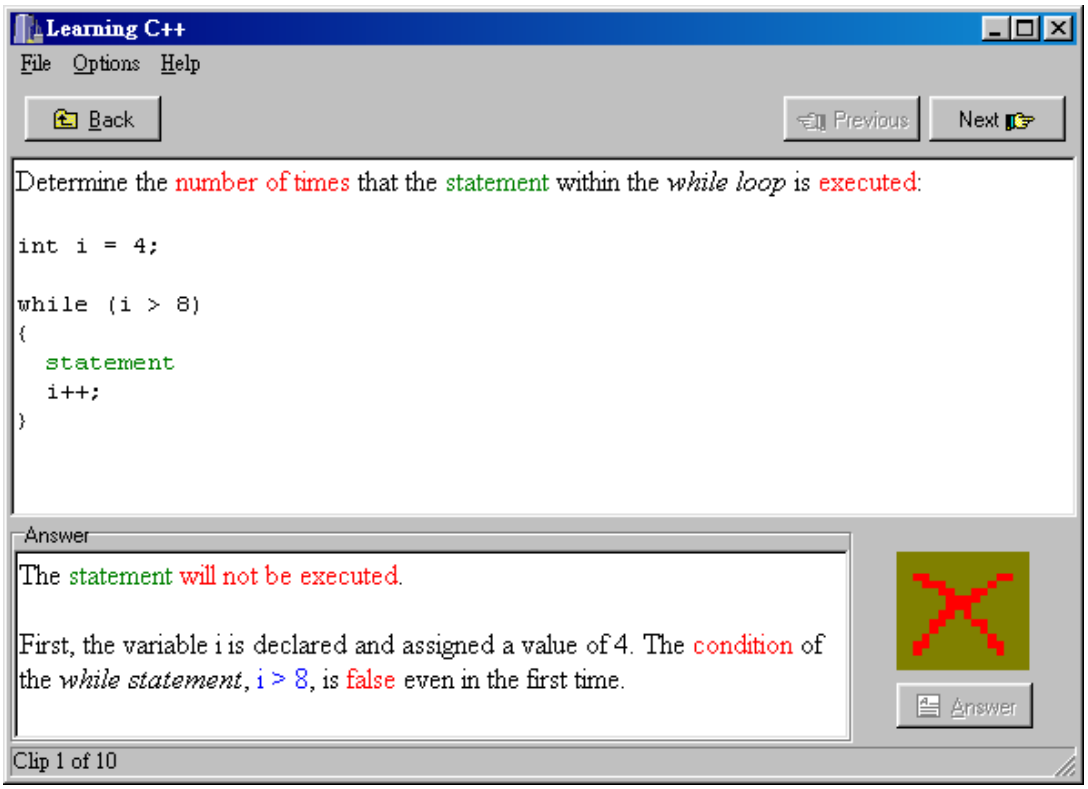

Figure 3.12 Detailed Answers to Quiz.

3.4.6 Quiz Questions

3.4.6.1 Goals

The design of the quiz questions focuses on the following areas:

- *Review learned materials* — by recalling the materials, the student remembers better.
- *Encourage thinking* — the more challenging the questions, the more the student's brain connections will be reinforced (neuroscience).
- *Accuracy* — the questions and the answers to the questions should be accurate, otherwise it will mislead the student.

3.4.6.2 Examples

In order to provide some review for the learned materials, some of the quiz questions are directly produced from the tutorial, with a keyword missing. This kind of question is not challenging, but serves well in helping the student to recall what he or she has learned. For example,

```
In a for loop, the computer tests the condition _____ it executes
the statements within the body of the loop.
```

Table 3.3 Example of a Recall Question.

For some challenging questions, the answers are not immediately obvious. These kinds of questions are quite demanding, and the student may make mistakes easily; however, the student has learned something while he or she is thinking, regardless of the correctness of the answer. For example,

```
Complete the program fragment so it produces the output given:

for (int i = _____; i != _____; i += _____)
  cout << setw(5) << i + 2;

Output:
    1    4    7    10    13
```

Table 3.4 Example of a Challenging Question.

3.4.7 Implementation Details

3.4.7.1 Goals

At the implementation level, the design focuses on the following areas:

- *Robustness* — the system must be robust; that is, it will not go down or crash the computer for any software failures (e.g., tutorial file missing) or user actions (e.g., pressing the wrong button).
- *Correctness* — the system should run correctly under normal situations. It should not respond incorrectly to a user for any action.

- *Expansibility* — the system components must be easily expansible; the developer should be able to add to the CAI materials without modifying the source code.
- *Speediness* — the response time for the system should be short.

3.4.7.2 Object-Oriented Design in RAD

In an RAD environment, the interface is comprised of objects of interface components like buttons, text boxes, picture boxes, check boxes, etc. To program is to specify the codes associated with an action on an object. This is called the object-based programming. The following diagram shows the flow diagram for an application:

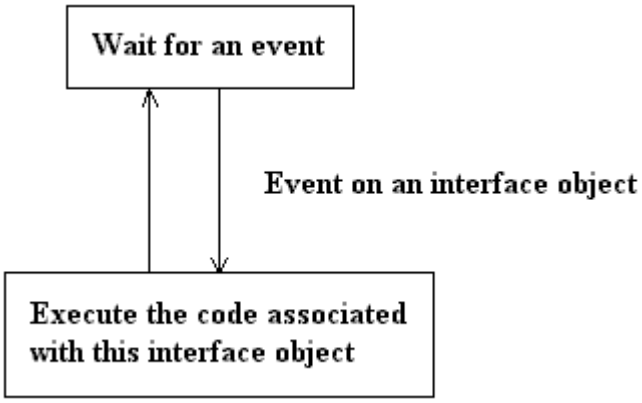

Figure 3.13. Program Flow Cycle for RAD Application.

3.4.7.3 Object-Oriented Programming

Object-oriented programming is different from object-based programming. In object-based programming, the objects represent the user interface components that are generally pre-defined by the development tools. In object-oriented programming, the focus is on the data representation of your program rather than that of the interface components.

The mixing of object-oriented programming and object-based programming sometimes confuses the programmer because they are used within the same program and they are coded in a similar way.

3.4.7.4 Object Representation

The CAI program is represented mainly by the following data structures: Tutorial, Demo, and Quiz. They are included in the following table:

Object	Operation	Description
Tutorial	`Constructor`	Takes a default tutorial directory and a default sound directory, and creates a new object
	`GetDirectory`	Returns the default tutorial directory
	`GetDemoIndexDirectoryAt`	Gets the name of the index file for the i-th demo clip
	`GetQuizIndexDirectoryAt`	Gets the name of the index file for the i-th quiz clip
	`GetNumClips`	Returns the total number of clips for this Tutorial
	`SetCurrentClipNum`	Takes a clip number and sets the clip to be the current one
	`GetCurrentClipNum`	Returns the current clip number
	`AdvanceClip`	Advances a clip
	`ReverseClip`	Reverses a clip
	`IsFirstClip`	Returns true if the current clip of the tutorial is the first one
	`IsLastClip`	Returns true if the current clip of the tutorial is the last one
	`Show`	Displays the current clip of the tutorial
Demo	`Constructor`	Takes a default demo directory and a default sound directory, and creates a new object
	`GetDirectory`	Returns the default demo directory

	`GetNumClips`	Returns the total number of clips for this demo
	`SetCurrentClipNum`	Takes a clip number and set the clip to be the current one
	`GetCurrentClipNum`	Returns the current clip number
	`AdvanceClip`	Advances a clip
	`ReverseClip`	Reverses a clip
	`IsFirstClip`	Returns true if the current clip of the demo is the first one
	`IsLastClip`	Returns true if the current clip of the demo is the last one
	`Show`	Displays the current clip of the demo
Quiz	`Constructor`	Takes a default quiz directory and the number of questions in the quiz, and creates a new object
	`Destructor`	Used to deallocate quiz questions
	`AddQuestion`	Take a pointer to QuizQuestion and add the question being pointed to
	`SetCurrentClipNum`	Takes a clip number and set the clip to be the current one
	`GetCurrentClipNum`	Returns the current clip number
	`IsFirstClip`	Returns true if the current clip of the demo is the first one
	`IsLastClip`	Returns true if the current clip of the demo is the last one
	`AdvanceClip`	Advances a clip
	`ReverseClip`	Reverses a clip
	`SetNumCorrect`	Takes a number that represents the number of correct answers made by the user
	`SetNumIncorrect`	Takes a number that represents the number of incorrect answers made by the user

	`GetNumClips`	Returns the total number of quiz clips (questions)
	`GetNumCorrect`	Returns the number of correctly answered quiz questions
	`GetNumIncorrect`	Returns the number of incorrectly answered quiz questions
	`GetCurrentQuizQuestion`	Returns a pointer to the current quiz question of type QuizQuestion
	`ShowAnswer`	Displays the answer of the current quiz question
	`Show`	Displays the current clip of the quiz
QuizQuestion (abstract base)	`Submit`	Submits the answer
	`IsSubmitted`	Returns true if the answer is submitted
	`SetUserAnswer (pure virtual)`	Perform dynamic bindings through polymorphism
	`GetUserAnswer (pure virtual)`	
	`GetCorrectAnswer (pure virtual)`	
	`IsCorrect (pure virtual)`	
	`EnableInput (pure virtual)`	
	`DisableInput (pure virtual)`	
	`ShowAnswer (pure virtual)`	
	`Show (pure virtual)`	
MultipleChoice: QuizQuestion	`Constructor`	Takes a directory to the question index file and creates a new object
	`SetUserAnswer (virtual)`	Takes the user answer and stores it
	`GetCorrectAnswer (virtual)`	Returns the correct answer
	`EnableInput (virtual)`	Enables answer input
	`DisableInput (virtual)`	Disables answer input
	`IsCorrect (virtual)`	Returns true for a correct answer
	`ShowAnswer (virtual)`	Displays the correct answer
	`Show (virtual)`	Displays the quiz question
FillInTheBlanks: QuizQuestion	`Constructor`	Takes a directory to the question index file and creates a new object

	`GetNumBlanks`	Returns the total number of blanks
	`SetCurrentBlankNum`	Sets the blank number receiving input
	`GetCurrentBlankNum`	Returns the current blank number receiving input
	`AdvanceBlank`	Advances a blank for input
	`ReverseBlank`	Reverses a blank for input
	`IsFirstBlank`	Returns true if the current blank of the input is the first one
	`IsLastBlank`	Returns true if the current blank of the input is the last one
	Other member functions are the same as MultipleChoice:QuizType	
ShortQuestion: QuizQuestion	`Constructor`	Takes a directory to the question index file and creates a new object
	Other member functions are the same as MultipleChoice:QuizType	

Table 3.5 Object Representation for the CAI software.

The Tutorial object serves the purpose of displaying the appropriate tutorial pages. The constructor takes a programmer-specified default tutorial directory and a default sound directory for the corresponding index files. This provides flexibility on the locations of the tutorial materials. Also, the IsFirstClip and IsLastClip methods are used to determine if some of the navigator buttons need to be grayed out. The Show method displays the current tutorial clip.

The Demo object serves the purpose of displaying the appropriate demonstration pages. Its specification is very similar to the Tutorial object, except its member functions manipulate a different set of object-based components.

The Quiz object is a little bit complicated. It serves the purpose of managing the progress of the quiz and keeps its statistics. Beyond the common methods of the other two objects, it provides the AddQuestion method that adds a quiz question to its

database. Also, the GetNumCorrect and GetNumIncorrect methods keep track of how well the user is doing in answering the quiz questions.

The QuizQuestion object represents a quiz question. It is an abstract base class that serves dynamic binding over the objects MultipleChoice, FillInTheBlanks, and ShortQuestion. The SetUserAnswer and GetUserAnswer methods are used for storing and retrieving user answers. They are provided for recalling the incorrect answers given from the user in case he or she wants to see them again. The Show method displays the quiz question according to its type. Again, the design is fully expansible because we could define new types of quiz questions in the future.

