

## **INFORMATION TO USERS**

**This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.**

**The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.**

**In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.**

**Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.**

**Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.**

**ProQuest Information and Learning  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
800-521-0600**

**UMI<sup>®</sup>**



**AN INTERACTIVE, MULTIMEDIA INSTRUCTIONAL  
TOOL FOR COMPUTER SCIENCE STUDENTS**

**A THESIS  
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
MASTER OF SCIENCE  
IN  
COMPUTER SCIENCE  
UNIVERSITY OF REGINA**

**By  
Chi Song  
Regina, Saskatchewan  
July 25, 2001**

**© Copyright 2001: Chi Song**



**National Library  
of Canada**

**Acquisitions and  
Bibliographic Services**

**395 Wellington Street  
Ottawa ON K1A 0N4  
Canada**

**Bibliothèque nationale  
du Canada**

**Acquisitions et  
services bibliographiques**

**395, rue Wellington  
Ottawa ON K1A 0N4  
Canada**

*Your file Votre référence*

*Our file Notre référence*

**0-612-66251-9**

**The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.**

**The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.**

**L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.**

**L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.**

**Canada**

UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH


CERTIFICATION OF THESIS WORK

We, the undersigned, certify that Chi Song, candidate for the Degree of Master of Science, has presented a thesis on *An Interactive, Multimedia Instructional Tool for Computer Science Students*, that the thesis is acceptable in form and content, and that the student demonstrated a satisfactory knowledge of the field covered by the thesis in an oral examination held on July 19, 2001.

External Examiner:

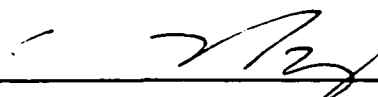
  
\_\_\_\_\_  
Dr. D. Chandler, Department of Chemistry


Internal Examiners:

  
\_\_\_\_\_  
Dr. L. Symes, Supervisor

\_\_\_\_\_

\_\_\_\_\_

  
\_\_\_\_\_

  
\_\_\_\_\_

**UNIVERSITY OF REGINA**  
**FACULTY OF GRADUATE STUDIES AND RESEARCH**  
**PERMISSION TO USE POSTGRADUATE THESIS**

**TITLE OF THESIS:** An Interactive, Multimedia Instructional Tool for Computer Science Students

**NAME OF AUTHOR:** Chi Song

**DEGREE:** Master of Science

In presenting this thesis in partial fulfillment of the requirements for a postgraduate degree from the University of Regina, I agree that the Libraries of this University shall make it freely available for inspection. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the professor or professors who supervised my thesis work, or in their absence, by the Head of the Department or the Dean of the Faculty in which my thesis work was done. It is understood that with the exception of UMI Dissertations Publishing (UMI) that any copying, publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Regina in any scholarly use which may be made of my material in my thesis.

**SIGNATURE:** \_\_\_\_\_

*Chi Song*

**DATE:** \_\_\_\_\_

*July 19th, 2001*

# Abstract

Interactive multimedia identifies a relatively new technology that integrates a variety of media (including video, sound, music, photographs, drawings, computer graphics, and animation) into a single electronic medium, and it provides the ability to link ideas in a nonlinear format. Increasingly, with the rapid advances in digital and computer technology, it is digital-based storage platforms (multimedia hardware systems) that are gaining attention for multimedia delivery. Compact disc read-only memory (CD-ROM) and other digital-based platforms are evolving rapidly; these new technologies are enhancing the capabilities of multimedia.

The primary goal of this study is to provide CD-ROM and World Wide Web (WWW) based multimedia instructional tools for computer science students. They can be used by introductory level students to learn the basic fundamental concepts of programming with Java.

This research will also assess the multimedia tool kit's value in aiding the student's ability to learn fundamental Computer Science concepts, and an assessment will be made of its merits on various facts, such as effective understanding of concepts, time of learning, etc.

# Acknowledgments

I would like to express thanks to my supervisor Dr. Larry Symes, who stimulated my interest in this topic, gave unsparingly of his time, critical advice and encouragement, and patiently guided me from the research stage to the writing of this thesis.

I gratefully acknowledge the generous academic and financial support of the Faculty of Graduate Studies and Research and the Computer Science Department.

I would also like to thank my external examiner, Dr. David Chandler, Department of Chemistry, for his valuable comments and corrections.

I would like to thank my internal committee members, Dr. Yi-yu Yao and Dr. Chang-Nian Zhang for their attention to my research work and many valuable comments. Thanks also to Dr. Chun-Hua Guo for chairing my thesis defense.

I would like to express thanks to Dr. Maguire, who gave me financial support to finish this research work. The work was also supported by Student Connection Program.

I am also grateful to other faculty members of the Computer Science Department and Senthil Kumar Anandan, with whom I discussed various problems from time to time.

I would like to express thanks to Connie Maguire, for recording all the audio files, and her excellent job on proofreading work.



Finally, I offer thanks to my wife Yan Zhao and family members for their constant love, encouragement and moral support. To them, I dedicate this study with my warmest love and profound gratitude.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgments</b>	<b>ii</b>
<b>Table of Contents</b>	<b>iv</b>
<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Notations</b>	<b>1</b>
<b>Chapter 1 Introduction</b>	<b>2</b>
1.1 Multimedia and Objectives of Learning . . . . .	3
1.2 Multimedia and Computer-Assisted Instruction (CAI) . . . . .	4
1.3 Multimedia and WWW (World Wide Web) . . . . .	5
1.4 CD-ROM . . . . .	7
1.5 Multimedia Development Tools . . . . .	7
1.6 Research Motivation . . . . .	8
1.7 Organization of the Thesis . . . . .	9
<b>Chapter 2 Literature Review</b>	<b>11</b>
2.1 Two typical earlier CAI systems . . . . .	11
2.1.1 PLATO . . . . .	11
2.1.2 TICCIT . . . . .	13

2.2	Intelligent Computer-Assisted Instruction (ICAI) . . . . .	14
2.2.1	STEAMER . . . . .	14
2.2.2	PROUST . . . . .	15
2.2.3	LISP-Tutor . . . . .	17
2.2.4	GUIDON . . . . .	18
2.3	World Wide Web (WWW)-based Tutoring Systems . . . . .	19
2.3.1	What is WWW? . . . . .	19
2.3.2	Why Use the WWW for Distance Learning? . . . . .	21
2.3.3	WebCT (Web Course Tool) . . . . .	22
2.3.4	Centra99 (Centra Symposium) . . . . .	24
2.3.5	AIMIT . . . . .	25
2.4	Informal Comparison . . . . .	26
 <b>Chapter 3 Instructional System Design-The Design and Development Phase</b>		<b>28</b>
3.1	The Delivery Platform and Media of AIMIT . . . . .	29
3.2	Authoring tool–Macromedia’s Director 7.0 . . . . .	29
3.3	Authoring Workstations and Complementary Hardware . . . . .	33
3.4	The strategy: Project Analysis and Design . . . . .	33
3.4.1	Define the goals . . . . .	33
3.4.2	Resources inventory . . . . .	34
3.4.3	Information (knowledge) Design and AIMIT Diagram . . . . .	34
3.4.4	Graphic User Interface (GUI) Design . . . . .	35
3.4.5	Navigation and Interactivity . . . . .	36
3.4.6	Creating Storyboard . . . . .	38
 <b>Chapter 4 Instructional System Design-The Implementation and Delivery Phase</b>		<b>40</b>
4.1	Director’s Developing Environment for AIMIT . . . . .	40
4.1.1	The Basic Components of Macromedia Director Movie . . . . .	40
4.1.2	Interactive Action Lingo . . . . .	43
4.2	Object-oriented programming in Director . . . . .	44

4.3	Window Management (User Interface) in AIMIT . . . . .	47
4.3.1	Home Index Window . . . . .	48
4.3.2	Tutoring Window . . . . .	51
4.4	Handling Sound in AIMIT . . . . .	54
4.4.1	Recording Sound File by using Cool Edit Pro . . . . .	55
4.4.2	Controlling sound in the Score Window . . . . .	56
4.4.3	Synchronizing Audio in AIMIT . . . . .	58
4.5	Shockwave Movie . . . . .	59
4.6	Java Sample Codes . . . . .	62
4.6.1	Temperature . . . . .	62
4.6.2	Two-Dimensional Array . . . . .	63
4.6.3	For-While Loop . . . . .	64
4.7	Final testing of AIMIT . . . . .	66
4.7.1	Testing the Director project-AIMIT . . . . .	66
4.7.2	Creating the Projector of AIMIT . . . . .	67
<b>Chapter 5</b>	<b>The Evaluation of the Experiment with AIMIT</b>	<b>69</b>
5.1	Evaluation Methodology . . . . .	69
5.2	Evaluation Methods for AIMIT . . . . .	71
5.3	General Criteria and Expert Review Criteria . . . . .	74
5.3.1	General Criteria . . . . .	74
5.3.2	Expert Review Criteria . . . . .	79
5.4	Summative Evaluation . . . . .	80
5.4.1	Control/Experimental Group Test . . . . .	80
5.4.2	Evaluation Questionnaire . . . . .	85
<b>Chapter 6</b>	<b>Conclusions and Future Work</b>	<b>89</b>
6.1	Conclusions . . . . .	89
6.2	Contributions . . . . .	90
6.3	Future Work . . . . .	90

<b>Appendix A</b>	<b>96</b>
A.1 Ethics Committee Approval . . . . .	97
A.2 Letter of Consent . . . . .	99
A.3 Questionnaires . . . . .	102

## List of Tables

2.1	Informal Comparison Chart of CAI Systems . . . . .	26
4.1	Browsers for Shockwave Movie . . . . .	62
5.1	Organization of First Experiment . . . . .	81
5.2	Organization of Second Experiment . . . . .	82
5.3	Results of the Control Group on the “For/While Loop” . . . . .	83
5.4	Results of the Experimental Group on the “For/While Loop” . . . . .	83
5.5	Results of the Control Group on the “Array” . . . . .	84
5.6	Results of the Experimental Group on “Array” . . . . .	85
5.7	Experimental Group Evaluation Questionnaire Results . . . . .	87