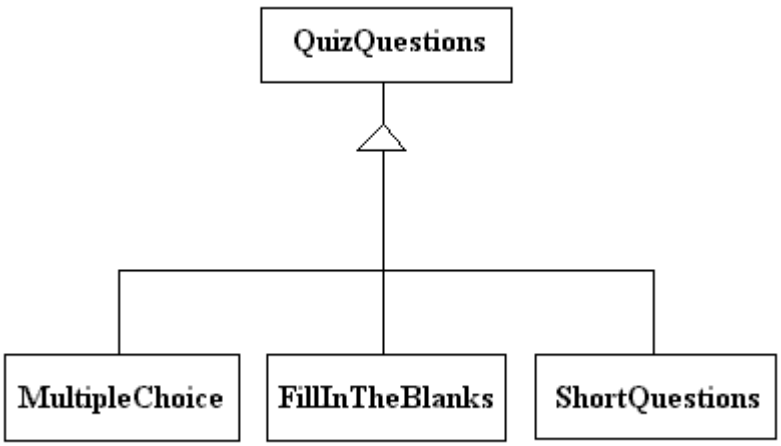

Figure 3.14 Class Hierarchy for the CAI Program.

3.4.7.5 Polymorphism and Virtual Functions

A major trick in building a flexible system is through the use of virtual functions to achieve polymorphism. The idea is that a pointer to a child class could be cast to a pointer to its base class. On the other hand, while this base class pointer is de-referenced to access the virtual member functions of the base class, the actual member functions called would be the corresponding member functions of its child classes, depending on the type of the pointer stored into the base class pointer.

In performing polymorphism, there must be at least one child object inherited from a base object. The member functions for both the base and the child class must be declared “virtual”. Usually, the base class serves just as an interface to the different child classes, and therefore the member functions are declared as “pure virtual”. This means the implementation of the member functions are left for its child. In this case, the base class is called an abstract based class, and no instances of it could be created.

3.4.7.6 Directory and File Format

The default directories for the tutorials, demonstrations, quizzes, and sounds databases are “Tutorials”, “Demos”, “Quizzes”, and “Sounds”, respectively. They are relative to the base directory where the CAI software is installed. Under the “Tutorial” directory, there is an index file named “index.dat” that stores the necessary information for a tutorial clip. An entry for the index file is as follows:

```
[Clip 6]

Content File:
Looping\Do-While Loop\Example of a Do-While Loop.rtf

Sound File:
Looping\s_06.wav

Relative Quiz Directory:
Looping

Relative Demo Directory:
Looping\Example of a Do-While Loop
```

Table 3.6 Tutorial Index File Format.

The clip starts from 0 to any size one could make. The paths given are relative to their default directories given in the constructors. For the relative quiz and demo directories, they further specify the paths of the index files needed to be further referenced.

Also, there is a “contents.dat” which stores the topics of presentation. It is merely used by the object-based component TtreeView in C++Builder:

```
Looping
        What is a Loop?
        How many different types of Loops are there in C++?
        While Loop
                Example of a While Loop
        Do-While Loop
                Example of a Do-While Loop
        For Loop
                Example of a For Loop
                More on For Loop
        Exiting a Loop
        Skipping a Loop Iteration
```

Table 3.7 The Contents of “contents.dat”.

In the quiz directory, there is an index file that lists the names of the data files for the quiz questions. This also tells the quantity of the quiz questions for a particular section. The data files of the quiz questions are of the form q_XX.dat. Here is an example of a quiz question data file:

```
Question File:
Quiz\Looping\Do-While Loop\q_02.rtf

Answer File:
Quiz\Looping\Do-While Loop\a_02.rtf

Possible type(s):
MultipleChoice
FillInTheBlanks
ShortQuestion

Correct answer(s):
after

Choices:
when
before
after
The condition is not tested
```

Table 3.8 Example of a Quiz Question Data File (q_XX.dat).

The “Question File” and “Answer File” entries tell the locations of the files. The example shown all the three possible types that could be made from this question, but any one or two of them could be removed.

The “Correct Answer(s)” entry specifies the correct answer in form of plain text. Finally, “Choices” specifies a list of choices that are used only by a multiple-choice question.

In the demo directory, there is also an index file that tells the locations of the image and sound files for each clip of the demonstrations. Here is an example:

```
[Clip 1]

Image File:
Example of a Do-While Loop_01.bmp

Sound File:
Looping\Do-While Loop\s_01.wav
```

Table 3.9 Example of a Tutorial Index File (index.dat).

The directories are structured in such a way that a separate directory is given for every different topic. This helps to mange the software better once its database gets larger, and moreover this means better expansibility.

3.4.7.7 Object-Oriented Design: Before and After

Before using object-oriented methodology, the program was developed using the object-based feature provided by C++Builder. Since there were no classes for classifying three different types of quiz questions (multiple-choice questions, fill in the blanks, and short questions), the traditional way of classifying was to use a switch statement. For example,

```
switch (GetType())  // type of quiz question
{
  // multiple-choice questions
  case MC:  FormMain->RadioGroupMC->BringToFront();
            FormMain->RadioGroupMC->Items->Clear();
```

```
            ...;
            break;

  // fill in the blanks
  case FIB: FormMain->GroupBoxFillIntheBlanks->BringToFront();
            ...;
            break;

  // short questions
  default:  ...;
            break;
}
```

Table 3.10 Procedural Object-Based Programming extracted from part of the Second Version of the code.

However, as the program grew, the number of statements for each case became larger and larger. Because the statements are loosely related, debugging such a program was a nightmare.

This may be one reason why the object-oriented programming paradigm is highly recommended for developing large programs. In the object-oriented paradigm, statements from different types are grouped into member functions naturally:

```
void MultipleChoice::Show() const
{
  FormMain->RadioGroupMC->BringToFront();
  FormMain->RadioGroupMC->Items->Clear();
  ...
}

void FillInTheBlanks::Show() const
{
  FormMain->GroupBoxFillIntheBlanks->BringToFront();
  ...
}

void ShortQuestion::Show() const
{
  ...
}
```

Table 3.11 Object-Oriented Programming Paradigm.

Suppose the classes MultipleChoice, FillInTheBlanks, and ShortQuestion are inherited from a common abstract base class, QuizQuestion. Then the statements:

```
QuizQuestion *q[3];

q[0] = (QuizQuestion *)new MultipleChoice();
q[1] = (QuizQuestion *)new FillInTheBlanks();
q[2] = (QuizQuestion *)new ShortQuestion();

for (int i = 0; i < 3; i++)
{
  q[i]->Show();
}
```

Table 3.12 Polymorphism.

will call the corresponding Show functions according to their original types. This is called polymorphism because the same variable (q[i]) is used, but different methods are invoked.

4 RESULTS

4.1 Evaluating the Program

4.1.1 Introduction

Once the program was developed, it was evaluated by ten volunteers. In order to obtain useful feedback from such a small group of participants, an additional observation was made on how the people interacted with the system. This was a simplified usability test in which each user was encourage to verbalize his or her thoughts about the program as it was being used, and the observer looked at how the user interacted with the system. This is called "simplified" because the test process was not recorded.

The test was conducted using the participants' home computers so as to minimize the side effect generated by unfamiliar systems. It was also important to make sure the CAI program could run on all computers that use Windows.

After using the program, a survey was given to each participant [Table 4.1]. This was a paper survey created using Microsoft Word 2000. Again, the survey was limited to 10 questions so as to not overwhelm the participants. These questions require only numerical answers from 1 to 10 representing the levels of agreement on certain statements. There was a "bonus" question at the end to let the participants provide other opinions about the program.

Program Evaluation Form

For the following, rate from (strongly disagree) 1 to 10 (strongly agree):

1.	The presentation of information is clear.	_____
2.	The aim of the lesson is clear.	_____
3.	The program content is accurate.	_____
4.	The program is easy to use.	_____
5.	The interface is attractive.	_____
6.	The program provides good interaction.	_____

7.	The presentation rate and sequence could be controlled.	_____
8.	The quiz questions help to reinforce the material being taught.	_____
9.	I like using the program to learn C++ rather than reading a book.	_____
10.	The overall quality is good.	_____

Do you have other opinions?

Table 4.1 Computer-Aided Instruction (CAI) Survey.

4.1.2 Results

The following table summarizes the results from the 10 respondents:

Respondent / Aspect	1	2	3	4	5	6	7	8	9	10	**Average**
Clear presentation	8	8	(7)	(10)	7	9	10	8	7	9	**8.25**
Aim of lesson	8	8	9	(10)	10	8	10	(7)	8	9	**8.75**
Accurate content	9	9	-	(10)	10	10	10	-	(8)	-	**9.60**
Easy to use	8	9	(10)	10	10	7	10	8	(6)	7	**8.63**
Attractive interface	8	(2)	7.5	9	6	7	(10)	5	7	6	**6.94**
Good interaction	9	(5)	(10)	10	6.5	7	10	6	5	7	**7.56**
Controllable presentation rate and sequence	8	(7)	-	(10)	9	10	10	8	7	8	**8.57**
Questions reinforce materials	8	8	(10)	10	8.5	10	10	(6)	7	10	**8.94**
Like using the program to learn than books	8	8	9	(10)	10	10	10	8	(4)	9	**9.00**
Overall quality	9	8	9	(10)	(7)	8	10	7	7	8.5	**8.31**

***: Notes

- In each cell of the table (except experience), the range of the score is from 1 to 10 inclusive. A higher score means higher user satisfaction.
- There were 4 males and 6 females in the population, aged 19 to 23.

Legend:

(n): Extreme data. The high and low values were discarded in calculating the average score for that category.
-: no response given

Table 4.2 Results on Program Evaluation Form.

The program scored the highest in accuracy of content. Most respondents agreed that the drill and practice questions helped them to reinforce the materials being taught.

On the other hand, the program was relatively weak in the areas of user interface and interaction. Luckily the average scores for these categories were still promising. One problem found during the observation was that people got confused switching between the three components of the CAI programs (tutorial, demo, and quiz sessions).

Besides the above evaluation results, respondents also gave extra comments. In general, they agreed that the interface was simple and user-friendly. One respondent suggested the use of more color, and two respondents suggested the separation of long paragraphs into shorter paragraphs.

The respondents reacted favorably to the audio aspect of the program. This showed that the inclusion of the voice feature could catch the user's attention.

4.1.3 Conclusion

The program generally scored high in many aspects, but there is still room for improvement. From the survey results, a number of improvements were made to the interface. For example, color was added to indicate important keywords and sentences. Also, longer paragraphs were broken into sentences to further minimize the memory load of the user [Section 3.4.6.5].

The results also showed that it is difficult to make a CAI program with sufficient interaction. This may be one of the reasons why people prefer to learn from lectures.

In programming, it takes time to incorporate more interaction because it requires a lot of multimedia presentations.

5 CONCLUSIONS

5.1 Conclusions and Possible Extensions

By studying the criteria that contribute to the effectiveness of CAI programs, a fairly effective and extensible CAI program has been developed. The three components of the program, tutorials, demonstrations, and quizzes, could serve as a backbone to most instructional materials. With some knowledge about the basic structure of the system, the contents of the tutorials could be easily extended in a cost effective way.

Also, this project has successfully incorporated knowledge about learning theories into the design of a more effective CAI program. This provides a good basis for the future development of CAI programs.

There are many possible extensions to this project. Due to time constraints, only a small subset of the C++ language was covered. If there were more tutorial materials, the program would be more useful. Also, many more quiz questions could be added so that the user could have more practice using what they have learned.

Better graphics could lead to a better interface, and better animation would provide added interest for visual learners.

As far as the Internet is concerned, the program could update itself from the Web. The quiz questions could be updated periodically so that they would be more effective. In addition, the materials could be accessed while the computer is offline.

In conclusion, if more effort is put into relating the design of CAI software to the way people learn, CAI can revolutionize the way people acquire knowledge.

APPENDIX A – Survey Results

Survey on CAI Program Design (Results):

Number of Response = 19

Question 1:
==========
Number of people who have used CAI programs before = 4

Question 2:
==========
How much they think they have learned from those programs (out of 10):
4, 2, 3, 6

Question 3:
==========
Features that people like/dislike most about those CAI programs:
- Colorful, the activities are quite interesting
- Don't know, but dislike.
- It includes many visual representation ..... and there is a game about testing how much I learnt from the course.
- User unfriendly.

Question 4 & 5:
==============

Number of people

	most preferred	top 2 preferred	last 2 preferred	least preferred
Book	6	15	13	3
Course	9	13	9	6
Computer	4	10	15	10
Other				
Notes	0	0	1	0
	most important	**top 2 important**	**last 2 important**	**least important**
System				
Easy to use	15	19	16	4
Performance	4	18	19	15
Other				
Memory Usage	0	0	1	1
Self-explanatory	0	1	1	0
Useful	0	0	1	1
Presentation				
Text layout	10	13	4	2
Color	2	6	8	6
Graphics	5	10	1	1
Sound	0	1	18	8
Animation	2	7	10	5
Tutorial Contents				
Quantity	3	5	14	12
Easy to understand	9	15	4	2
Meaningfully structured	5	11	8	3
Accuracy	2	6	13	4
Drill and Practice Exercises				
Quantity	4	5	14	9
Revise learned materials	7	13	6	2
Encourage thinking	7	14	5	1
Feedback	1	5	14	9

Question 6:
==========
What types of interaction could a CAI program provide so that it consistently maintains the user's interest?

- live chat
- feedback for user's performance
- more animation and graphics
- Friendly environment
- It's better to provide more update information. It can also link to more update related websites for providing supplementary information.
- Simple and consistent interface.

- Easy to navigate.
- some q&a between sessions... or interesting animations...

Question 7:
==========
How do you think the tutorial materials could be organized so that they are more interesting than the material in a book?

- More colors, graphics (both moving and stationary), fonts, etc.
- Not really since book and computer is diff media! maybe better layout
- using animation coz this can't be available in books
- interactive between instuctor and audiences
- Easy to use and easy to understand the ideas behind it.
- less words, more practice
- After each part of content, it needs to have concise and precise summmary with many short questions for revising (e.g. MCQ). Also, it is better including easily available glossary for difficult words.
- Good interactive, else there is no difference than a book with extra meaningless animation.
- customary response and feedback to answers and problems

Question 8:
==========
Some experts think that the drill and practice questions in CAI software are ineffective. Do you agree? Why? How would you suggest the questions be designed so that they become more effective?

- Agree to a certain extent.
- Don't understand the materials thoroughly when attempting the questions.
- More applicable and clear examples given might help.
- People not willing to think... If guide by a tutor... much better... also the diff of media.....
- I disagree. but no idea about how to become more effective.
- i don't agree becoz whether the questions are effective or not depends on the program producer instead on the program media
- Less pages, and more summarization with topics and ideas CAI trying to deliver to viewers, so it will be easier to read. Reading on a computer will hurt you eyes in the long run. Further a book is more accessible than with a computer. It's lighter and it's not as difficult as setting up a computer to on-line.
- I don't think that's ineffective. For example, in science subjects, books cannot provide interactive medium for us to study such as 3D graphs, concise Q and A for revision. For those Q, it may be designed into many levels. Then, we can have more chance as self-learning and improvement.

Question 9:
==========
Reasonable Price of a CAI program (in US dollars):

	Number of people
$0, it should be free	2
>$0 to $10	2
>$10 to $20	5
>$20 to $30	4
>$30 to $50	0
>$50 to $100	3
>$100	2
No response	1

Question 10:
===========
What else would you like to see in the future CAI programs?

- no big difference from attending a class
- VR technique is added
- CAI over the internet/mobile networks
- More graphics, and animations
- More scientific CAI programs can be available since on the market, only more basic level CAI can be provided. Also, it's better to have it's own website so that updated information can be provided.

APPENDIX B – SOURCE CODE (Learning C++)

About.h

```
//---------------------------------------------------------------------------
#ifndef AboutH
#define AboutH
//---------------------------------------------------------------------------
#include <vcl\System.hpp>
#include <vcl\Windows.hpp>
#include <vcl\SysUtils.hpp>
#include <vcl\Classes.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Controls.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Buttons.hpp>
#include <vcl\ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TFormAbout : public TForm
{
__published:
        TPanel *Panel1;
        TImage *ProgramIcon;
        TLabel *ProductName;
        TLabel *Version;
        TLabel *Copyright;
        TLabel *Comments;
  TButton *ButtonOK;
  TLabel *Label2;
  TLabel *Label1;
  TBevel *Bevel1;
  TBevel *Bevel2;
  TLabel *Label3;
  TLabel *Label4;
  void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
  void __fastcall FormActivate(TObject *Sender);
private:
public:
        virtual __fastcall TFormAbout(TComponent* AOwner);
};
//---------------------------------------------------------------------------
extern PACKAGE TFormAbout *FormAbout;
//---------------------------------------------------------------------------
#endif
```

Demo.h

```
//---------------------------------------------------------------------------
#ifndef DemoH
#define DemoH
//---------------------------------------------------------------------------
#include <vector>
using namespace std;
//---------------------------------------------------------------------------
class Demo
{
  private:
    AnsiString Directory;
    AnsiString SoundDirectory;
    int NumClips;
    int CurrentClipNum;
    vector <AnsiString> viLocations;
    vector <AnsiString> viSoundLocations;

  public:
    // takes a default demo directory and a default sound directory
    Demo(AnsiString directory, AnsiString soundDirectory);

    // returns the default demo directory
    AnsiString GetDirectory() const;

    // returns the total number of clips for this demo
    int GetNumClips() const;

    // takes a clip number and set the clip to be the current one
    bool SetCurrentClipNum(int currentClipNum);

    // returns the current clip number
    int GetCurrentClipNum() const;

    // advance a clip
    bool AdvanceClip();

    // reverse a clip
    bool ReverseClip();

    // returns true if the current clip of the demo is the first one
    bool IsFirstClip() const;

    // returns true if the current clip of the demo is the last one
    bool IsLastClip() const;

    // displays the current clip of the demo
    void Show() const;
};
//---------------------------------------------------------------------------
#endif
```

FillInTheBlanks.h

```
//---------------------------------------------------------------------------
#ifndef FillInTheBlanksH
#define FillInTheBlanksH
//---------------------------------------------------------------------------
#include "QuizQuestion.h"
//---------------------------------------------------------------------------
class FillInTheBlanks: public QuizQuestion
{
  private:
    AnsiString QuestionFile;
    AnsiString AnswerFile;
    AnsiString CorrectAnswer;
    AnsiString UserAnswer;
    int NumBlanks;
    int CurrentBlankNum;

  public:
    FillInTheBlanks(AnsiString indexFile);

    // returns the total number of blanks
    int GetNumBlanks() const;

    // sets the blank number receiving input
    bool SetCurrentBlankNum(int currentClipNum);

    // returns the current blank number receiving input
    int GetCurrentBlankNum() const;

    // advances a blank for input
    bool AdvanceBlank();

    // reverses a blank for input
    bool ReverseBlank();

    // returns true if the current blank of the input is the first one
    bool IsFirstBlank() const;

    // returns true if the current blank of the input is the last one
    bool IsLastBlank() const;

    // returns the i-th blank of the user answer
    AnsiString GetUserAnswer(int index) const;

    // takes the i-th blank of the user answer and stores it
    void SetUserAnswer(int index, AnsiString userAnswer);

    // stores the given user answer
    virtual void SetUserAnswer(AnsiString userAnswer);

    // takes the user answer and stores it
    virtual AnsiString GetUserAnswer() const;

    // returns the correct answer
    virtual AnsiString GetCorrectAnswer() const;

    // enables answer input
    virtual void EnableInput() const;

    // disable answer input
    virtual void DisableInput() const;

    // returns true for a correct answer, false otherwise
    virtual bool IsCorrect() const;

    // displays the correct answer
    virtual void ShowAnswer() const;

    // displays the quiz question
    virtual void Show() const;
};
//---------------------------------------------------------------------------
#endif
```

Main.h

```
//---------------------------------------------------------------------------
#ifndef MainH
#define MainH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <Menus.hpp>
#include <ToolWin.hpp>
#include <ExtCtrls.hpp>
#include <Buttons.hpp>
#include <MPlayer.hpp>
//---------------------------------------------------------------------------
class TFormMain : public TForm
{
__published:   // IDE-managed Components
  TPanel *PanelTutorial;
  TSplitter *Splitter1;
  TPanel *PanelTutorialWindow;
  TPanel *Panel3;
  TPanel *Panel6;
  TBitBtn *BitBtnTutorialPrevious;
  TBitBtn *BitBtnTutorialNext;
  TBitBtn *BitBtnTutorialDemo;
  TBitBtn *BitBtnTutorialQuiz;
  TRichEdit *RichEditTutorial;
  TGroupBox *GroupBox1;
  TTreeView *TreeViewContents;
  TMainMenu *MainMenu1;
  TMenuItem *mnuFile;
  TMenuItem *mnuFileExit;
  TMenuItem *mnuHelp;
  TPanel *PanelQuiz;
  TPanel *PanelDemo;
  TPanel *Panel4;
  TPanel *Panel5;
  TBitBtn *BitBtnDemoPrevious;
  TBitBtn *BitBtnDemoNext;
  TBitBtn *BitBtnDemoBack;
  TPanel *Panel2;
  TImage *ImageDemo;
  TStatusBar *StatusBarMain;
  TPanel *Panel1;
  TPanel *Panel7;
  TBitBtn *BitBtnQuizPrevious;
  TBitBtn *BitBtnQuizNext;
  TBitBtn *BitBtnQuizBack;
  TMenuItem *mnuHelpAbout;
  TMenuItem *mnuOptions;
  TMenuItem *mnuOptionsSound;
  TPanel *Panel8;
  TPanel *Panel9;
  TPanel *Panel10;
  TPanel *PanelUserAnswer;
  TPanel *Panel11;
  TImage *ImageQuizResult;
  TBitBtn *BitBtnQuizSubmit;
  TPanel *Panel12;
  TGroupBox *GroupBoxShortQuestion;
  TMemo *MemoShortQuestion;
  TRadioGroup *RadioGroupMultipleChoice;
  TGroupBox *GroupBoxFillIntheBlanks;
  TEdit *EditFillInTheBlanks;
  TBitBtn *BitBtnFillInTheBlanksUp;
  TBitBtn *BitBtnFillInTheBlanksDown;
  TRichEdit *RichEditQuiz;
  TMediaPlayer *MediaPlayer;
  TBitBtn *BitBtnQuizAnswer;
  TGroupBox *GroupBoxAnswer;
  TRichEdit *RichEditAnswer;
  TSplitter *Splitter2;
```

```
  void __fastcall BitBtnTutorialPreviousClick(TObject *Sender);
  void __fastcall BitBtnTutorialNextClick(TObject *Sender);
  void __fastcall FormCreate(TObject *Sender);
  void __fastcall TreeViewContentsChange(TObject *Sender, TTreeNode *Node);
  void __fastcall FormActivate(TObject *Sender);
  void __fastcall BitBtnTutorialDemoClick(TObject *Sender);
  void __fastcall BitBtnDemoBackClick(TObject *Sender);
  void __fastcall BitBtnDemoNextClick(TObject *Sender);
  void __fastcall BitBtnDemoPreviousClick(TObject *Sender);
  void __fastcall FormClose(TObject *Sender, TCloseAction &Action);
  void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
  void __fastcall mnuFileExitClick(TObject *Sender);
  void __fastcall BitBtnQuizBackClick(TObject *Sender);
  void __fastcall RadioGroupMultipleChoiceClick(TObject *Sender);
  void __fastcall BitBtnQuizSubmitClick(TObject *Sender);
  void __fastcall BitBtnQuizNextClick(TObject *Sender);
  void __fastcall BitBtnQuizPreviousClick(TObject *Sender);
  void __fastcall MemoShortQuestionExit(TObject *Sender);
  void __fastcall BitBtnFillInTheBlanksDownClick(TObject *Sender);
  void __fastcall BitBtnFillInTheBlanksUpClick(TObject *Sender);
  void __fastcall BitBtnTutorialQuizClick(TObject *Sender);
  void __fastcall mnuOptionsSoundClick(TObject *Sender);
  void __fastcall mnuHelpAboutClick(TObject *Sender);
  void __fastcall BitBtnQuizAnswerClick(TObject *Sender);
  void __fastcall EditFillInTheBlanksKeyDown(TObject *Sender, WORD &Key,
          TShiftState Shift);
  void __fastcall EditFillInTheBlanksChange(TObject *Sender);
  void __fastcall EditFillInTheBlanksEnter(TObject *Sender);
  void __fastcall MemoShortQuestionEnter(TObject *Sender);
private:       // User declarations
public:        // User declarations
  __fastcall TFormMain(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TFormMain *FormMain;
//---------------------------------------------------------------------------
#endif
```