## List of Figures

2.1	WebCT Main Tool Page . . . . .	23
2.2	Main Page of Centra99 . . . . .	24
3.1	Authoring Workstations . . . . .	29
3.2	Sample Code of Lingo . . . . .	31
3.3	Score Window of Director 7.0 . . . . .	32
3.4	Project Diagram . . . . .	35
3.5	Screen Window Layout of AIMIT . . . . .	36
3.6	Navigation Map of AIMIT . . . . .	37
3.7	Sample Page of Story Board . . . . .	39
4.1	Basic Components in Director Movie . . . . .	41
4.2	Score Window of Navigation Lingo . . . . .	45
4.3	Home Index Window . . . . .	49
4.4	Tweening the Sprite . . . . .	52
4.5	Format of Sound File . . . . .	55
4.6	Cool Edit Pro's Editor Window . . . . .	57
4.7	Sound Channels in Score Window . . . . .	57
4.8	Projector Options Box . . . . .	68
5.1	Results of Pre- and Post- Tests . . . . .	86

# List of Notations

## Notations:

AIMIT	An Interactive Multimedia Instruction Tool
CAI	Computer-assisted instruction
CAL	Computer-aided learning
CBI	Computer-based instruction
CBE	Computer-based education
CMI	Computer-managed instruction
CEI	Computer-enriched instruction
CBT	Computer-based training
CD-I	Compact disc-interactive
CD-ROM	Compact disc-read only memory
HTML	Hypertext markup language
ICAI	Intelligent computer-assisted instruction
ISDN	Integrated Services Digital Networks
WWW	World Wide Web
WebCT	Web Course Training



# Chapter 1

## Introduction

“Literally, multimedia is the integration of two or more communications media. It is the use of text and sounds, plus still and moving pictures to convey ideas...it is built around the premise that anything words can do, words with sounds and pictures can do better” [1].

Interactive multimedia identifies a relatively new technology that integrates a variety of media (including video, sound, music, photographs, drawings, computer graphics, and animation) into a single electronic medium, and it provides the ability to link ideas in a nonlinear format. Increasingly, with the rapid advances in digital and computer technology, it is the digital-based storage media that are gaining attention for multimedia delivery. Compact disc read-only memory (CD-ROM) and other digital-based media (e.g. Intranet, Internet) are evolving rapidly, and these new technologies are enhancing the capabilities of multimedia.

“Well-designed interactivity initiates and conducts an electronic conversation between the presenter and the audience, or between the user and the program,” says Robert Lindstrom of New Media magazine. “As the conversation develops, each learns more about the other in an effort to get the information they’re after or to better focus a response” [2].

Certainly multimedia resources are increasingly embraced in elementary and secondary education. In that sector, both the range of products and the applications for this technology are expanding rapidly. By contrast, higher education has not demonstrated equal enthusiasm, but this does not necessarily mean that multimedia have

little to offer.

One of the major achievements of multimedia systems is that they have made it possible to represent many concepts by using animation, 3-D graphics and sound. Well-designed multimedia learning materials can be used to help students learn more easily and more quickly through the use of illustration, animation, different structuring of materials, and increased control of and interaction with learning materials. For instance, when a computer science instructor teaches the programming concept of “Bubble Sort”, he or she can use multimedia technology to translate the concept from a static description into a dynamic process. Thus students can understand the programming concept visually and perhaps learn it more quickly.

Computer Science departments are facing an enormous demand for instructor time because of rapidly increasing student population and class size, and many faculty members, whose own experience was formed when they were students attending university which was then reserved for a relatively small elite, have never come to terms with the implications of “mass” higher education. They view with genuine distress the lack of interaction and communication that results from large class size, and the impact on their research time of increasingly demanding administrative and teaching duties. Many faculty members now recognize that these unpleasant circumstances are unlikely to go away unless alternatives that can provide effective instruction are found.

## **1.1 Multimedia and Objectives of Learning**

Reynolds and Anderson (1992)[3] describe the relevance of multimedia to three objectives of learning:

- Cognitive objectives. Used to teach recognition or discrimination of applicable visual stimuli and audio stimuli.
- Psychomotor objectives. An excellent tool to recreate real world conditions.

- **Affective objectives.** Interactive multimedia is very useful in the affective domain. The strength of detailed portrayal of situations and interactive participation of the learner increases its usefulness for affective domain objectives.

According to Dual Coding theory[4][5][44][45], information is processed through one of two generally independent channels. One channel processes verbal information, such as text or audio. The other channel processes nonverbal images, such as illustrations and sounds in the environment. Information can be processed through both channels. This occurs, for example, when a person sees a picture of a dog and also processes the word “dog.” Information processed through both channels is called referential processing and has an additive effect on recall [6][7]. They conclude that learning is better when the information is referentially processed through two channels than when the information is processed through only one channel. Referential processing may produce this additive effect because the learner creates more cognitive paths that can be followed to retrieve the information.

Soloway[8] describes a situation in which, with the help of multimedia, a visually oriented student was able to use images from art to illustrate abstract concepts in physics. Livergood[9] concludes in his research study that while “the addition of multimedia content and methodology to traditional instructional material does not invariably improve test scores of undergraduate students...presenting material in a modified multimedia intelligent tutoring system can significantly improve test scores.”

## **1.2 Multimedia and Computer-Assisted Instruction (CAI)**

In general, computer-assisted instruction (CAI) is a generic term covering all kinds of instructional systems regardless of the subject domain or their design. They are also referred to as computer-based instruction (CBI) or computer-aided learning (CAL). In particular, CAI refers to earlier instructional systems that are less adaptive to the particular requirements of individual students. Examples of CAI systems include PLATO (Bitzer & Skaperdas, 1970) and TICCIT (Merrill, Schneider, and Fletcher, 1980). These early CAI systems were mostly designed by experienced teachers for the specific domains in which they were expert. The systems were student oriented

and mostly linear CAI: that is, all users (learners) were treated the same and were led through an identical learning path. Later, branching CAI systems evolved with different learning paths for students to follow, depending on their requirements.

There are an increasing number of multimedia applications in CAI, including composition using multimedia, CD-ROM training using multimedia and multimedia-based photoelectronic sensor vision tutorial. The World Lecture Hall ([www.utexas.edu](http://www.utexas.edu)) offers hundreds of courses from universities all over the world, many with multimedia components. Others include San Francisco State University's "Introduction to Multimedia" course, which features excellent design, interactive multimedia, graphics, streaming audio, and chat ([www.cel.sfsu.edu:80/MSP/msp.html](http://www.cel.sfsu.edu:80/MSP/msp.html)), and Banner 2000, which uses Macromedia Authorware (Wayne State University) .

The reason for an expanded interest in multimedia-supported CAI is understandable. According to a research study by the U.S. Department of Defense[47], courses delivered by interactive multimedia instruction systems did as well as or better than, courses given by traditional methods. Research results indicated that:

1. students like interactive multimedia instruction better than traditional methods;
2. students learn more from interactive multimedia instruction than from traditional methods;
3. students probably retain more from interactive multimedia instruction than when using traditional methods;
4. interactive multimedia instruction takes less time than conventional methods.

### **1.3 Multimedia and WWW (World Wide Web)**

Many instructors are now using the World Wide Web both as a presentational tool in lectures and also as a means of making lecture notes conveniently available to students outside of the classroom. The World Wide Web has an additional advantage in that instructors can access other sites from around the world through Internet links, and bring materials from these sites into their lectures.

Another use of the Web is to create databases of slides, photographs, illustrations, and video clips that can be drawn from for a lecture, or made available to students for on-line access. By employing software such as WebCT (Web Course Tool), the Web can also be used to create on-line discussion forums for students and instructors.

One disadvantage of using the Web is that it requires a special computer language, Hyper Text Markup Language (HTML) to create Web pages. Although new Web-development tools and the automatic conversion of word-processed documents into HTML make it easier for instructors to develop Web pages, other developments, such as Java programming, make it more complex.

One of the more explosive growth areas of the World Wide Web is the opportunity it affords for distance learning. With courses ranging from "Rhetoric of the Road" at the University of Texas-Austin, to "Slacker Studies" at the State University of New York-Cobleskill, we are beginning to see video, audio and even some interactivity in many offerings. Of particular interest are the newer interactive multimedia technologies found on the Web. These include Java, Virtual Reality Modeling Language (VRML), Macromedia company's Shockwave , etc..

In one experiment conducted at the University of Massachusetts, the path of a ball on a circular track was videotaped; a segment of the track was then removed and another video segment captured the path of the ball as it was about to leave the track. When queried at this point as to where the ball will go next, many students get stuck; their intuitive senses tell them the ball will go off in an upward path or a downward path but not the straight ahead path that it actually takes. A third video segment showed the actual straight-lined path of the ball leaving the track. An animation done in Macromedia's Director and then put on the Web with Shockwave showed that, a circle is really composed of an infinite number of straight lines and the ball goes off in a straight line. This straight line path is, in fact, counterintuitive to a large number of students. "Our research indicated that students using these multimedia supplements to their regular course materials did significantly better on tests than students who did not use them"[10].

## **1.4 CD-ROM**

CD-ROM technology was introduced in 1985 as a medium for the mass storage of computer-readable text. Each 4.72-inch disc stores approximately 650/800 megabytes (MB) of digital data. CD-ROM discs can store digital text, graphics, audio, and video images. Because of its large storage capacity and high-speed data transfer, CD-ROM is often used for storage and retrieval of large bodies of information such as extended databases and encyclopedias.

CD-ROMs are replacing floppy disks in the computer industry, and CD-ROM drives are standard on personal computers today. More and more software companies are shipping their multimedia products on CD-ROMs. Language laboratories, computer-aided design in architecture, simulated science experiments, and large research databases containing multimedia resources such as graphics, compressed video, and audio are examples of the main uses of multimedia CD-ROMs.

CD-ROMs have been the ideal medium for multimedia production. Currently, multimedia materials with video and audio clips generally require too much network bandwidth for convenient delivery over public Internet systems. A small number of instructors are using multimedia CD-ROM technology to support teaching. Multimedia CD-ROMs are usually used in computer laboratories (where desktop personal computers may be networked to a local server), or on stand-alone computers using a CD-ROM drive.

## **1.5 Multimedia Development Tools**

There is an increasing amount of off-the-shelf software now available that can be integrated into regular classroom teaching or computer laboratory work. Choosing the multimedia development tool is one of the most important project decisions. Faculty should be aware of some key considerations: “development time, distribution, uniqueness of features, technical resources, and target hardware”[11]. At the outset, the prospective multimedia author should identify who will be using the tool and how much time they have to learn it, what specialized features the tool should support, what level of support is offered by the vendor, and with what equipment. In

most cases, faculty will be satisfied with some authoring software such as Microsoft's Word, PowerPoint and FrontPage. Typically, authoring software "not only enables the creation of multimedia-based [materials], but it provides a system to manage the delivery"[12]. They are useful for "situations where one does not envision distributing many copies of the program or a long lifespan for the product"[11]. Authoring systems offer the advantage of being available in a range of complexity and cost. Generally, more complex and expensive programs such as Macromedia's Authorware require a greater investment of time in learning the program. At the other end of the range, Microsoft's PowerPoint or Apple's HyperCard allow the user to create multimedia presentations more easily and affordably, but with less flexibility. The alternative to authoring software is to use a programming language. This is more appropriate "for applications that will be sold off the shelf, or require some special new features or capabilities"[11]. Before any choice is made, the purpose of the project must be clear: "many projects have gotten into trouble midway through, when the development tool, originally selected for its ease-of-use, proves inadequate to accommodate the final design"[11].

Although the number of commercial CD-ROMs suitable for application in higher education is increasing, it is still often difficult to find the right kind of material to meet a particular instructor's needs. Hence, the use of multimedia to support classroom teaching is still relatively low in higher education.

## **1.6 Research Motivation**

New technologies such as the World Wide Web and multimedia have the potential to widen access to students, increase flexibility for "traditional" students, and improve the quality of teaching by achieving higher levels of learning. Under the right circumstances, multimedia technology can lead to improvement of teaching and learning.

The primary goal of this thesis research is to provide a CD-ROM and WWW based multimedia instructional tool for computer science students. The multimedia

tool will be used by introductory-level students to learn the basic fundamental concepts of programming with Java. Two concepts, “loop” and “array” are selected for illustrations in this multimedia tool.

Experience of computer science lecturers informs us that, introductory-level students are generally not capable of visualizing the dynamic process when the instructor employs only static examples. A portion of this Master of Science research is to develop a multimedia tool kit and to assess its value in aiding the student’s ability to learn fundamental Computer Science concepts. The students, in addition to their regular modes of learning (e.g., lectures, labs, etc.), will be allowed to use this tool kit, and an assessment will be made of its merits on various factors, such as effective understanding of concepts, time of learning, etc. This study is significant in terms of guidance for necessary improvements to such multimedia tool kit.

## **1.7 Organization of the Thesis**

This thesis consists of six chapters.

A number of selected CAI tools are investigated in Chapter 2. All these tools have been used to support teaching activities at universities. A literature review of these tools is discussed in detail, including background, capability, strengths and weaknesses, relative cost of creation and maintenance.

Chapter 3 presents how we design and develop of our multimedia instructional tool-AIMIT, such as the delivery platform and delivery media of AIMIT, and the reason why we choose Macromedia’s Director 7.0 as development tool. The authoring workstations and complementary hardware of AIMIT are discussed in this chapter, and the strategy of analysis and design used in this project is also presented.

In Chapter 4, the implementation and delivery details are introduced. The Director’s developing environment of AIMIT and Window Management of AIMIT are discussed. In this chapter, we also describe the process to produce the audio files for AIMIT, and examine the technology of Shockwave movie. Delivering the product is presented in the last Section of Chapter 4.

Chapter 5 describes the evaluation of the multimedia instructional tool, AIMIT.



The general methodology used for the evaluation is introduced, and the evaluation methods used for AIMIT are discussed. The general criteria and expert review criteria applied for formative evaluation are presented in detail in this chapter. Finally, we discuss the methods used for summative evaluation, and examine the results from the experiments.

A brief conclusion is made in Chapter 6, outlining the summary of contributions. Some suggestions for future work are also discussed in this chapter.

A photocopy of the consent form along with the questionnaire used in the project are presented in Appendix A. The Ethical clearance approval letter from the Research Ethics Board is also included in Appendix A.

## **Chapter 2**

# **Literature Review**

In general, computer-assisted instruction (CAI) is a generic term covering all kinds of instructional systems regardless of the subject domain or the design. These systems are also referred to as computer-based instruction (CBI), computer-based education (CBE), computer-managed instruction (CMI), computer-enriched instruction (CEI), or computer-aided learning (CAL)[13][14][15]. In particular, CAI refers to earlier instructional systems that are less adaptive to the particular requirements of individual students.

Early example of CAI systems includes PLATO and TICCIT. Normally, these specific systems were designed by experienced teachers for the specific domains in which they were expert. The systems were student oriented and mostly linear CAI; that is, all users (learners) were treated the same and were led through an identical learning path. Later, branching CAI systems evolved; that is, there are different learning paths for students to follow, depending on their requirements. In fact, many present simulation and learning games also qualify as CAI.

### **2.1 Two typical earlier CAI systems**

#### **2.1.1 PLATO**

PLATO is a timesharing system designed for Computer-Based Education. In 1963, Don Bitzer, a professor at the University of Illinois, Champaign Urbana, was

responsible for organizing a team of engineers to design the PLATO hardware. The partnership between the University of Illinois' Computer Education Research Laboratory (CERL) under the direction of Donald Bitzer, Control Data Corporation, and the National Science Foundation, resulted in the development of PLATO.

The original goal of the PLATO project was to design a computer-based system for instruction. PLATO was designed to use a mainframe-based system rather than a smaller minicomputer because of greater program and storage capability. Kinzer et al.[16] states that "the goal of PLATO was to deliver cost-effective, computer-assisted instruction". A larger library of programs would be available for student use, more sophisticated programs could keep track of individual students' progress, and the number of simultaneous users could be dramatically increased [17].

The PLATO system used a special-purpose programming language called TUTOR to write educational software. PLATO remained a small communications system during the 1960s supporting only a single classroom of terminals. In 1972, PLATO was moved to a mainframe environment that allowed up to one thousand users to connect simultaneously[18].

In 1973, PLATO IV was introduced. It was a large time-shared instructional system. Hundreds of terminals were available with each terminal servicing one monitor and keyboard. All data and programs were stored on a central computer. Hundreds of students could use the system simultaneously to access and use interactive educational and communications software. This system allowed instructors to design instructional material at the same time students were studying lessons [19]. In the Fall of 1973, David R. Woolley designed communications software for the PLATO system called Notes. From this program, other on-line communication programs such as Talkomatic, Term-Talk, Personal Notes, and Group notes were developed and released.

In 1975, Control Data Corporation set up its own PLATO system in Minneapolis and began turning PLATO into a product. By 1985, over 100 PLATO systems were operating at sites around the world, about 60% of them running around the clock.

### 2.1.2 TICCIT

Another earlier major CAI system, TICCIT, was designed to be the primary, rather than the supplemental, medium of instruction for 5,000 college students “using minicomputers, color TV, graphics, and the expertise of content specialists and psychologists well-versed in instructional design” [20]. According to Saettler[23], the MITRE Corporation and the University of Texas initially intended to introduce TICCIT for elementary schools in 1969. Other authors state that TICCIT mathematics and English freshman-level courses were eventually launched at two community colleges, Phoenix College in Arizona, and Northern Virginia College in Alexandria in 1971-72 ([20]; [21]). An instructional design tool called RULEG running on TICCIT was developed at the University of Texas and Brigham Young University, and funded by a grant from the National Science Foundation in 1977[22]. RULEG provides a general statement, or rule, as well as examples of how the rule is applied. Niemiec and Walberg[24] state that this tool was innovative because the “instructional tactics were unique to the system and not particular to the authors of programs.” TICCIT attempted to test the effectiveness of computer-aided instruction (CAI) against the traditional classroom format[23]. TICCIT, together with PLATO, received 60 million dollars in funding from the National Science Foundation, and both were formally evaluated by the Educational Testing Service (ETS)[20].

ETS’s evaluation was mixed. Both the TICCIT mathematics and English course students reported “significant achievement” over the traditional classroom formats; however, more students favored lecture classes over TICCIT English courses, and fewer students completed the TICCIT math courses when compared to the standard[20]. Both the design and the implementation of the TICCIT project had influence upon the development of subsequent Instructional Technology. “For the first time, a large scale project emphasized innovative approaches to hardware as well as in-depth consideration of learning theory and instructional strategies in the design of the course materials[20].” The CAI research highlighted factors that influence effective learning. For example, Chambers concluded “many students simply did not complete the mathematics CAI course, apparently because the faculty paid little or no attention to their needs”[20]. The faculty’s minimal interaction was perhaps the result of fear

of technology or inadequate training. Most instructional development plans today analyze the needs of all users, both students and instructors, and try to build in adequate support. Also, TICCIT strongly emphasized the concept of learner control as well as component design theory[21]. However, the TICCIT math students did not receive sufficient feedback about their progress, and consequently made poor control decisions about what and when to study, to practise, and to test.

Instructional developers who design a tutorial program such as this must embed interaction either into the program itself or through instructor training, and they must also evaluate how much learner control is appropriate given the skill base of the targeted learners.

## **2.2 Intelligent Computer-Assisted Instruction (ICAI)**

Intelligent Instructional Systems (IIS) emerged as a result of increasing emphasis on the system-learner interactivity as well as the application of AI techniques in the instructional process. IIS is generally known as Intelligent Computer-Assisted Instruction (ICAI) or Intelligent Tutoring Systems (ITS). Frequently, these terms are used interchangeably.

In particular, it is believed that ICAI systems are more complete in terms of overall instructional strategy and coverage [48]. They cover course planning, teaching, and tutoring as well as assessment aspects of instruction, whereas ITS may be viewed as a subset or branch of ICAI, as ITS often focus only on the tutoring aspect of instruction. Examples of ICAI development include STEAMER[49] , PROUST [50], LISP-tutor[51] , GUIDON [52].