MultipleChoice.h

```
//---------------------------------------------------------------------------
#ifndef MultipleChoiceH
#define MultipleChoiceH
//---------------------------------------------------------------------------
#include <vector>
using namespace std;
//---------------------------------------------------------------------------
#include "QuizQuestion.h"
//---------------------------------------------------------------------------
class MultipleChoice: public QuizQuestion
{
  private:
    AnsiString QuestionFile;
    AnsiString AnswerFile;
    vector<AnsiString> viChoices;
    AnsiString CorrectAnswer;
    AnsiString UserAnswer;

  public:
    // takes a directory to the question index file and creates a new object
    MultipleChoice(AnsiString indexFile);

    // stores the given user answer
    virtual void SetUserAnswer(AnsiString userAnswer);

    // takes the user answer and stores it
    virtual AnsiString GetUserAnswer() const;

    // returns the correct answer
    virtual AnsiString GetCorrectAnswer() const;

    // enables answer input
    virtual void EnableInput() const;

    // disables answer input
    virtual void DisableInput() const;

    // returns true for a correct answer
    virtual bool IsCorrect() const;

    // displays the correct answer
    virtual void ShowAnswer() const;

    // displays the quiz question
    virtual void Show() const;
};
//---------------------------------------------------------------------------
#endif
```

Quiz.h

```
//---------------------------------------------------------------------------
#ifndef QuizH
#define QuizH
//---------------------------------------------------------------------------
#include <vector>
using namespace std;
//---------------------------------------------------------------------------
#include "QuizQuestion.h"
//---------------------------------------------------------------------------
class Quiz
{
  private:
    int CurrentClipNum;
    int NumCorrect;
    int NumIncorrect;
    vector<QuizQuestion *> viQuizQuestions;
    AnsiString GetRandomQuestionType(AnsiString dataFile) const;

  public:
    // takes a default quiz directory and the number of questions in the quiz
    Quiz(AnsiString Directory, int numQuestions);

    // used to deallocate quiz questions
    ~Quiz();

    // take a pointer to QuizQuestion and add the question being pointed to
    void AddQuestion(QuizQuestion *quizQuestion);

    // takes a clip number and set the clip to be the current one
    bool SetCurrentClipNum(int currentClipNum);

    // returns the current clip number
    int GetCurrentClipNum() const;

    // returns true if the current clip of the demo is the first one
    bool IsFirstClip() const;

    // returns true if the current clip of the demo is the last one
    bool IsLastClip() const;

    // advance a clip
    bool AdvanceClip();

    // reverse a clip
    bool ReverseClip();

    // takes a number that represents the number of correct answers made by the user
    void SetNumCorrect(int numCorrect);

    // takes a number that represents the number of incorrect answers made by the user
    void SetNumIncorrect(int numIncorrect);

    // returns the total number of quiz clips (questions)
    int GetNumClips() const;

    // returns the number of correctly answered quiz questions
    int GetNumCorrect() const;

    // returns the number of incorrectly answered quiz questions
    int GetNumIncorrect() const;

    // returns a pointer to the current quiz question of type QuizQuestion
    QuizQuestion *GetCurrentQuizQuestion() const;

    // displays the answer of the current quiz question
    void ShowAnswer() const;

    // displays the current clip of the quiz
    void Show() const;
};
//---------------------------------------------------------------------------
#endif
```

QuizMenu.h

```
//---------------------------------------------------------------------------
#ifndef QuizMenuH
#define QuizMenuH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TFormQuizMenu : public TForm
{
__published:   // IDE-managed Components
  TPanel *Panel13;
  TLabel *Label1;
  TLabel *Label2;
  TLabel *Label3;
  TUpDown *UpDownQuizNumQuestions;
  TEdit *EditQuizNumQuestions;
  TButton *ButtonGo;
  TButton *ButtonCancel;
  void __fastcall ButtonCancelClick(TObject *Sender);
  void __fastcall ButtonGoClick(TObject *Sender);
  void __fastcall EditQuizNumQuestionsExit(TObject *Sender);
  void __fastcall FormActivate(TObject *Sender);
private:       // User declarations
public:        // User declarations
  __fastcall TFormQuizMenu(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TFormQuizMenu *FormQuizMenu;
//---------------------------------------------------------------------------
#endif
```

QuestionQuestion.h

```
//---------------------------------------------------------------------------
#ifndef QuizQuestionH
#define QuizQuestionH
//---------------------------------------------------------------------------
class QuizQuestion
{
  private:
    bool Submitted;

  public:
    QuizQuestion();

    // submits the answer
    void Submit();

    // returns true if the answer is submitted
    bool IsSubmitted() const;

    // virtuak functions for dynamic bindings
    virtual void SetUserAnswer(AnsiString userAnswer) = 0;
    virtual AnsiString GetUserAnswer() const = 0;
    virtual AnsiString GetCorrectAnswer() const = 0;
    virtual bool IsCorrect() const = 0;
    virtual void EnableInput() const = 0;
    virtual void DisableInput() const = 0;
    virtual void ShowAnswer() const = 0;
    virtual void Show() const = 0;
};
//---------------------------------------------------------------------------
#endif
```

QuizResults.h

```
//---------------------------------------------------------------------------
#ifndef QuizResultsH
#define QuizResultsH
//---------------------------------------------------------------------------
#include <vcl\System.hpp>
#include <vcl\Windows.hpp>
#include <vcl\SysUtils.hpp>
#include <vcl\Classes.hpp>
#include <vcl\Graphics.hpp>
#include <vcl\Forms.hpp>
#include <vcl\Controls.hpp>
#include <vcl\StdCtrls.hpp>
#include <vcl\Buttons.hpp>
#include <vcl\ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TFormQuizResults : public TForm
{
__published:
  TLabel *Label1;
  TLabel *Label2;
  TButton *ButtonOK;
  TLabel *Label3;
  TEdit *EditTotalNumQuestions;
  TLabel *Label4;
  TEdit *EditNumCorrectAnswers;
  TEdit *EditNumIncorrectAnswers;
  TLabel *Label5;
  TEdit *EditNumOmittedQuestions;
  TLabel *Label6;
  TLabel *Label7;
  TLabel *Label8;
  TLabel *LabelGrade;
  TLabel *LabelScore;
  void __fastcall FormActivate(TObject *Sender);
private:
public:
        virtual __fastcall TFormQuizResults(TComponent* AOwner);
};
//---------------------------------------------------------------------------
extern PACKAGE TFormQuizResults *FormQuizResults;
//---------------------------------------------------------------------------
#endif
```

Shared.h

```
//---------------------------------------------------------------------------
#ifndef SharedH
#define SharedH
//---------------------------------------------------------------------------
// maximum number of characters per line
const int MAX_NUMCHARS = 256;
//---------------------------------------------------------------------------
#endif
```

ShortQuestion.h

```
//---------------------------------------------------------------------------
#ifndef ShortQuestionH
#define ShortQuestionH
//---------------------------------------------------------------------------
#include "QuizQuestion.h"
//---------------------------------------------------------------------------
class ShortQuestion: public QuizQuestion
{
  private:
    AnsiString QuestionFile;
    AnsiString AnswerFile;
    AnsiString CorrectAnswer;
    AnsiString UserAnswer;

  public:
    // takes a directory to the question index file and creates a new object
    ShortQuestion(AnsiString indexFile);

    // stores the given user answer
    virtual void SetUserAnswer(AnsiString userAnswer);

    // takes the user answer and stores it
    virtual AnsiString GetUserAnswer() const;

    // returns the correct answer
    virtual AnsiString GetCorrectAnswer() const;

    // enables answer input
    virtual void EnableInput() const;

    // disables answer input
    virtual void DisableInput() const;

    // returns true for a correct answer
    virtual bool IsCorrect() const;

    // displays the correct answer
    virtual void ShowAnswer() const;

    // displays the quiz question
    virtual void Show() const;
};
//---------------------------------------------------------------------------
#endif
```

Tutorial.h

```
//---------------------------------------------------------------------------
#ifndef TutorialH
#define TutorialH
//---------------------------------------------------------------------------
#include <vector>
using namespace std;
//---------------------------------------------------------------------------
class Tutorial
{
  private:
    AnsiString Directory;
    AnsiString SoundDirectory;
    int NumClips;
    int CurrentClipNum;
    vector <AnsiString> viFileLocations;
    vector <AnsiString> viSoundLocations;
    vector <AnsiString> viQuizIndexes;
    vector <AnsiString> viDemoIndexes;

  public:
    // takes a default tutorial directory and a default sound directory
    Tutorial(AnsiString directory, AnsiString soundDirectory);

    // returns the default tutorial directory
    AnsiString GetDirectory() const;

    // gets the name of the index file for the i-th demo clip
    AnsiString GetDemoIndexDirectoryAt(int i) const;

    // gets the name of the index file for the i-th quiz clip
    AnsiString GetQuizIndexDirectoryAt(int i) const;

    // returns the total number of clips for this Tutorial
    int GetNumClips() const;

    // takes a clip number and sets the clip to be the current one
    bool SetCurrentClipNum(int currentClipNum);

    // returns the current clip number
    int GetCurrentClipNum() const;

    // advance a clip
    bool AdvanceClip();

    // reverse a clip
    bool ReverseClip();

    // returns true if the current clip of the tutorial is the first one
    bool IsFirstClip() const;

    // returns true if the current clip of the tutorial is the last one
    bool IsLastClip() const;

    // displays the current clip of the tutorial
    void Show() const;
};
//---------------------------------------------------------------------------
#endif
```

About.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include "About.h"
#include "Main.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TFormAbout *FormAbout;
//---------------------------------------------------------------------------
__fastcall TFormAbout::TFormAbout(TComponent* AOwner)
        : TForm(AOwner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFormAbout::FormClose(TObject *Sender,
      TCloseAction &Action)
{
  FormMain->MediaPlayer->Pause();
}
//---------------------------------------------------------------------------
void __fastcall TFormAbout::FormActivate(TObject *Sender)
{
  FormMain->MediaPlayer->Pause();
}
//---------------------------------------------------------------------------
```

Demo.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include <fstream>
using namespace std;
//---------------------------------------------------------------------------
#include "Demo.h"
#include "Shared.h"
#include "Main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
//---------------------------------------------------------------------------
Demo::Demo(AnsiString directory, AnsiString soundDirectory)
{
  char line[MAX_NUMCHARS];
  NumClips = 0;

  Directory = directory;
  SoundDirectory = soundDirectory;
  CurrentClipNum = 0;

  fstream fsIndex((Directory + "\\index.dat").c_str());

  // how big is the index file?
  while (!fsIndex.eof())
  {
    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);

    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    viLocations.insert(viLocations.end(), AnsiString(line));

    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    viSoundLocations.insert(viSoundLocations.end(), AnsiString(line));

    fsIndex.getline(line, MAX_NUMCHARS);

    NumClips++;
  }

  fsIndex.close();
}
//---------------------------------------------------------------------------
AnsiString Demo::GetDirectory() const
{
  return Directory;
}
//---------------------------------------------------------------------------
int Demo::GetNumClips() const
{
  return NumClips;
}
//---------------------------------------------------------------------------
bool Demo::SetCurrentClipNum(int currentClipNum)
{
  // invalid clip number
  if (currentClipNum < 0 || currentClipNum >= NumClips)
    return false;

  CurrentClipNum = currentClipNum;

  return true;
}
//---------------------------------------------------------------------------
int Demo::GetCurrentClipNum() const
{
  return CurrentClipNum;
}
```

```
//---------------------------------------------------------------------------
bool Demo::AdvanceClip()
{
  if (IsLastClip() == true)
    return false;

  SetCurrentClipNum(GetCurrentClipNum() + 1);

  return true;
}
//---------------------------------------------------------------------------
bool Demo::ReverseClip()
{
  if (IsFirstClip() == true)
    return false;

  SetCurrentClipNum(GetCurrentClipNum() - 1);

  return true;
}
//---------------------------------------------------------------------------
bool Demo::IsFirstClip() const
{
  if (CurrentClipNum == 0)
    return true;

  return false;
}
//---------------------------------------------------------------------------
bool Demo::IsLastClip() const
{
  if (CurrentClipNum == NumClips - 1)
    return true;

  return false;
}
//---------------------------------------------------------------------------
void Demo::Show() const
{
  FormMain->ImageDemo->Picture->LoadFromFile(Directory + "\\" +
viLocations[CurrentClipNum]);
  FormMain->StatusBarMain->SimpleText = AnsiString("Clip ") + (GetCurrentClipNum() + 1)
+ " of " + GetNumClips();

  if (IsFirstClip() == true)
  {
    FormMain->BitBtnDemoPrevious->Enabled = false;
    FormMain->BitBtnDemoNext->Enabled = true;
  }
  else if (IsLastClip() == true)
  {
    FormMain->BitBtnDemoPrevious->Enabled = true;
    FormMain->BitBtnDemoNext->Enabled = false;
  }
  else
  {
    FormMain->BitBtnDemoPrevious->Enabled = true;
    FormMain->BitBtnDemoNext->Enabled = true;
  }

  // play sound if it is enabled
  if (FormMain->mnuOptionsSound->Checked && viSoundLocations[CurrentClipNum] != "")
  {
    FormMain->MediaPlayer->FileName = SoundDirectory + "\\" +
viSoundLocations[CurrentClipNum];
    FormMain->MediaPlayer->Open();
    FormMain->MediaPlayer->Play();
  }
}
//---------------------------------------------------------------------------
```