### **2.2.1 STEAMER**

Before STEAMER, most IT systems were non-graphical. STEAMER is a learning system to help people learn to operate “power stations,” which simulates a graphically interactive interface of the controls and console of a power station. STEAMER is called an “interactive inspectable simulation” by its authors. The graphical components in STEAMER are not only operating elements which take the place of a

keyboard, or mere illustrations of the language content, but an important explanatory part of the communication between system and user. The authors describe the graphical representations on the screen as “dynamic graphical explanations” [49], which are well suited to depicting dynamic recursive models difficult to express with language means. “Such a qualitative graphical interface can operate as a continuous explanation of the behavior of the system being modeled by allowing a user to more directly apprehend the relationships that are typically described by experts” [49].

After training with STEAMER, researchers found that the students could begin to exhibit expert-like behavior by running simulations of the process, by testing hypotheses before resorting to the physical system, and by exploring the system even if it resulted in errors in performance.

### **2.2.2 PROUST**

PROUST (Program Understanding for Students) is an intelligent tutoring system which finds errors (“bugs”) in Pascal programs written by novice programmers. It is not confined to a narrow class of error, but designed to find all bugs in most beginner’s programs. It is claimed [25] that PROUST can identify seventy percent of all the bugs in the programs that students write for moderately complex programming assignments. Having identified the bug, PROUST determines how the bug can be corrected and suggests why the bug arose in the first place. PROUST therefore is part of a wider system which assigns exercises to students, analyses their work and gives them helpful suggestions.

The teaching of computer programming requires a great deal of individualization. A class of 100 students writes 100 different programs, each different in design and with different bugs for the same programming exercise. Bugs in programming arise from a number of different sources: accidental omissions (missing variable declaration or initialization), failure to work out how all the different procedures work together (each piece of the program appears correct, but the program does not run properly), and misconceptions about the language (the program does not do what is expected of it).

If an automatic tutor is to cope with variations and with types of student errors, it must understand what the programmer is trying to do. PROUST achieves its level of competence by being provided in advance with a description of the problems set for the students. Each of the problems given to students is also coded in a frame-based problem-description language and added to PROUST's library. Furthermore, PROUST has a further knowledge base of common bugs in Pascal programs. Therefore, as long as it knows what problem the student is trying to solve, and what possible mistakes are possible in the language, it can identify them in the student's various programs.

PROUST synthesizes each program, looking up the corresponding problem description in the library and forming a hypothesis about the methods by which programmers may solve each part of the problem. If one of these hypotheses fits the student's code, then PROUST concludes that the student is correct. If not, it checks its library of common bugs to see if any of them fit the code.

PROUST has been tested on a large number of Yale University undergraduates. It has also been used on a bank of recordings of student programs submitted to the Pascal compiler. PROUST has managed to score well over 70 percent in the tests identifying all of the bugs in Pascal programs. However, according to Johnson and Soloway, "17 percent of programs are analyzed partially... 4 percent of the programs, deviated from PROUST's expectations so drastically, it could not analyze them at all" [25]. A major problem with PROUST is that when it fails to understand a program completely, its ability to recognize bugs deteriorates dramatically. This indicates the sensitive role of a complete problem-description library.

PROUST has been developed at a cost of half-a-million dollars over a 4-year period at Yale University with two programming assistants. It consists of 15,000 lines of LISP code runs on a DEC VAX750. It takes 3-5 minutes to correctly identify bugs in novices' Pascal programs.

Micro-PROUST is another version of the system running on an IBM PC (512K) with Golden Common LISP. This has supposedly taken one programmer two months to produce and only has one-fifth of the bug catalogue of PROUST. It takes only 90 seconds to run but has an unknown rate of success.

### **2.2.3 LISP-Tutor**

As the result of the experimental nature of work in the area of ICAI, no clear general architecture for such systems can be currently identified. However, the work of Carnegie-Mellon University Psychologist John Anderson, and his colleagues, on the LISP tutor[26], and the geometry tutor[27] is offering a strong hint of a breakthrough.

There are four components to their ICAI architecture:

1. Domain expert: this module is capable of actually solving problems in the domain. This is sometimes also referred to as the 'ideal student' model.
2. Bug catalogue: this is an extensive library of common misconceptions and errors in a domain.
3. Tutoring knowledge: this module contains the strategies used to teach the domain knowledge.
4. User interface: this is the module which administers interaction between the tutor and student.

The student's interface is through a smart screen editor. As long as the student does not make an error, the tutor remains quiet and therefore is seen as no more than an editor. If the student inputs an error in his/her program, the system diagnoses the error and provides feedback in the form of a hint.

The LISP tutor contains approximately 325 production rules about planning and coding LISP programs and 475 buggy versions of those rules. It is claimed to be "effective in diagnosing and responding to between 45 and 80 percent of students' errors"[26]. It can be run under VMS or UNIX operating systems on DEC VAXs. A single work station with 2 megabytes of memory could support one user. With 3-4 megabytes on a VAX730, it can support two users, and with 6-8 megabytes available it could be used as a time-sharing program. The LISP tutor is commercially available from Advanced Computer Tutoring Inc.

The LISP Tutor, has been shown to have a significant effect on student mastery of the LISP. For example, in one controlled experiment, two groups of students attended



the same lectures on LISP programming, and one group completed the exercises in the traditional manner while the other group used the LISP Tutor. The students using the LISP Tutor spent 30 percent less time on the exercises and scored 43 percent better on a post-test than those not using the LISP Tutor[28] . Further classroom use of the LISP Tutor at Carnegie-Mellon University for a one-semester course in LISP showed that students were able, as a whole, to achieve a full letter grade improvement in their final course grade by using the LISP Tutor when compared with previous classes that did not use the tutor. Interestingly, poor students demonstrated the most significant performance improvement[28].

#### **2.2.4 GUIDON**

GUIDON is a system which uses MYCIN[46] for tutoring the students. It includes an independent module containing teaching expertise as well as a limited amount of competence to carry out a coherent dialogue with the student. GUIDON attempts to transfer expertise to the students exclusively through case dialogues where a sick patient (the “case”) is described to the student in general terms. The student is asked to play the role of a physician and ask for information which he thinks might be relevant to this case. GUIDON compares the student’s questions to those which MYCIN would have asked and critiques him on this basis. GUIDON uses a “closed world” model where all objects, attributes and values that are relevant to a case are determined before the tutorial session begins.

GUIDON can therefore provide at any point:

1. A list of all data relevant at that point;
2. Sub-tasks to be performed;
3. Hints on how to go about a problem;
4. A summary of all evidence already discussed;
5. Demonstration of a task;
6. Completion of a diagnosis if the student gives up.

Clancey's work on GUIDON shows a great deal of promise for the use of expert systems in the next generation of CAI systems. In such a frame-oriented CAI system, the medical and teaching expertise is "compiled" into the branching structure of the frames while in GUIDON these are available for inspection and change while the program is running. This also provides the possibility of quick and simple modifications to be made to the system without needing to worry about the fixed structure of the dialogue in advance.

## **2.3 World Wide Web (WWW)-based Tutoring Systems**

### **2.3.1 What is WWW?**

The Internet is the world's largest, most powerful computer network connecting personal computers, sophisticated mainframes, and high speed supercomputers around the globe. Current estimates suggest that millions of computers are part of the Internet.

Because a myriad of computers and programs are part of the Internet, incompatibility problems can result because information is created using different computers and software. In 1989, a group of scientists at the European Laboratory for Particle Physics (CERN) in Geneva, Switzerland began developing an Internet tool that would link information produced by all of the CERN researchers. The tool provided a way

to link textual information on different computers and created by different scientists. The object was to overcome issues of incompatibility and utilize a new way of linking made possible by computers, called "hypertext". Rather than presenting information in a linear or hierarchical fashion, hypertext permits information to be linked in a web-like structure. Nodes of information can be linked to other nodes of information in multiple ways. As a result, users can dynamically criss-cross the information web using pieces in the order most convenient to them. The CERN project resulted in an innovative front-end to the Internet, now referred to as the World-Wide Web (WWW).

The WWW provides users with a uniform and convenient means of accessing the vast resources of the Internet. In 1993, the National Center for Supercomputing Applications (NCSA) at the University of Illinois pushed the CERN scientists' idea further by creating a software tool called Mosaic. Mosaic is an easy to use graphical user interface that permits text, graphics, sound and video to be hyperlinked. Mosaic was the first of the Internet tools that are now referred to as "Web browsers". Other well-known browsers include Netscape (the first commercial browser developed by some of the programmers involved with the Mosaic project) and Microsoft's Internet Explorer. Web browsers permit users to connect to the Internet and facilitate accessing information located on another remote computer. The Web browser links to the remote computer just long enough so that the information you need can be sent to your computer for you to view. Documents created to be viewed by a browser are formatted using Hypertext Markup Language (HTML).

HTML solves incompatibility problems by using standardized tags which indicate such things as whether a piece of text should be plain, bold, italic, or linked to another piece of text. Pages of information on a computer formatted with HTML and accessible to someone with a Web browser, are referred to as "home pages" or "Web pages".

Originally, the WWW was used to retrieve information from all over the world. Very soon, however, it became clear that the WWW could allow for extended interactivity. With the increased utilization of the interactive features of the WWW a lot of learning systems emerged that introduce users into various domains. The

number of learning courses is exploding, and one can see a lot of interesting features that have emerged with the improved capabilities of new WWW browsers. These include Blackboard, Convene, eCollege.com, WebCT, Centra99 Symposium, Virtue-U, etc. As well, some of these WWW-based tutoring systems have been built to support distance learning, among the most popular are: WebCT, Centra99 Symposium, Blackboard, etc.