FillInTheBlanks.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include <fstream>
using namespace std;
//---------------------------------------------------------------------------
#include "FillInTheBlanks.h"
#include "Shared.h"
#include "Main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
//---------------------------------------------------------------------------
FillInTheBlanks::FillInTheBlanks(AnsiString indexFile)
{
  char line[MAX_NUMCHARS];

  CurrentBlankNum = 0;

  fstream fsIndex(indexFile.c_str());

  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);

  // location of the question file
  QuestionFile = AnsiString(line);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);

  // location of the answer file
  AnswerFile = AnsiString(line);
  fsIndex.getline(line, MAX_NUMCHARS);

  do  // skip possible type(s) data
  {
    fsIndex.getline(line, MAX_NUMCHARS);
  } while (AnsiString(line) != "");

  fsIndex.getline(line, MAX_NUMCHARS);

  NumBlanks = 0;

  // read correct answer
  fsIndex.getline(line, MAX_NUMCHARS);

  while (AnsiString(line) != "")
  {
    CorrectAnswer += AnsiString(line) + (char)13 + (char)10;
    NumBlanks++;
    fsIndex.getline(line, MAX_NUMCHARS);
  }

  fsIndex.close();
}
//---------------------------------------------------------------------------
int FillInTheBlanks::GetNumBlanks() const
{
  return NumBlanks;
}
//---------------------------------------------------------------------------
bool FillInTheBlanks::SetCurrentBlankNum(int currentBlankNum)
{
  // invalid blank number
  if (currentBlankNum < 0 || currentBlankNum >= NumBlanks)
    return false;

  CurrentBlankNum = currentBlankNum;

  return true;
}
//---------------------------------------------------------------------------
int FillInTheBlanks::GetCurrentBlankNum() const
```

```
{
  return CurrentBlankNum;
}
//---------------------------------------------------------------------------
bool FillInTheBlanks::AdvanceBlank()
{
  if (IsLastBlank() == true)
    return false;

  SetCurrentBlankNum(GetCurrentBlankNum() + 1);

  return true;
}
//---------------------------------------------------------------------------
bool FillInTheBlanks::ReverseBlank()
{
  if (IsFirstBlank() == true)
    return false;

  SetCurrentBlankNum(GetCurrentBlankNum() - 1);

  return true;
}
//---------------------------------------------------------------------------
bool FillInTheBlanks::IsFirstBlank() const
{
  if (CurrentBlankNum == 0)
    return true;

  return false;
}
//---------------------------------------------------------------------------
bool FillInTheBlanks::IsLastBlank() const
{
  if (CurrentBlankNum == NumBlanks - 1)
    return true;

  return false;
}
//---------------------------------------------------------------------------
void FillInTheBlanks::SetUserAnswer(int index, AnsiString userAnswer)
{
  AnsiString s;

  for (int blankNum = 0; blankNum < GetNumBlanks(); blankNum++)
    if (blankNum == index)  // need to replace?
      s += userAnswer/*.Trim()*/ + (char)13 + (char)10;
    else
      s += GetUserAnswer(blankNum) + (char)13 + (char)10;

  UserAnswer = s;
}
//---------------------------------------------------------------------------
AnsiString FillInTheBlanks::GetUserAnswer(int index) const
{
  int left, right;

  for (left = 0, right = left; right < UserAnswer.Length(); right++)
    // check for line feed
    if (UserAnswer.SubString(right, 1) == (char)10)
    {
      if (index-- == 0)
        return UserAnswer.SubString(left, right - left - 1).Trim();

      left = right + 1;
    }

  return UserAnswer.SubString(left, right - left).Trim();
}
//---------------------------------------------------------------------------
void FillInTheBlanks::SetUserAnswer(AnsiString userAnswer)
{
  UserAnswer = userAnswer;
}
//---------------------------------------------------------------------------
AnsiString FillInTheBlanks::GetUserAnswer() const
{
```

```
  return UserAnswer;
}
//---------------------------------------------------------------------------
AnsiString FillInTheBlanks::GetCorrectAnswer() const
{
  return CorrectAnswer;
}
//---------------------------------------------------------------------------
void FillInTheBlanks::EnableInput() const
{
  FormMain->EditFillInTheBlanks->ReadOnly = false;
}
//---------------------------------------------------------------------------
void FillInTheBlanks::DisableInput() const
{
  FormMain->EditFillInTheBlanks->ReadOnly = true;
}
//---------------------------------------------------------------------------
bool FillInTheBlanks::IsCorrect() const
{
  return GetUserAnswer().Trim() == GetCorrectAnswer().Trim();
}
//---------------------------------------------------------------------------
void FillInTheBlanks::ShowAnswer() const
{
  FormMain->RichEditAnswer->Lines->LoadFromFile(AnswerFile);
}
//---------------------------------------------------------------------------
void FillInTheBlanks::Show() const
{
  FormMain->GroupBoxFillIntheBlanks->BringToFront();
  FormMain->RichEditQuiz->Lines->LoadFromFile(QuestionFile);

  // Show the current blank
  for (int i = FormMain->RichEditQuiz->FindText("_", 0, FormMain->RichEditQuiz-
>Text.Length(), TSearchTypes() << stMatchCase), j = 0; i != -1; i = FormMain-
>RichEditQuiz->FindText("_", i + 5, FormMain->RichEditQuiz->Text.Length(),
TSearchTypes() << stMatchCase), j++)
  {
    FormMain->RichEditQuiz->SelStart = i;
    FormMain->RichEditQuiz->SelLength = 1;
    Graphics::TColor CurrColor = FormMain->RichEditQuiz->SelAttributes->Color;

    if (j == GetCurrentBlankNum())
      FormMain->RichEditQuiz->SelAttributes->Color = clRed;

    Graphics::TFontStyles CurrStyle = FormMain->RichEditQuiz->DefAttributes->Style;
FormMain->RichEditQuiz->SelAttributes->Style = FormMain->RichEditQuiz->SelAttributes-
>Style << fsUnderline;

    if (GetUserAnswer(j) == "")
      FormMain->RichEditQuiz->SelText = "_____";
    else
      FormMain->RichEditQuiz->SelText = GetUserAnswer(j).c_str();

    FormMain->RichEditQuiz->SelAttributes->Style = CurrStyle;
    FormMain->RichEditQuiz->SelAttributes->Color = CurrColor;
  }

  // restore previous user answer if any
  FormMain->EditFillInTheBlanks->Text = GetUserAnswer(GetCurrentBlankNum());

  // disable or enable navigation buttons
  if (IsFirstBlank() == true)
    FormMain->BitBtnFillInTheBlanksUp->Enabled = false;
  else
    FormMain->BitBtnFillInTheBlanksUp->Enabled = true;

  if (IsLastBlank() == true)
    FormMain->BitBtnFillInTheBlanksDown->Enabled = false;
  else
    FormMain->BitBtnFillInTheBlanksDown->Enabled = true;

  FormMain->EditFillInTheBlanks->SetFocus();
}
//---------------------------------------------------------------------------
```

Learning.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
USERES("Learning.res");
USEFORM("Main.cpp", FormMain);
USEUNIT("Tutorial.cpp");
USEUNIT("Demo.cpp");
USEUNIT("Shared.cpp");
USEUNIT("MultipleChoice.cpp");
USEUNIT("Quiz.cpp");
USEUNIT("QuizQuestion.cpp");
USEUNIT("ShortQuestion.cpp");
USEUNIT("FillInTheBlanks.cpp");
USEFORM("QuizMenu.cpp", FormQuizMenu);
USEFORM("About.cpp", FormAbout);
USEFORM("QuizResults.cpp", FormQuizResults);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
  try
  {
     Application->Initialize();
     Application->CreateForm(__classid(TFormMain), &FormMain);
     Application->CreateForm(__classid(TFormQuizMenu), &FormQuizMenu);
     Application->CreateForm(__classid(TFormAbout), &FormAbout);
     Application->CreateForm(__classid(TFormQuizResults), &FormQuizResults);
     Application->Run();
  }
  catch (Exception &exception)
  {
     Application->ShowException(&exception);
  }
  return 0;
}
//---------------------------------------------------------------------------
```

Main.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include <stdlib.h>
//---------------------------------------------------------------------------
#include "About.h"
#include "QuizResults.h"
#include "Main.h"
#include "QuizMenu.h"
#include "Tutorial.h"
#include "Demo.h"
#include "Quiz.h"
#include "MultipleChoice.h"
#include "ShortQuestion.h"
#include "FillInTheBlanks.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
//---------------------------------------------------------------------------
TFormMain *FormMain;
Tutorial *tutorial;
Demo *demo = NULL;
Quiz *quiz = NULL;
//---------------------------------------------------------------------------
__fastcall TFormMain::TFormMain(TComponent* Owner)
  : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnTutorialPreviousClick(TObject *Sender)
{
  tutorial->ReverseClip();
  tutorial->Show();
  TreeViewContents->Selected->GetPrev()->Selected = true;
  TreeViewContents->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnTutorialNextClick(TObject *Sender)
{
  tutorial->AdvanceClip();
  tutorial->Show();
  TreeViewContents->Selected->GetNext()->Selected = true;
  TreeViewContents->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::FormCreate(TObject *Sender)
{
  randomize();

  tutorial = new Tutorial("Tutorials", "Sounds\\Tutorials");

  // initializes content menu
  TreeViewContents->LoadFromFile(tutorial->GetDirectory() + "\\contents.dat");
  TreeViewContents->FullExpand();
  TreeViewContents->Selected = TreeViewContents->TopItem;

  tutorial->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::TreeViewContentsChange(TObject *Sender,
      TTreeNode *Node)
{
  tutorial->SetCurrentClipNum(Node->AbsoluteIndex);
  tutorial->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::FormActivate(TObject *Sender)
{
  mnuHelpAboutClick(Sender);
  TreeViewContents->SetFocus();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TFormMain::BitBtnTutorialDemoClick(TObject *Sender)
{
  demo = new Demo(AnsiString("Demos\\") + tutorial-
>GetDemoIndexDirectoryAt(TreeViewContents->Selected->AbsoluteIndex), "Sounds\\Demos");
  demo->Show();
  PanelDemo->BringToFront();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnDemoBackClick(TObject *Sender)
{
  delete demo;
  demo = NULL;
  tutorial->Show();
  PanelTutorial->BringToFront();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnDemoPreviousClick(TObject *Sender)
{
  demo->ReverseClip();
  demo->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnDemoNextClick(TObject *Sender)
{
  demo->AdvanceClip();
  demo->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::FormClose(TObject *Sender, TCloseAction &Action)
{
  delete tutorial;

  if (demo != NULL)
    delete demo;

  if (quiz != NULL)
    delete quiz;
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::FormCloseQuery(TObject *Sender, bool &CanClose)
{
  switch (MessageBox(Handle, "Are you sure you want to exit this application?",
"Learning C++ - Confirmation", MB_YESNO | MB_ICONQUESTION))
  {
    case ID_YES:  break;
    case ID_NO:   CanClose = false;
  }
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::mnuFileExitClick(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnQuizBackClick(TObject *Sender)
{
  FormQuizResults->ShowModal();
  delete quiz;
  quiz = NULL;
  tutorial->Show();
  PanelTutorial->BringToFront();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::RadioGroupMultipleChoiceClick(TObject *Sender)
{
  // stores user answer
  quiz->GetCurrentQuizQuestion()->SetUserAnswer(RadioGroupMultipleChoice->Items-
>Strings[RadioGroupMultipleChoice->ItemIndex]);
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnQuizSubmitClick(TObject *Sender)
{
  quiz->GetCurrentQuizQuestion()->Submit(); // submit the current quiz question

  if (quiz->GetCurrentQuizQuestion()->IsCorrect())
    quiz->SetNumCorrect(quiz->GetNumCorrect() + 1);
  else
```