### **2.3.2 Why Use the WWW for Distance Learning?**

The WWW and Web browsers have made the Internet a more user-friendly environment. The ability to integrate graphics, text, and sound into a single tool means that novice users do not have to struggle with such a steep learning curve. In addition, organizations and individuals can create home pages independently and link to other home pages on their own computers or to pages created by others on different computer systems. For educators, the WWW provides an exciting new opportunity for distance teaching and learning. The WWW can be used by the distance educator to build a classroom home page. The home page can cover information about the class including the syllabus, exercises, literature references, and instructor's biography. The instructor can also provide links to information on the WWW that would be useful to students in the class (e.g., research data on agricultural markets, global climate change, or space missions). Other links can access library catalogs or each student's individual home page. In addition, the home page can link students to a discussion list or listserv set up for student communication. It is also a relatively simple matter to use the homepage to create forms that students can fill out and that will end up being sent to you as an e-mail message.

Distance education allows you to take a course anytime and anywhere. It gives you the flexibility of studying what you want (e.g. a student can register a course he/she is interested in), when you want (e.g. Tuesday evening at 9:00, Saturday morning at 6:30, during your lunch hour at work), and where you want (e.g. from your home, from your office, while travelling). Distance education courses teach the very same competencies as face-to-face courses offered on a campus; the difference is in

the delivery. Instead of having an instructor provide a lecture and lead a classroom discussion, distance courses are delivered over the Internet, by using combinations of audio and/or videocassettes, print and correspondence. Most courses require a textbook and a specially developed course packet. Some courses require CD-ROMs, audio and/or videocassettes. What you will need is specified with each course. You can consult a course schedule or speak with an advisor regarding the equipment needs of each distance learning course.

### **2.3.3 WebCT (Web Course Tool)**

WebCT is a tool that facilitates the creation of sophisticated World Wide Web-based educational environments. It does this in three ways:

1. It provides an interface allowing the design of the presentation of the course (Customize Page, see (1) on Figure 2.1), so the instructors do not need to spend a great deal of time learning HTML.
2. It provides a set of educational tools to facilitate on-line learning, communication and collaboration, such as Bulletin Board(2), Private E-Mail(3), On-line Chat(4), Streaming Video(5).
3. It provides a set of administrative tools to assist the instructor in the process management and continuous improvement of the course, such as Student Management ( i.e., marking).

WebCT was built by educators at the University of British Columbia as a tool to allow educators to build Web-based learning environments without a lot of time, resources or technical expertise. The project leader, Murray Goldberg, won the university teaching prize in the first year as a faculty member at UBC, and has published works in the area of the educational effectiveness and student acceptance of web-based learning resources, both in a distance learning setting and as a supplement to lecture-based courses. Still, WebCT is pedagogically neutral. Its goal is to provide a set of sophisticated tools useful for a broad range of teaching methodologies, and still make it easy to experiment with new techniques.

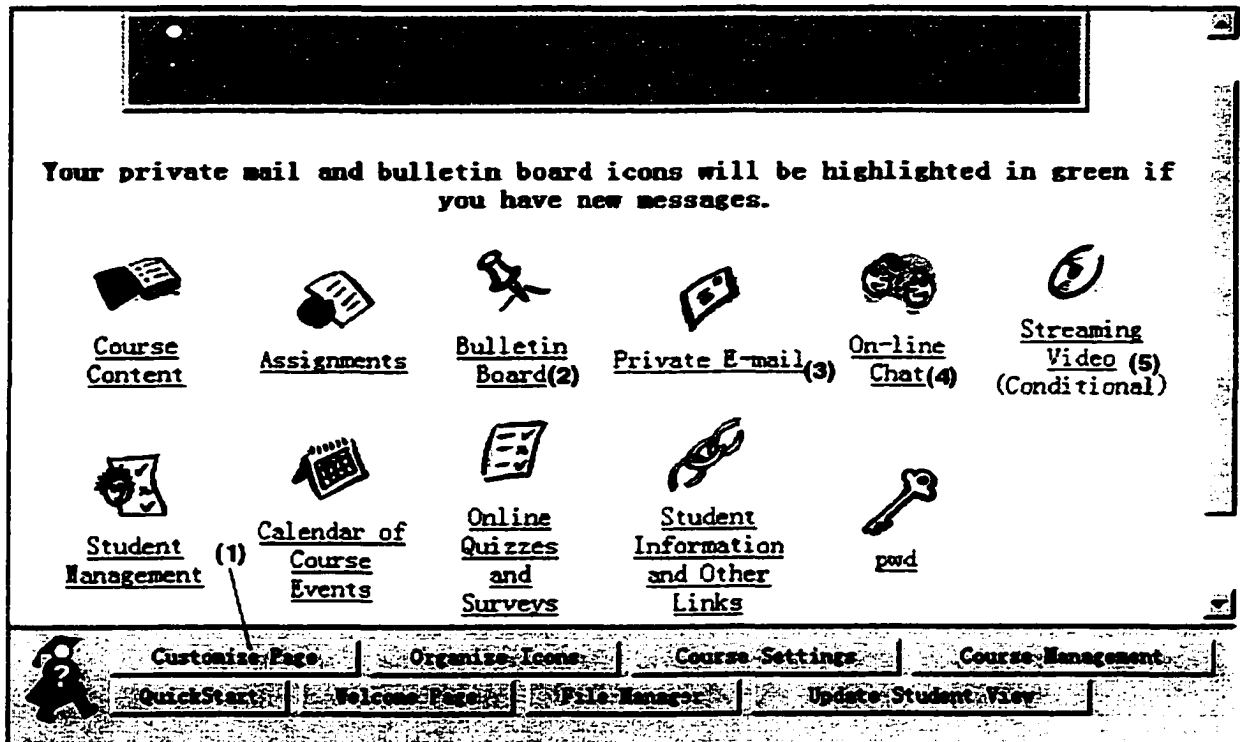


Figure 2.1: WebCT Main Tool Page

Today, WebCT has created a clear leader in the Web-based learning marketplace. As of April 2000, WebCT has more than 6 million student accounts in 147,000 courses at more than 1350 colleges and universities in 55 countries. Students spend an average of 243 minutes each, per course, per month, using this Internet-based application to access course content, take quizzes, submit homework and interact with instructors[55]. By offering a rich suite of course tools, WebCT enables instructors to quickly and easily create and customize their courses. WebCT is becoming popular in University of Regina; more than 200 WebCT courses have been created since it was deployed. In the winter semester of 2001, more than 50 classes used WebCT for supporting teaching.

### 2.3.4 Centra99 (Centra Symposium)

Centra99 is a Web-based enterprise application that enables live eLearning and business collaboration over intranets, extranets, and the Internet by supporting a wide range of live business interactions. Centra Symposium is highly flexible, allowing live eLearning activities to be planned or spontaneous, structured or ad hoc, for internal or external audiences, large or small. Centra Symposium also enables self-paced viewing of recorded events.

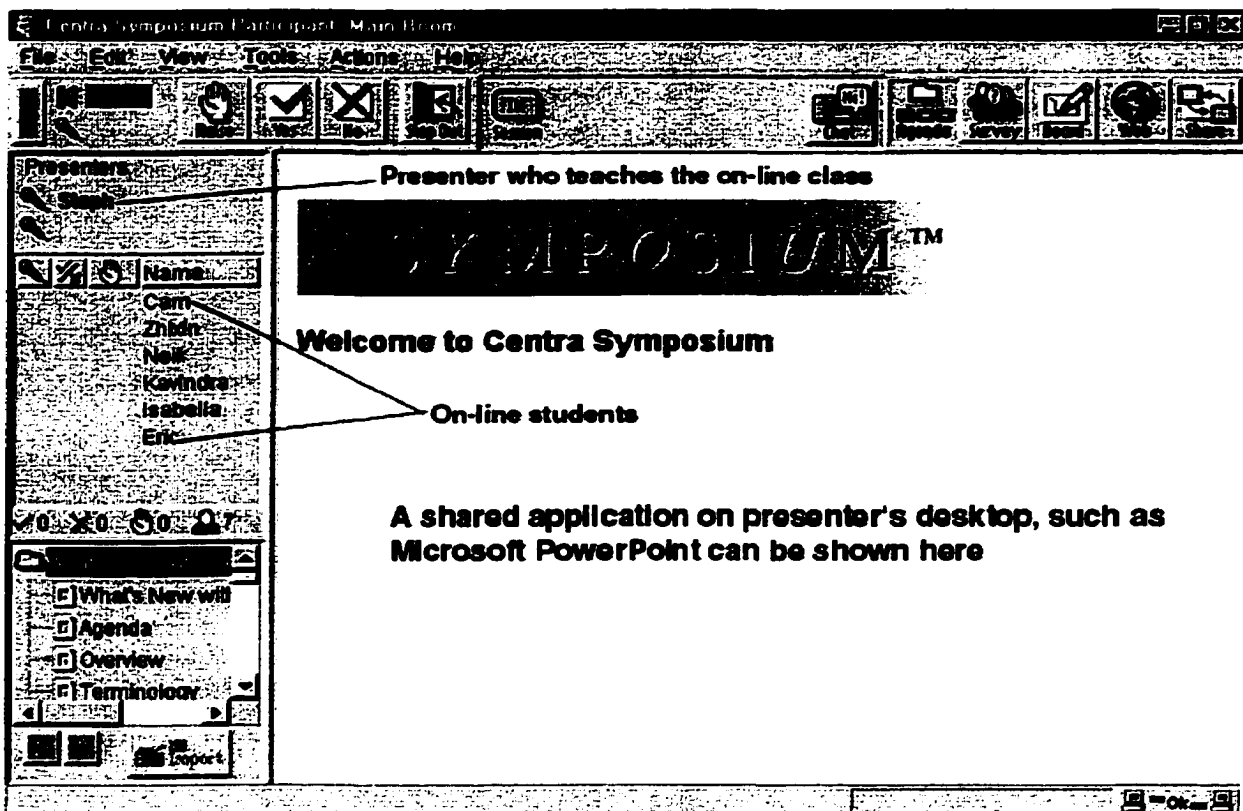


Figure 2.2: Main Page of Centra99

The Centra Symposium client is ideal for highly interactive team collaboration and hands-on training applications. The product presents a broad range of features to provide a rich, collaborative experience, including integrated multi-way Internet audio conferencing, interactive yes/no feedback, live application sharing, whiteboard, Web Safari, breakout rooms, just-in-time content updates, IP multicast option, text

chat, and spontaneous polling and evaluation tools. Each Centra Symposium event can support up to 250 simultaneous users in a live, structured environment.