```
    quiz->SetNumIncorrect(quiz->GetNumIncorrect() + 1);

  quiz->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnQuizNextClick(TObject *Sender)
{
  quiz->AdvanceClip();
  quiz->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnQuizPreviousClick(TObject *Sender)
{
  quiz->ReverseClip();
  quiz->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::MemoShortQuestionExit(TObject *Sender)
{
  // stores user answer
  quiz->GetCurrentQuizQuestion()->SetUserAnswer(MemoShortQuestion->Text);
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnFillInTheBlanksDownClick(TObject *Sender)
{
  // static binding
  ((FillInTheBlanks *)quiz->GetCurrentQuizQuestion())->AdvanceBlank();
  quiz->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnFillInTheBlanksUpClick(TObject *Sender)
{
  // static binding
  ((FillInTheBlanks *)quiz->GetCurrentQuizQuestion())->ReverseBlank();
  quiz->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnTutorialQuizClick(TObject *Sender)
{
  if (FormQuizMenu->ShowModal() == mrOk)  // take a quiz?
  {
     quiz = new Quiz(AnsiString("Quiz\\") + tutorial-
>GetQuizIndexDirectoryAt(TreeViewContents->Selected->AbsoluteIndex),
StrToInt(FormQuizMenu->EditQuizNumQuestions->Text));

    PanelQuiz->BringToFront();
    quiz->Show();
  }
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::mnuOptionsSoundClick(TObject *Sender)
{
  mnuOptionsSound->Checked = !mnuOptionsSound->Checked;

  if (!mnuOptionsSound->Checked)
    MediaPlayer->Close();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::mnuHelpAboutClick(TObject *Sender)
{
  FormAbout->ShowModal();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::BitBtnQuizAnswerClick(TObject *Sender)
{
  quiz->ShowAnswer();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::EditFillInTheBlanksKeyDown(TObject *Sender,
      WORD &Key, TShiftState Shift)
{
  if (Key == 38)  // up arrow
  {
    ((FillInTheBlanks *)quiz->GetCurrentQuizQuestion())->ReverseBlank();
    quiz->Show();
  }
```

```
  if (Key == 40)  // down arrow
  {
    ((FillInTheBlanks *)quiz->GetCurrentQuizQuestion())->AdvanceBlank();
    quiz->Show();
  }
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::EditFillInTheBlanksChange(TObject *Sender)
{
  ((FillInTheBlanks *)quiz->GetCurrentQuizQuestion())->SetUserAnswer(((FillInTheBlanks
*)quiz->GetCurrentQuizQuestion())->GetCurrentBlankNum(), EditFillInTheBlanks->Text);
  quiz->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::EditFillInTheBlanksEnter(TObject *Sender)
{
  // block text
  FormMain->EditFillInTheBlanks->SelStart = 0;
  FormMain->EditFillInTheBlanks->SelLength = FormMain->EditFillInTheBlanks-
>Text.Length();
}
//---------------------------------------------------------------------------
void __fastcall TFormMain::MemoShortQuestionEnter(TObject *Sender)
{
  // block text
  FormMain->MemoShortQuestion->SelStart = 0;
  FormMain->MemoShortQuestion->SelLength = FormMain->MemoShortQuestion->Text.Length();
}
//---------------------------------------------------------------------------
```

MultipleChoice.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include <fstream>
using namespace std;
//---------------------------------------------------------------------------
#include "MultipleChoice.h"
#include "Shared.h"
#include "Main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
//---------------------------------------------------------------------------
MultipleChoice::MultipleChoice(AnsiString indexFile)
{
  char line[MAX_NUMCHARS];

  fstream fsIndex(indexFile.c_str());

  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);

  // location of the question file
  QuestionFile = AnsiString(line);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);

  // location of the answer file
  AnswerFile = AnsiString(line);
  fsIndex.getline(line, MAX_NUMCHARS);

  do  // skip possible type(s) data
  {
    fsIndex.getline(line, MAX_NUMCHARS);
  } while (AnsiString(line) != "");

  fsIndex.getline(line, MAX_NUMCHARS);

  // read correct answer
  fsIndex.getline(line, MAX_NUMCHARS);
  CorrectAnswer = AnsiString(line);

  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);

  // read choices
  fsIndex.getline(line, MAX_NUMCHARS);

  while (AnsiString(line) != "")
  {
    viChoices.insert(viChoices.end(), AnsiString(line));
    fsIndex.getline(line, MAX_NUMCHARS);
  }

  fsIndex.close();
}
//---------------------------------------------------------------------------
void MultipleChoice::SetUserAnswer(AnsiString userAnswer)
{
  UserAnswer = userAnswer;
}
//---------------------------------------------------------------------------
AnsiString MultipleChoice::GetUserAnswer() const
{
  return UserAnswer;
}
//---------------------------------------------------------------------------
AnsiString MultipleChoice::GetCorrectAnswer() const
{
  return CorrectAnswer;
}
//---------------------------------------------------------------------------
```

```
void MultipleChoice::EnableInput() const
{
  FormMain->RadioGroupMultipleChoice->Enabled = true;
}
//---------------------------------------------------------------------------
void MultipleChoice::DisableInput() const
{
  FormMain->RadioGroupMultipleChoice->Enabled = false;
}
//---------------------------------------------------------------------------
bool MultipleChoice::IsCorrect() const
{
  return GetUserAnswer() == GetCorrectAnswer();
}
//---------------------------------------------------------------------------
void MultipleChoice::ShowAnswer() const
{
  FormMain->RichEditAnswer->Lines->LoadFromFile(AnswerFile);
}
//---------------------------------------------------------------------------
void MultipleChoice::Show() const
{
  FormMain->RadioGroupMultipleChoice->BringToFront();
  FormMain->RichEditQuiz->Lines->LoadFromFile(QuestionFile);

  // Show the blank if any
  if (FormMain->RichEditQuiz->FindText("_", 0, FormMain->RichEditQuiz->Text.Length(),
TSearchTypes() << stMatchCase) != -1)
  {
    FormMain->RichEditQuiz->SelStart = FormMain->RichEditQuiz->FindText("_", 0,
FormMain->RichEditQuiz->Text.Length(), TSearchTypes() << stMatchCase);
    FormMain->RichEditQuiz->SelLength = 1;
    FormMain->RichEditQuiz->SelText = "_____";
  }

  FormMain->RadioGroupMultipleChoice->Items->Clear();

  for (unsigned i = 0; i < viChoices.size(); i++)
  {
    FormMain->RadioGroupMultipleChoice->Items->Add(viChoices[i].c_str());

    // restore previous user answer if any
    if (viChoices[i] == UserAnswer)
      FormMain->RadioGroupMultipleChoice->ItemIndex = i;
  }
}
//---------------------------------------------------------------------------
```

Quiz.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include <fstream>
#include <string>
#include <vector>
using namespace std;
//---------------------------------------------------------------------------
#include "Shared.h"
#include "Quiz.h"
#include "Main.h"
#include "MultipleChoice.h"
#include "FillInTheBlanks.h"
#include "ShortQuestion.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
//---------------------------------------------------------------------------
AnsiString Quiz::GetRandomQuestionType(AnsiString dataFile) const
{
  char line[MAX_NUMCHARS];
  vector <AnsiString> viQuestionTypes;

  fstream fsIndex(dataFile.c_str());

  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);

  fsIndex.getline(line, MAX_NUMCHARS);

  while (AnsiString(line) != "")
  {
    viQuestionTypes.insert(viQuestionTypes.end(), AnsiString(line));
    fsIndex.getline(line, MAX_NUMCHARS);
  }

  fsIndex.close();

  return viQuestionTypes[random(viQuestionTypes.size())];
}
//---------------------------------------------------------------------------
Quiz::Quiz(AnsiString Directory, int numQuestions)
{
  // locations of quiz question data files
  vector <AnsiString> viLocations;
  char line[MAX_NUMCHARS];
  int NumClips = 0;

  CurrentClipNum = 0;
  NumCorrect = 0;
  NumIncorrect = 0;

  fstream fsIndex((Directory + "\\index.dat").c_str());

  // how big is the index file?
  while (!fsIndex.eof())
  {
    fsIndex.getline(line, MAX_NUMCHARS);
    viLocations.insert(viLocations.end(), AnsiString(line));
    NumClips++;
  }

  fsIndex.close();

  // select the questions randomly including their types
  for (int i = 0; i < numQuestions; i++)
  {
    int selectedQuestionNum = random(NumClips);
```

```
    AnsiString selectedQuestionType = GetRandomQuestionType(Directory + "\\" +
viLocations[selectedQuestionNum]);

    // the destructor will take care of the memory deallocation
    if (selectedQuestionType == "MultipleChoice")
      AddQuestion(new MultipleChoice(Directory + "\\" +
viLocations[selectedQuestionNum]));
    else if (selectedQuestionType == "FillInTheBlanks")
      AddQuestion(new FillInTheBlanks(Directory + "\\" +
viLocations[selectedQuestionNum]));
    else if (selectedQuestionType == "ShortQuestion")
      AddQuestion(new ShortQuestion(Directory + "\\" +
viLocations[selectedQuestionNum]));
  }
}
//---------------------------------------------------------------------------
Quiz::~Quiz()
{
  for (unsigned i = 0; i < viQuizQuestions.size(); i++)
    delete viQuizQuestions[i];
}
//---------------------------------------------------------------------------
void Quiz::AddQuestion(QuizQuestion *quizQuestion)
{
  viQuizQuestions.insert(viQuizQuestions.end(), quizQuestion);
}
//---------------------------------------------------------------------------
bool Quiz::SetCurrentClipNum(int currentClipNum)
{
  // invalid clip number
  if (currentClipNum < 0 || (unsigned)currentClipNum >= viQuizQuestions.size())
    return false;

  CurrentClipNum = currentClipNum;

  return true;
}
//---------------------------------------------------------------------------
int Quiz::GetCurrentClipNum() const
{
  return CurrentClipNum;
}
//---------------------------------------------------------------------------
bool Quiz::IsFirstClip() const
{
  if (CurrentClipNum == 0)
    return true;

  return false;
}
//---------------------------------------------------------------------------
bool Quiz::IsLastClip() const
{
  if ((unsigned)CurrentClipNum == viQuizQuestions.size() - 1)
    return true;

  return false;
}
//---------------------------------------------------------------------------
bool Quiz::AdvanceClip()
{
  if (IsLastClip() == true)
    return false;

  SetCurrentClipNum(GetCurrentClipNum() + 1);

  return true;
}
//---------------------------------------------------------------------------
bool Quiz::ReverseClip()
{
  if (IsFirstClip() == true)
    return false;

  SetCurrentClipNum(GetCurrentClipNum() - 1);

  return true;
```

```
}
//---------------------------------------------------------------------------
int Quiz::GetNumClips() const
{
  return viQuizQuestions.size();
}
//---------------------------------------------------------------------------
int Quiz::GetNumCorrect() const
{
  return NumCorrect;
}
//---------------------------------------------------------------------------
int Quiz::GetNumIncorrect() const
{
  return NumIncorrect;
}
//---------------------------------------------------------------------------
void Quiz::SetNumCorrect(int numCorrect)
{
  NumCorrect = numCorrect;
}
//---------------------------------------------------------------------------
void Quiz::SetNumIncorrect(int numIncorrect)
{
  NumIncorrect = numIncorrect;
}
//---------------------------------------------------------------------------
QuizQuestion *Quiz::GetCurrentQuizQuestion() const
{
  return viQuizQuestions[GetCurrentClipNum()];
}
//---------------------------------------------------------------------------
void Quiz::ShowAnswer() const
{
  FormMain->BitBtnQuizAnswer->Enabled = false;
  FormMain->BitBtnQuizSubmit->Enabled = false;
  FormMain->GroupBoxAnswer->BringToFront();

  // call the corresponding ShowAnswer function according to the type of question
  viQuizQuestions[CurrentClipNum]->ShowAnswer();
}
//---------------------------------------------------------------------------
void Quiz::Show() const
{
  // call the corresponding Show function according to the type of question
  viQuizQuestions[CurrentClipNum]->Show();

  FormMain->StatusBarMain->SimpleText = AnsiString("Clip ") + (GetCurrentClipNum() + 1)
+ " of " + viQuizQuestions.size();

  // disable or enable navigation buttons

  if (IsFirstClip() == true)
    FormMain->BitBtnQuizPrevious->Enabled = false;
  else
    FormMain->BitBtnQuizPrevious->Enabled = true;

  if (IsLastClip() == true || GetCurrentQuizQuestion()->IsSubmitted() == false)
    FormMain->BitBtnQuizNext->Enabled = false;
  else
    FormMain->BitBtnQuizNext->Enabled = true;

  FormMain->BitBtnQuizAnswer->Enabled = true;
  FormMain->BitBtnQuizSubmit->Enabled = true;

  // display submit or answer button depend on whether the question is answered
  // also disable input if the question has already answered
  if (GetCurrentQuizQuestion()->IsSubmitted())
  {
    FormMain->BitBtnQuizAnswer->BringToFront();
    viQuizQuestions[CurrentClipNum]->DisableInput();
    FormMain->BitBtnQuizAnswer->SetFocus();

    // display result image
    if (viQuizQuestions[CurrentClipNum]->IsCorrect())
      FormMain->ImageQuizResult->Picture->LoadFromFile("Images\\Correct.bmp");
    else
```

```
      FormMain->ImageQuizResult->Picture->LoadFromFile("Images\\Incorrect.bmp");
  }
  else
  {
    FormMain->BitBtnQuizSubmit->BringToFront();
    viQuizQuestions[CurrentClipNum]->EnableInput();
    FormMain->ImageQuizResult->Picture = NULL;
  }
}
//---------------------------------------------------------------------------
```

QuizMenu.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include "QuizMenu.h"
#include "Main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormQuizMenu *FormQuizMenu;
//---------------------------------------------------------------------------
__fastcall TFormQuizMenu::TFormQuizMenu(TComponent* Owner)
  : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFormQuizMenu::ButtonCancelClick(TObject *Sender)
{
  ModalResult = mrCancel;
}
//---------------------------------------------------------------------------
void __fastcall TFormQuizMenu::ButtonGoClick(TObject *Sender)
{
  ModalResult = mrOk;
}
//---------------------------------------------------------------------------
void __fastcall TFormQuizMenu::EditQuizNumQuestionsExit(TObject *Sender)
{
  if (StrToInt(EditQuizNumQuestions->Text) < 1 || StrToInt(EditQuizNumQuestions->Text)
> 60)
  {
    Application->MessageBox("You can only choose between 1 to 60 inclusive", "Learning
C++ - Invalid number of questions", MB_OK);
    EditQuizNumQuestions->SetFocus();
  }
}
//---------------------------------------------------------------------------
void __fastcall TFormQuizMenu::FormActivate(TObject *Sender)
{
  FormMain->MediaPlayer->Close();
  EditQuizNumQuestions->SetFocus();
}
//---------------------------------------------------------------------------
```

QuizQuestion.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include "QuizQuestion.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
//---------------------------------------------------------------------------
QuizQuestion::QuizQuestion()
{
  Submitted = false;
}
//---------------------------------------------------------------------------
void QuizQuestion::Submit()
{
  Submitted = true;
}
//---------------------------------------------------------------------------
bool QuizQuestion::IsSubmitted() const
{
  return Submitted;
}
//---------------------------------------------------------------------------
```

QuizResults.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include "QuizResults.h"
#include "Quiz.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TFormQuizResults *FormQuizResults;
extern Quiz *quiz;
//---------------------------------------------------------------------------
__fastcall TFormQuizResults::TFormQuizResults(TComponent* AOwner)
        : TForm(AOwner)
{
}
//---------------------------------------------------------------------------
void __fastcall TFormQuizResults::FormActivate(TObject *Sender)
{
  EditTotalNumQuestions->Text = quiz->GetNumClips();
  EditNumCorrectAnswers->Text = quiz->GetNumCorrect();
  EditNumIncorrectAnswers->Text = quiz->GetNumIncorrect();
  EditNumOmittedQuestions->Text = quiz->GetNumClips() - quiz->GetNumCorrect() - quiz-
>GetNumIncorrect();
  LabelScore->Caption = (int)(quiz->GetNumCorrect() / quiz->GetNumClips() * 100);

  // no negative scores
  if (LabelScore->Caption < 0)
    LabelScore->Caption = 0;

  // show grades according to the scores
  if (StrToInt(LabelScore->Caption) >= 97)
    LabelGrade->Caption = "A+ :O";
  else if (StrToInt(LabelScore->Caption) >= 93)
    LabelGrade->Caption = "A :o";
  else if (StrToInt(LabelScore->Caption) >= 90)
    LabelGrade->Caption = "A- :)";
  else if (StrToInt(LabelScore->Caption) >= 87)
    LabelGrade->Caption = "B+ :P";
  else if (StrToInt(LabelScore->Caption) >= 83)
    LabelGrade->Caption = "B :p";
  else if (StrToInt(LabelScore->Caption) >= 80)
    LabelGrade->Caption = "B- :|";
  else if (StrToInt(LabelScore->Caption) >= 77)
    LabelGrade->Caption = "C+ :|";
  else if (StrToInt(LabelScore->Caption) >= 73)
    LabelGrade->Caption = "C :(";
  else if (StrToInt(LabelScore->Caption) >= 70)
    LabelGrade->Caption = "C- :(";
  else
    LabelGrade->Caption = ":(";
}
//---------------------------------------------------------------------------
```

Shared.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include "Shared.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
//---------------------------------------------------------------------------
```

ShortQuestion.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include <fstream>
using namespace std;
//---------------------------------------------------------------------------
#include "ShortQuestion.h"
#include "Shared.h"
#include "Main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
//---------------------------------------------------------------------------
ShortQuestion::ShortQuestion(AnsiString indexFile)
{
  char line[MAX_NUMCHARS];

  fstream fsIndex(indexFile.c_str());

  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);

  // location of the question file
  QuestionFile = AnsiString(line);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);
  fsIndex.getline(line, MAX_NUMCHARS);

  // location of the answer file
  AnswerFile = AnsiString(line);
  fsIndex.getline(line, MAX_NUMCHARS);

  do  // skip possible type(s) data
  {
    fsIndex.getline(line, MAX_NUMCHARS);
  } while (AnsiString(line) != "");

  fsIndex.getline(line, MAX_NUMCHARS);

  // read correct answer
  fsIndex.getline(line, MAX_NUMCHARS);

  while (AnsiString(line) != "")
  {
    CorrectAnswer += AnsiString(line) + (char)13 + (char)10;
    fsIndex.getline(line, MAX_NUMCHARS);
  }

  fsIndex.close();
}
//---------------------------------------------------------------------------
void ShortQuestion::SetUserAnswer(AnsiString userAnswer)
{
  UserAnswer = userAnswer;
}
//---------------------------------------------------------------------------
AnsiString ShortQuestion::GetUserAnswer() const
{
  return UserAnswer;
}
//---------------------------------------------------------------------------
AnsiString ShortQuestion::GetCorrectAnswer() const
{
  return CorrectAnswer;
}
//---------------------------------------------------------------------------
void ShortQuestion::EnableInput() const
{
  FormMain->MemoShortQuestion->ReadOnly = false;
}
//---------------------------------------------------------------------------
void ShortQuestion::DisableInput() const
{
```

```
  FormMain->MemoShortQuestion->ReadOnly = true;
}
//---------------------------------------------------------------------------
bool ShortQuestion::IsCorrect() const
{
  return GetUserAnswer().Trim() == GetCorrectAnswer().Trim();
}
//---------------------------------------------------------------------------
void ShortQuestion::ShowAnswer() const
{
  FormMain->RichEditAnswer->Lines->LoadFromFile(AnswerFile);
}
//---------------------------------------------------------------------------
void ShortQuestion::Show() const
{
  FormMain->GroupBoxShortQuestion->BringToFront();
  FormMain->RichEditQuiz->Lines->LoadFromFile(QuestionFile);

  // Show the blank if any
  if (FormMain->RichEditQuiz->FindText("_", 0, FormMain->RichEditQuiz->Text.Length(),
TSearchTypes() << stMatchCase) != -1)
  {
    FormMain->RichEditQuiz->SelStart = FormMain->RichEditQuiz->FindText("_", 0,
FormMain->RichEditQuiz->Text.Length(), TSearchTypes() << stMatchCase);
    FormMain->RichEditQuiz->SelLength = 1;
    FormMain->RichEditQuiz->SelText = "_____";
  }

  FormMain->MemoShortQuestion->Clear();

  // restore previous user answer if any
  FormMain->MemoShortQuestion->Text = UserAnswer;

  FormMain->MemoShortQuestion->SetFocus();
}
//---------------------------------------------------------------------------
```

Tutorial.cpp

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
#include <fstream>
#include <string>
using namespace std;
//---------------------------------------------------------------------------
#include "Tutorial.h"
#include "Shared.h"
#include "Main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
//---------------------------------------------------------------------------
Tutorial::Tutorial(AnsiString directory, AnsiString soundDirectory)
{
  char line[MAX_NUMCHARS];
  NumClips = 0;

  Directory = directory;
  SoundDirectory = soundDirectory;
  CurrentClipNum = 0;

  fstream fsIndex((Directory + "\\index.dat").c_str());

  while (!fsIndex.eof())
  {
    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);

    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    viFileLocations.insert(viFileLocations.end(), AnsiString(line));

    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    viSoundLocations.insert(viSoundLocations.end(), AnsiString(line));

    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    viQuizIndexes.insert(viQuizIndexes.end(), AnsiString(line));

    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    fsIndex.getline(line, MAX_NUMCHARS);
    viDemoIndexes.insert(viDemoIndexes.end(), AnsiString(line));

    fsIndex.getline(line, MAX_NUMCHARS);

    NumClips++;
  }

  fsIndex.close();
}
//---------------------------------------------------------------------------
AnsiString Tutorial::GetDirectory() const
{
  return Directory;
}
//---------------------------------------------------------------------------
AnsiString Tutorial::GetDemoIndexDirectoryAt(int i) const
{
  if (i < 0 || i >= GetNumClips())
    return AnsiString();

  return viDemoIndexes[i];
}
//---------------------------------------------------------------------------
AnsiString Tutorial::GetQuizIndexDirectoryAt(int i) const
{
  if (i < 0 || i >= GetNumClips())
```

```
    return AnsiString();

  return viQuizIndexes[i];
}
//---------------------------------------------------------------------------
int Tutorial::GetNumClips() const
{
  return NumClips;
}
//---------------------------------------------------------------------------
bool Tutorial::SetCurrentClipNum(int currentClipNum)
{
  // invalid clip number
  if (currentClipNum < 0 || currentClipNum >= NumClips)
    return false;

  CurrentClipNum = currentClipNum;

  return true;
}
//---------------------------------------------------------------------------
int Tutorial::GetCurrentClipNum() const
{
  return CurrentClipNum;
}
//---------------------------------------------------------------------------
bool Tutorial::AdvanceClip()
{
  if (IsLastClip() == true)
    return false;

  SetCurrentClipNum(GetCurrentClipNum() + 1);

  return true;
}
//---------------------------------------------------------------------------
bool Tutorial::ReverseClip()
{
  if (IsFirstClip() == true)
    return false;