Centra Symposium Features:

- **Streaming Audio and Video.** Centra Symposium is a live Internet collaboration software to offer fully integrated, multi-way audio conferencing as a standard feature accessible to all users on a LAN, Internet, or low bandwidth dial-up connection.
- **Live Application Sharing.** Centra Symposium delivers high-performance application sharing with integrated multi-way audio conferencing. Centra's flexible application sharing capabilities allow the delivery of applications in three different ways to participant desktops - group interactive, individual hands-on labs, and broadcast - from any PC over low-bandwidth network connections.
- **Uses All Dynamic Content.** Centra Symposium is compatible with all major content formats including video, audio, Microsoft PowerPointAE, graphics and animations, simulations, interactive multimedia, live applications, and other Web-friendly content. Using Centra Agenda Builder, anyone can quickly organize content and agendas for any structured online event. The session leader then displays content on desktops through simple point-and-click delivery.
- **Accessible Delivery.** Centra Symposium supports users on multiple desktop operating systems and browsers, through firewalls, proxy servers and over slow modem or WAN connections. It delivers full functionality over a single, 28.8 kbps dial-up connection.

### **2.3.5 AIMIT**

AIMIT (An Interactive, Multimedia Instruction Tool for computer science education) is built for this master thesis research, which provides a CD-ROM and WWW based multimedia instructional tool for computer science students. This interactive, multimedia instruction tool can be used to support introductory-level students in learning the basic programming concepts of Java. It is developed to aid the students'



ability to visualize the dynamic process of Java programming by using multimedia technology. It can be run from a CD-ROM, so it is portable. It can be posted on a Web site, so it is 24 hour accessible. Details of development and implementation of AIMIT are presented in Chapter Three and Chapter Four. In Chapter Five, the evaluation and experimental results will be discussed.

## 2.4 Informal Comparison

By looking at the development of CAI over the last thirty years, we see that improvement have been made on the richness of feedback and the degree of individualization offered the students. CAI systems have greatly improved in computational sophistication from their humble beginnings of replacing programmed learning machines. The following table summarizes an informal comparison based on several features :

	Plato	Ticcit	Steamer	Prost	Lisp-Tutor	Guidon	WebCT	Centra99	Aimit
Flexibility	Yes	No	No	Yes	Yes	Yes	Yes	No	Yes
Interactivity	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multimedia	Yes	No	Yes	No	No	No	Yes	Yes	Yes
OS	No	No	No	No	No	No	Yes	Yes	Yes
Availability	Access is limited	Access is limited	Limited	Limited	Limited	Limited	All the time	Limited	All the time
Modification	Difficult	Easy	Difficult	Difficult	Difficult	Difficult	Easy	Easy	Easy
Cost	Very High	High	High	High	High	Medium	High	High	Low

Table 2.1: Informal Comparison Chart of CAI Systems

- **Flexibility:** The ability of the system to adapt itself to different strategies without being geared to any particular one.

- **Interactivity:** Allowing the user to learn in an interactive way.
- **Multimedia:** Audio/video clips, animation can be embedded in the system.
- **Operating System (OS):** Capability of a program to be migrated easily from one operating system to another without re-compiling the program.
- **Availability:** It refers to access limits in terms of when and where the system is available, or the cost of using the system.
- **Ease of modification:** Capability to allow the developers to modify the program whenever it is necessary.
- **Cost:** The cost required to build, maintain, and modify the system.

## Chapter 3

# Instructional System Design-The Design and Development Phase

According to James Khazar, who serves as the art director for Macromedia's Instructional Media group, "Multimedia design and development is a complex, arduous, and harder-than-you-think proposition. Producing multimedia projects with quality, at a reasonable price, and on time calls for thinking things out ahead of time; establishing goals and milestones; and working with teams of brilliantly creative people who work like dogs. It's not that much different than any other kind of project where you build a one-of-a-kind product." [29]

Certain rules must be followed in order to achieve an effective multimedia production. These rules are: always plan ahead, keep the users informed, track the implementation against the plan, and test the results.

This chapter introduces the design and development of our multimedia project. Section 3.1 presents the delivery platform and delivery media of AIMIT. Section 3.2 discusses the reasons why we chose Macromedia's Director 7.0 as a development tool. The authoring workstations and complementary hardware of AIMIT are reviewed in Section 3.3. Section 3.4 examines the strategy of analysis and design used in this project.

### 3.1 The Delivery Platform and Media of AIMIT

At the very beginning, we need to decide on the minimum system requirements for running our project–AIMIT. The IBM compatible PC and Apple Macintosh are two popular types of microcomputers currently available on campus and at home; a CD-ROM drive is a standard feature on personal computers; the World Wide Web is the most popular mode of communication today. AIMIT runs on the following platforms: (See Figure 3.1)

- Macintosh
- Windows 95/98
- Windows NT/2000

AIMIT can be burned on a CD-ROM, and be executed directly from the CD-ROM, so it is portable. AIMIT, when saved in Shockmovie format can be Web based, and can be viewed by Netscape, Internet Explorer and American Online(AOL). It is accessible 24 hours a day.

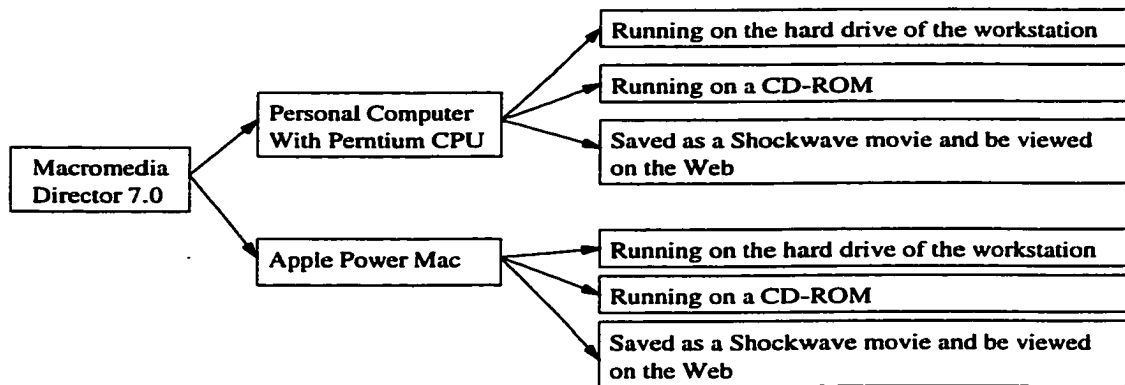


Figure 3.1: Authoring Workstations

### 3.2 Authoring tool–Macromedia’s Director 7.0

Choosing a multimedia development tool is one of the most important project decisions. Multimedia technology is changing rapidly, and the average life of a computer

if often less than four years. Specialized software such as PowerPoint, WebCT, or Director, are constantly being updated and improved. According to R. Strauss, developers should be aware of some key considerations: “development time, distribution, uniqueness of features, technical resources, and target hardware”[11]. At the outset, the prospective multimedia developer should identify the following key elements:

- User—who will be using the tool;
- Cost—how much time they spend to learn it and the cost of development;
- Compatibility— what level of technical support is offered and with what equipment.

In most cases, the developer will be satisfied with some commonly available version of authoring software. Barron and Orwig state that, typically, authoring software “not only enables the creation of multimedia-based [materials], but it provides a system to manage the delivery”[12]. Authoring systems offer the advantage of being available in a range of complexity and cost.

Macromedia’s Director 7.0 is chosen as the authoring tool for AIMIT. The advanced features of it are:

- Lingo is a powerful script language in Director. The best thing about Lingo, (Figure 3.2) in comparison to Java and C/C++, is that Lingo’s syntax is easy to understand, like Pascal’s syntax. Actual English-like commands and events such as [on mouseDown] (see 1 on Figure 3.2) or [go to movie] (see 2 on Figure 3.2) make it easy for the user when programming. On the other hand, a programmer can also use Lingo to the full extent of a complete programming language, such as C.
- Easy-to-use multimedia environment. When compared to other programming languages, such as C/C++ and Java, Director offers an easy-to-use multimedia development environment similar to standard applications, in addition to the powerful scripting language, Lingo.

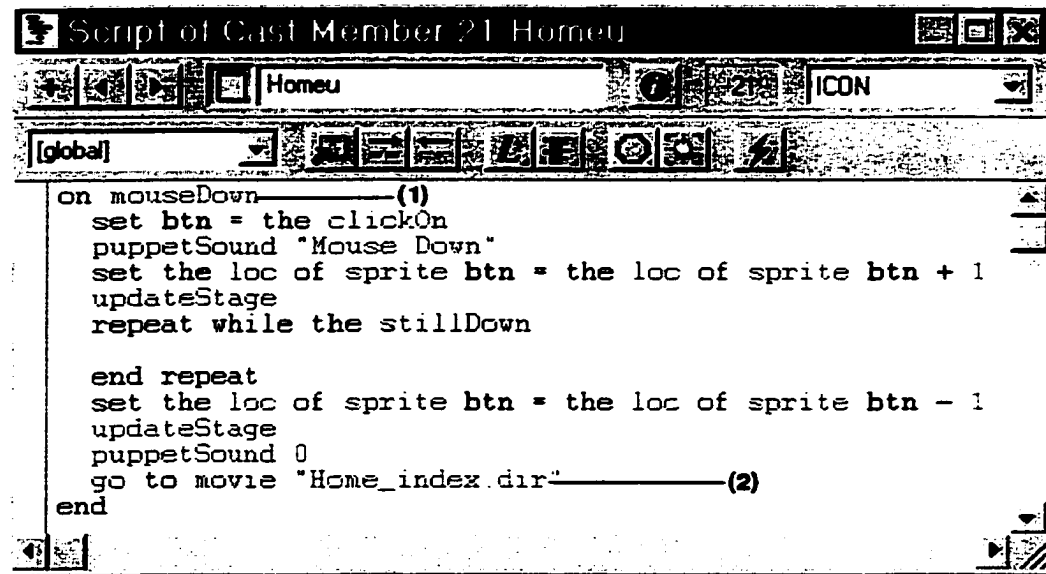


Figure 3.2: Sample Code of Lingo

- The Director Score window (Figure 3.3) provides a visual interface in which animation elements can be examined at every moment in time. It also allows for true cell animation, updating the movements of visual 120 elements in each frame. Behind the stage is the Director engine, capable of 120 frames per second. According to Marc Canter, the founder and later director of Video Works, "Time, as represented by the score window in Director, is the single aspect that differentiates multimedia from flat media, such as paper. After all, time is what animation, sound, and video are all about. Combining the time-line score window and WYSIWYG (what you see is what you get) layout capabilities, you create an intuitive system for multimedia composition." [30]
- Director 7.0 accepts and integrates media elements created in specialized tools for sound, digital video, and graphics. It supports a multitude of new graphics file formats, including GIF, JPEG, Photoshop PSD.

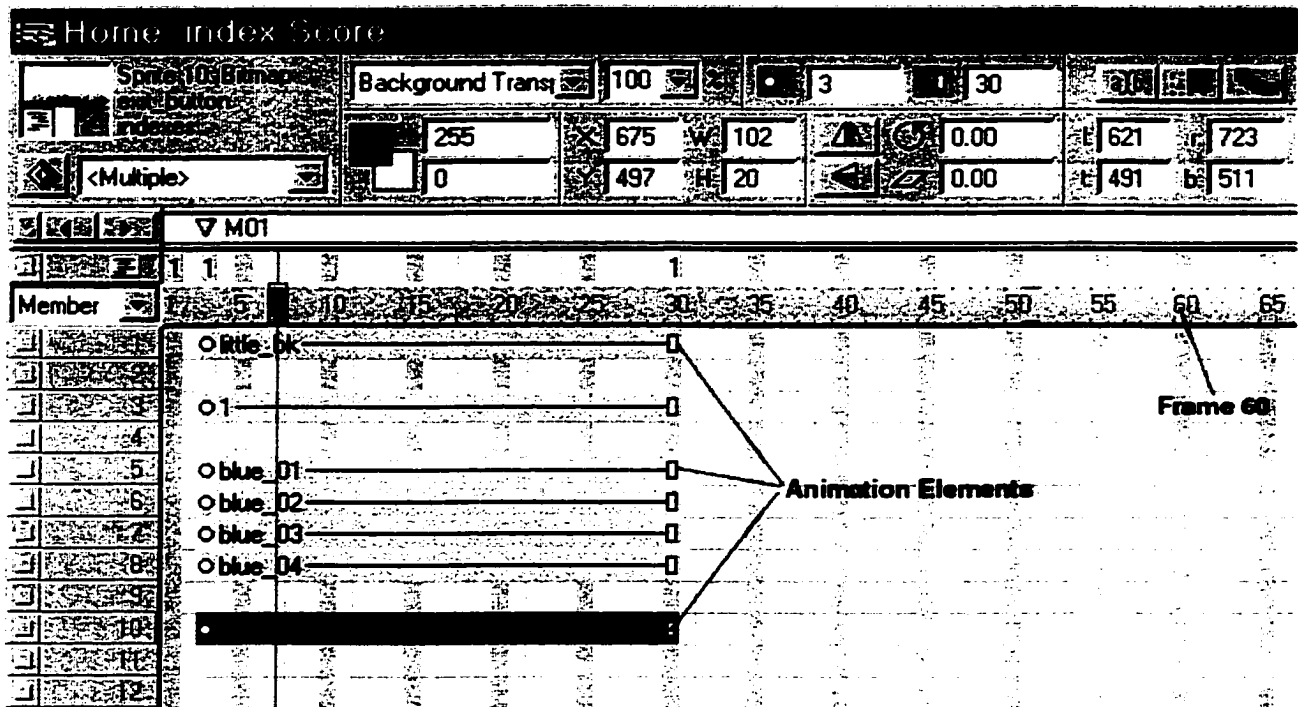


Figure 3.3: Score Window of Director 7.0

- Director's Authoring

- Director movies burned on CD. CD-ROM drives are standard on personal computers today, and CD-ROMs are a perfect medium for multimedia productions. What could be better than 650 MB of free space to fill up with stuff? The new DVD-ROM drives promise 4.7-8.5 gigabytes of storage space. Director projects, like AIMIT, can be burned on a CD-ROM by using a CD burner.
- Shockwave movie for Internet. One of Director's most powerful features is its integration with the Internet. AIMIT saved in Shockwave format can be embedded on the Web, and Shockwave for Director is the number one Internet plug-in today. It enables Web pages to display Director movies full of streaming audio, rotating logos, transitions, sound effects, interactivity, and more.

### **3.3 Authoring Workstations and Complementary Hardware**

The authoring workstation we used for AIMIT is as follows:

- 1) Intel Pentium II processor;
- 2) 16-bit video;
- 3) Microsoft Windows 98;
- 4) 64 MB RAM;
- 5) 6 GB hard drive space;
- 6) A 17-inch monitor;
- 7) Audio card;
- 8) 8x speed CD-ROM;
- 9) Headphone (with microphone);
- 10) Scanner;
- 11) CD-ROM burner.

### **3.4 The strategy: Project Analysis and Design**

#### **3.4.1 Define the goals**

During the analysis phase, we need to determine the goals and project specifications. The first step in defining these goals is to ask questions, such as, “What is this research project trying to accomplish?” “What programming concepts are you trying to teach?” “Why does the user want to use the product?”.



### **3.4.2 Resources inventory**

First, we need to collect background images, icons, sounds, color palettes, cursors, text books, Java programming source codes, audio text display, and put them into resources kits. These kits are independent building blocks for multimedia programming. Some resources, such as background images, are shareable. The collection of colours used in a picture form a palette. Pictures can be scanned in or copied from existing document files. Images can be edited either in Director's paint window or by using Adobe Photoshop. Sound effects, voice, and music can be recorded and reproduced. A significant amount of memory space is required for sound and video resources, so at least 64 MB RAM should be installed on the authoring workstation. Storage space for resource files can normally be reduced significantly if data compression technique is used.

Organizing resources into categories makes development and updating easier. The following resource categories are used for the development of AIMIT:

1. A stock of content, such as text, graphics, digital audio and video that will be digitized, and all other collateral material.
2. A list, or spreadsheet, of the content that needs to be created from scratch, which indicates media type, dimensions, resolution, time length.

Creating and editing content is often the most expensive part of a project, so making a list or spreadsheet is a good way to estimate costs and keep the budget in line.

### **3.4.3 Information (knowledge) Design and AIMIT Diagram**

In this step, a structure was designed for presenting the information (knowledge) we want the students to learn. Knowledge and expertise in Java programming was organized into manageable chunks based on our stated project goals (See Figure 3.4). Topic headings and subheadings were written hierarchically and links were drawn to connect the topics. Links between sections will become the paths that a user follows to access the information. Generally, links should flow logically between sections

without unexpected jumps or leaps. The user can then easily find the most important information (knowledge). Does it take more than three jumps to arrive at any place in the information structure? If so, we need to consider reorganizing, offering navigational shortcuts, or both. A simple diagram was drawn to focus on the information content and the logical connections among various forms of information. By breaking down the information into sections, drawing links, and creating a diagram, we set the stage for the storyboard, in which we start adding media and designing interfaces.

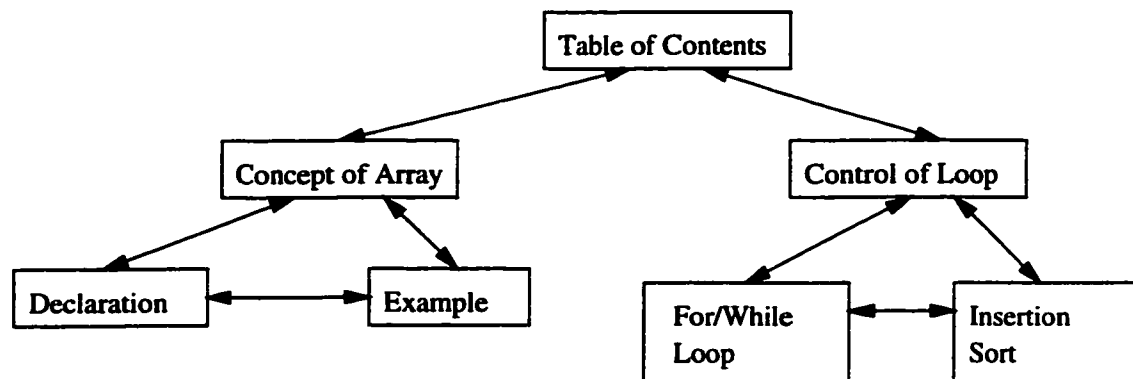


Figure 3.4: Project Diagram

### 3.4.4 Graphic User Interface (GUI) Design

AIMIT provides a multitude of design attributes, including text, visual aids and sounds, which allow users to learn in a variety of ways.

Consistency between applications is essential in AIMIT. A level of consistency is brought to every screen, but “look and feel” issues are also considered as well. According to the GUI design principle, the use of labels and icons must always be consistent. The same label or icon should always import the same meaning, and conversely the same meaning should always be represented by the same label or icon.

In addition to consistency of labelling, objects are also placed in a consistent manner (See Figure 3.5). Each screen window is clearly labelled, with the label appearing in the same location on every screen. A button bank appears in the same place along the left side of every screen. Some buttons must change to accommodate

the needs of any given screen window, but positionality is used consistently.