  SetCurrentClipNum(GetCurrentClipNum() - 1);

  return true;
}
//---------------------------------------------------------------------------
bool Tutorial::IsFirstClip() const
{
  if (CurrentClipNum == 0)
    return true;

  return false;
}
//---------------------------------------------------------------------------
bool Tutorial::IsLastClip() const
{
  if (CurrentClipNum == NumClips - 1)
    return true;

  return false;
}
//---------------------------------------------------------------------------
void Tutorial::Show() const
{
  FormMain->RichEditTutorial->Lines->LoadFromFile(Directory + "\\" +
viFileLocations[CurrentClipNum]);
  FormMain->StatusBarMain->SimpleText = AnsiString("Clip ") + (GetCurrentClipNum() + 1)
+ " of " + GetNumClips();

  // disable or enable navigation buttons
  if (IsFirstClip() == true)
    FormMain->BitBtnTutorialPrevious->Enabled = false;
  else
    FormMain->BitBtnTutorialPrevious->Enabled = true;

  if (IsLastClip() == true)
    FormMain->BitBtnTutorialNext->Enabled = false;
```

```
  else
    FormMain->BitBtnTutorialNext->Enabled = true;

  // disable or enable quiz button
  if (viQuizIndexes[CurrentClipNum] != "")
    FormMain->BitBtnTutorialQuiz->Enabled = true;
  else
    FormMain->BitBtnTutorialQuiz->Enabled = false;

  // disable or enable demo button
  if (viDemoIndexes[CurrentClipNum] != "")
    FormMain->BitBtnTutorialDemo->Enabled = true;
  else
    FormMain->BitBtnTutorialDemo->Enabled = false;

  // play sound if it is enabled
  if (FormMain->mnuOptionsSound->Checked && viSoundLocations[CurrentClipNum] != "")
  {
    FormMain->MediaPlayer->FileName = SoundDirectory + "\\" +
viSoundLocations[CurrentClipNum];
    FormMain->MediaPlayer->Open();
    FormMain->MediaPlayer->Play();
  }
}
//---------------------------------------------------------------------------
```

APPENDIX C – SOURCE CODE (Survey Form)

form.cgi (in Perl) [Section 3.3]

```
#!/usr/local/bin/perl

#
# form.cgi - a Perl script that manipulates the online survey
#

use CGI qw/:standard/;

$Q1 = param(Q1);
$Q1_extra = param(Q1_specify);

$Q2 = param(Q2);
$Q3 = param(Q3);

$Q4_book = param(Q4_book);
$Q4_course = param(Q4_course);
$Q4_computer = param(Q4_computer);
$Q4_other = param(Q4_other);
$Q4_specify = param(Q4_specify);

$Q5_system_easy = param(Q5_system_easy);
$Q5_system_performance = param(Q5_system_performance);
$Q5_system_other = param(Q5_system_other);
$Q5_system_specify = param(Q5_system_specify);

$Q5_presentation_text = param(Q5_presentation_text);
$Q5_presentation_color = param(Q5_presentation_color);
$Q5_presentation_graphics = param(Q5_presentation_graphics);
$Q5_presentation_sound = param(Q5_presentation_sound);
$Q5_presentation_animation = param(Q5_presentation_animation);
$Q5_presentation_other = param(Q5_presentation_other);
$Q5_presentation_specify = param(Q5_presentation_specify);

$Q5_tutorial_quantity = param(Q5_tutorial_quantity);
$Q5_tutorial_easy = param(Q5_tutorial_easy);
$Q5_tutorial_meaningfully = param(Q5_tutorial_meaningfully);
$Q5_tutorial_accuracy = param(Q5_tutorial_accuracy);
$Q5_tutorial_other = param(Q5_tutorial_other);
$Q5_tutorial_specify = param(Q5_tutorial_specify);

$Q5_drill_quantity = param(Q5_drill_quantity);
$Q5_drill_revise = param(Q5_drill_revise);
$Q5_drill_encourage = param(Q5_drill_encourage);
$Q5_drill_feedback = param(Q5_drill_feedback);
$Q5_drill_other = param(Q5_drill_other);
$Q5_drill_specify = param(Q5_drill_specify);

$Q6 = param(Q6);
$Q7 = param(Q7);
$Q8 = param(Q8);
$Q9 = param(Q9);
$Q10 = param(Q10);

$Name = param(Name);
$Email = param(Email);

open(DATA, ">>survey.txt");

print DATA "1. Have you ever used a CAI programs?\n";
print DATA "$Q1\n";
print DATA "(Please specify: $Q1_extra)\n";

print DATA "2. On a scale of 1 to 10, where 1 means nothing and 10 means I fully
mastered the subject, how much do you think you learned from the CAI program?\n";
print DATA "$Q2\n";

print DATA "3. What features did you like/dislike most about those CAI programs?\n";
print DATA "$Q3\n";

print DATA "4. Rank the following learning materials in the order of your ";
print DATA "preference:\n(1: most preferred, 2: next preferred, etc)\n";

print DATA "Book: $Q4_book\n";
```

```
print DATA "Course: $Q4_course\n";
print DATA "Computer: $Q4_computer\n";
print DATA "Other: $Q4_other ";
print DATA "(please specify: $Q4_specify)\n";

print DATA "5. Rank each of the following according to its order of importance for the
overall efficacy of a CAI program (please make separate rankings for each category
with 1 being the most important and 2 being the second most important, etc. Put an 'X'
if the item is not applicable):\n";
print DATA "* System:\n";
print DATA "Easy to use: $Q5_system_easy\n";
print DATA "Performance: $Q5_system_performance\n";
print DATA "Other: $Q5_system_other ";
print DATA "(Please specify: $Q5_system_specify)\n";

print DATA "* Presentation:\n";
print DATA "Text Layout: $Q5_presentation_text\n";
print DATA "Color: $Q5_presentation_color\n";
print DATA "Graphics: $Q5_presentation_graphics\n";
print DATA "Sound: $Q5_presentation_sound\n";
print DATA "Animation: $Q5_presentation_animation\n";
print DATA "Other: $Q5_presentation_other ";
print DATA "(please specify: $Q5_presentation_specify)\n";

print DATA "* Tutorial Contents:\n";
print DATA "Quantity: $Q5_tutorial_quantity\n";
print DATA "Easy to understand: $Q5_tutorial_easy\n";
print DATA "Meaningfully structured: $Q5_tutorial_meaningfully\n";
print DATA "Accuracy: $Q5_tutorial_accuracy\n";
print DATA "Other: $Q5_tutorial_other ";
print DATA "(please specify: $Q5_tutorial_specify)\n";

print DATA "* Drill and Practice Exercises\n";
print DATA "Quantity: $Q5_drill_quantity\n";
print DATA "Review learned materials: $Q5_drill_revise\n";
print DATA "Encourage thinking: $Q5_drill_encourage\n";
print DATA "Feedback: $Q5_drill_feedback\n";
print DATA "Other: $Q5_drill_other ";
print DATA "(please specify: $Q5_drill_specify)\n";

print DATA "6. What types of interaction could a CAI program provide so that it
consistently maintains the user's interest?\n";
print DATA "$Q6\n";

print DATA "7. How do you think the tutorial materials could be organized so that they
are more interesting than the material in a book?\n";
print DATA "$Q7\n";

print DATA "8. Some experts think that the drill and practice questions in CAI
software are ineffective. Do you agree? Why? How would you suggest the questions
be designed so that they become more effective?\n";
print DATA "$Q8\n";

print DATA "9. How much think is the most reasonable price for a CAI program?\n";
print DATA "$Q9\n";

print DATA "10. What would you like to see in the future CAI programs?\n";
print DATA "$Q10\n";

print DATA "Name:\n$Name\n";
print DATA "Email:\n$Email\n";

print DATA "-----------------------------------------------------------------------\n";

close DATA;

print "Content-type: text/html\n\n";
print "<html>\n";
print "<head>\n";
print "<title>Thank you</title>\n";
print "</head>\n";
print "<body>\n";
print "<h1>Thank you</h1>\n";
print "<P>The survey has been submitted successfully. ";
print "The results of this survey will be considered seriously in creating a ";
print "more effective CAI program for the future. I truly appreciate your ";
print "effort on this survey. Thank you!";
```

```
print "<p>Sincerely, <br>";
print "Sze-Wai Yam<br>";
print "<a href=\"mailto:cwai\@wpi.edu\">cwai\@wpi.edu</a></td>";
print "</body>\n";
print "</html>\n";
```

BIBLIOGRAPHY

Alessi, S.M.; and Trollip, S.R. "Computer-based Instruction: Methods and Development." Prentice Hall. Englewood Cliffs, N.J. 1991.

Army Training Information Management Program; the Army Training Support Center; and the United States Army. “Advanced Distributed Learning Initiative — Shareable Courseware Object Reference Model (DRAFT version 0.7.3).” 4 May 1999. http://www.atimp.army.mil/collab/sco-rm073.asp#_Toc450388105

Arnold, Douglas N. “Computer-Aided Instruction.” 29 Mar. 2000. Microsoft Encarta Online Encyclopedia 2000. http://encarta.msn.com

Bangert-Drowns, R. L.; Kulik, J. A.; and Kulik, C.C. "Effectiveness of Computer-Based Education in Secondary Schools." Journal of Computer-Based Instruction. 3 Dec. 1985: 59-68.

Beres, Marissa; Goodfellow, Lauren M; and Kassabian, Debra Shnorig. "Computer Aided Learning for Unit Operations." Interactive Qualifying Project, Worcester Polytechnic Institute 1999: 99D095M.

Bijou, Sidney W. “Behaviorism.” 20 April. 2000. Microsoft Encarta Online Encyclopedia 2000. http://encarta.msn.com

Cergneux, Maximo Jose; Ciszewski, Kevin Michael; and Dziczek, Erica Stephenia. "Computer-Aided Learning of Binary Distillation Concepts." Interactive Qualifying Project, Worcester Polytechnic Institute 1998: 98D086M.

Cotton, Kathleen. "Computer-Assisted Instruction." 7 Sep 1997. www.nwrel.org/scpd/sirs/5/cu10.html (7 Oct 1999)

ETCAI Teaching Electronics Technology. "Resources for Electronics Training." 3 Sep. 1999. http://www.datasync.com/~etcai/training.htm (7 Oct. 1999)

Frith, Vera; and Heckroodt, Oelof. "Computer-assisted-learning: guidance for non-experts." Sep. 1997. http://www.civeng.uct.ac.za/caleng/caldisad.htm (7 Oct. 1999)

Glover, Dick. “How do we learn?” http://context.tlsu.leeds.ac.uk/howdowe.asp (27 Mar. 2000)

Kazmierczak, Stephen Joseph; Morin, Brian Ross and Vella, Paul Raymond. "Computers and Education." Interactive Qualifying Project, Worcester Polytechnic Institute 1997: 97D264I.

Kinnaman, D. E. "What's the Research Telling Us?" Classroom Computer Learning 6 Oct. 1990:31-35; 38-39

Lewis, R.; and Tagg E. D. "Treads in Computer Assisted Education." Blackwell Scientific Publications. 1987: 4.

Munden, C. Dianne. “What is Computer Assisted Instruction?” 21 Aug. 1996. http://www.auburn.edu/~mundecd/cai.html (4)

On Purpose Associates. “About Learning/Theories.” Funderstanding. 1998. http://www.funderstanding.com/theories1.html

Robertson, E. B.; Ladewig, B. H.; Strickland, M. P.; and Boschung, M. D. "Enchancement of Self-Esteem Through the Use of Computer-Assisted Instruction." Journal of Educational Research. 5 Aug. 1987: 314-316

Shneiderman, Ben. “Designing the User Interface — Strategies for Effective Human-Computer Interaction (Third Edition)” Addison-Wesley Longman, Inc. 1998: 74-76.

Vockell, Edward L.; and Schwartz, Eileen M. "The Computer in the Classroom, Second Edition" McGRAW-HILL, Inc. 1992: 63.

WAVE Technologies International, Inc. “Learning: The Critical Technology; A whitepaper on adult education in the information age.” (9 April 2000)

White House CAI and CBT results. http://www.whitehouse.gov/WH/New/edtech/perform.html (7 Oct. 1999)