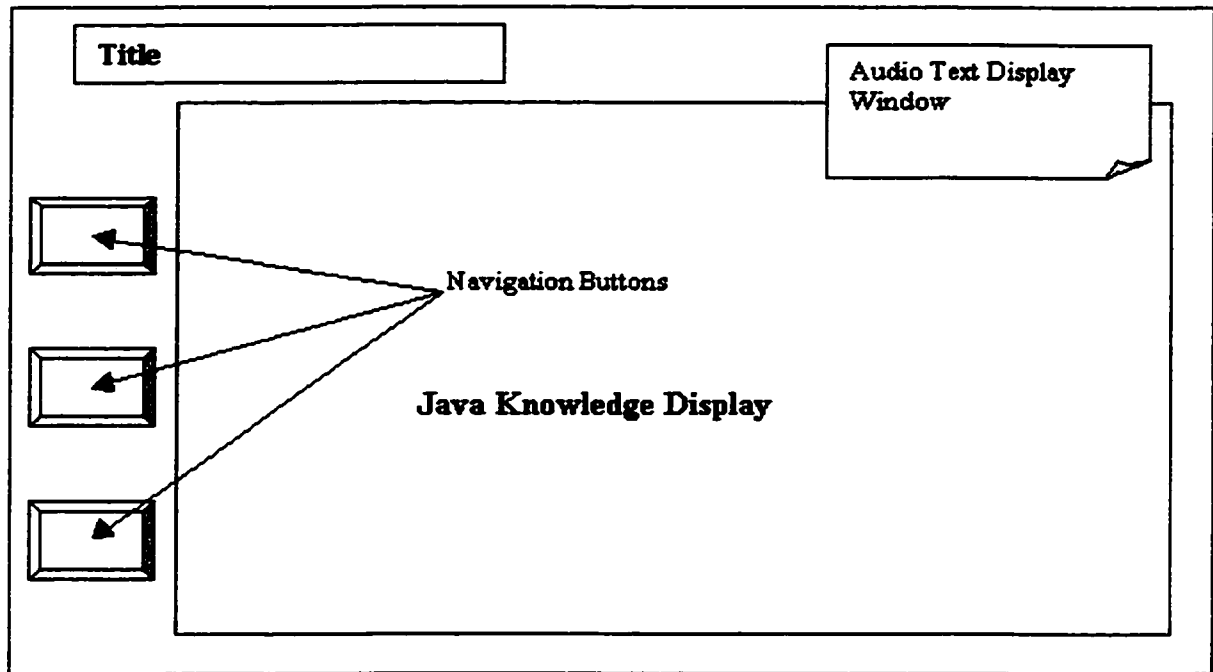


Figure 3.5: Screen Window Layout of AIMIT

### 3.4.5 Navigation and Interactivity

AIMIT is designed in a non-linear fashion (See Figure 3.6). This allows users to enter the course where they need the most help and skip sections that they have already mastered.

User interface of AIMIT provides permanent objects as unchanging reference points around which the users can navigate. If they ever become lost or disoriented, they should be able to quickly find the permanent objects, and from there, get to where they need to be. On the Macintosh, the apple menu and applications menu are examples of permanent objects, no matter what application the user is in, those objects will appear on the screen.

AIMIT provides a link to the table of contents on every screen to minimize jumping through hoops to navigate an information structure. All navigation is reversible.

If the user jumps somewhere, he/she should be able to jump back. Buttons and controls that have related functions are grouped together and kept in a consistent screen location.

The function of each button and control is clear in AIMIT. Different looking buttons do different things. If the button that takes the user to the next screen is a right-pointing arrow, a similar right-pointing arrow will not be used for another function. We use both icons and text to label the navigation buttons, and the first letter of each main word in the main label is capitalized. Buttons can provide feedback when a user clicks on them, like playing a mouse click sound and moving a little bit. Buttons that do not respond when a user clicks on them are lifeless and confusing.

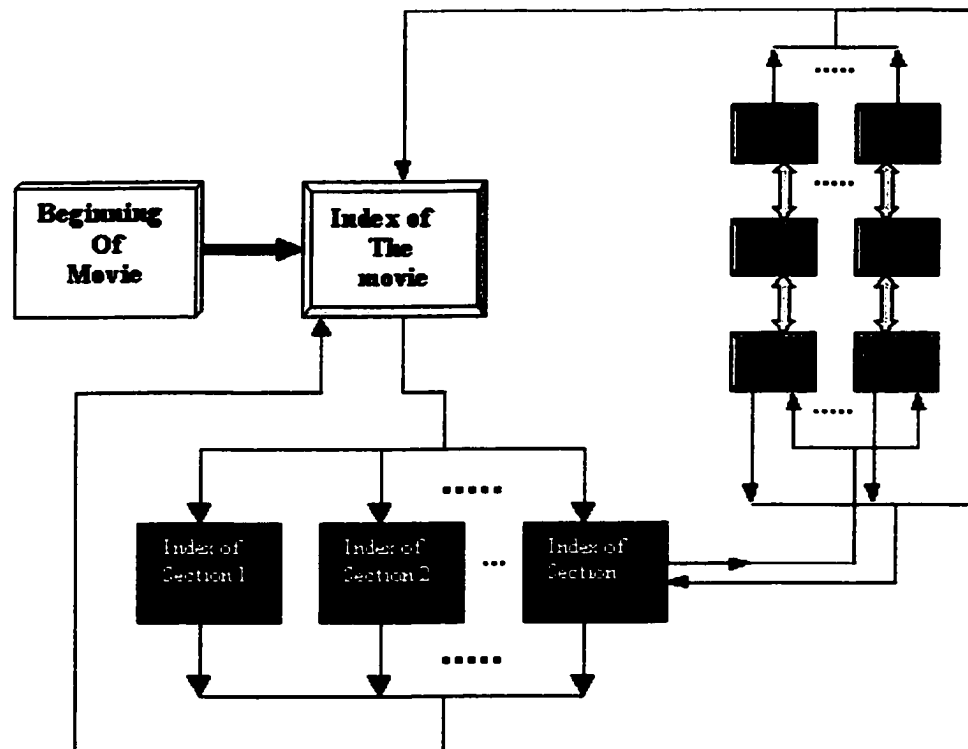


Figure 3.6: Navigation Map of AIMIT

### **3.4.6 Creating Storyboard**

The storyboard is the main outline for the project. It merges the diagram, script, screen layout, and navigation design into a single document.

1) **Script Gathers** the narration and text (audio display text) and puts them into one document in the order that they will appear in AIMIT. This document is the text script. Some of AIMIT's slide screens are heavily time-based and require synchronizing audio, voice narration, and screen display. We create a time bar for these screens. Text script is used as a basis and used to insert audio and animation cues when and where they should occur.

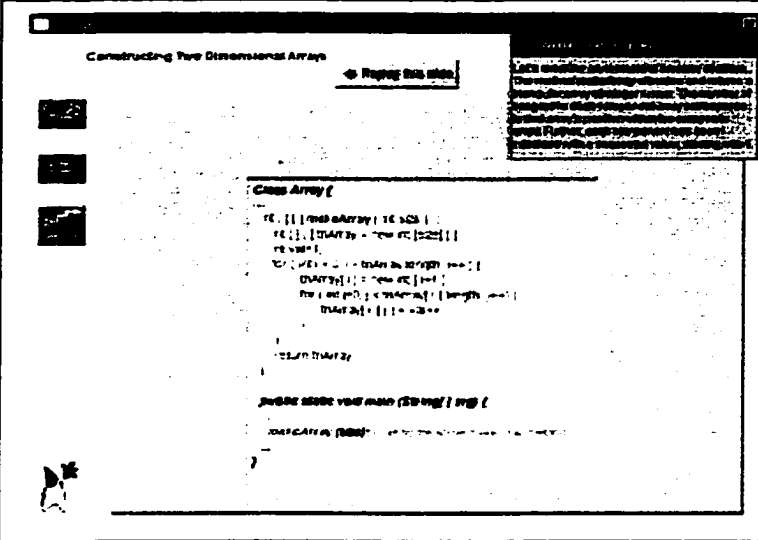
2) **Screen layout**

Screen layout defines the position of interface objects and media on the computer screen. Designers in print media have traditionally used grids to provide consistency, balance, and structure in page design and layout. Director 7.0 comes with the capability to define grids on the stage during authoring. Screen layouts in AIMIT have many of the same design criteria as print pages. A grid as a framework is used for designing screens. The design process is as follows:

- Create a grid by drawing a chart like the above Figure 3.6.
- Add the navigation controls and screen objects that will be common to all screens. These include such objects as headings and labels.
- Add background picture and layers to the screen.

● **Creating the sample page of the story board**

Now it is time to create the storyboard pages. A sample page is shown in Figure 3.7. The number of screens (slides) for each section of the diagram should be determined, and the content to be displayed on the screen should be decided. The storyboard is the roadmap for production; it provides a visual overview of the entire project. By updating the storyboard to reflect completed, added, deleted, or revised elements as the project progresses, we can get a quick idea of how the development process is proceeding.



The screenshot shows a presentation slide with a title bar 'Constructing Two Dimensional Arrays' and a 'Replay this slide' button. On the left, there are three navigation buttons. The main content area displays Java code for a method named 'makeArray' that returns a triangular array of integer arrays. A small Java icon is visible in the bottom left corner of the slide.

**Artwork:** A Java icon is shown on the left bottom of the screen.

**Animation:** None on this slide.

**Audio Text Display:**  
Let's examine an example of an array of arrays. The method `makeArray` allocates and returns a triangular array of integer arrays. The number of integers in each component array corresponds to that array's position within the composite array. Further, each component has been initialized with a sequential value, starting with 1.

**Java Sample Code:**

```
int [][] makeArray ( int size ) {
    int [][] triArray = new int [size] [] ;
    int val=1;
    for ( int i=0; i < triArray.length; i++){
        triArray[i] = new int [i+1];
        for ( int j=0; j < triArray[i].length; j++)
        {
            triArray[i][j] = val++;
        }
    }
    return triArray;
}
```

**Interactivity:** A HAND cursor will be shown when the cursor passes over the Buttons.

Buttons depress and make a sound when the user clicks on them.

**Home** button goes to Home Index with "go to" command.  
**Back** button goes back to last section with "go to movie" command.  
**Next Step** button goes to next slide with "go marker(2)" command.  
**Replay this slide** button relays this slide with "go marker(1)" command.

Figure 3.7: Sample Page of Story Board

## Chapter 4

# Instructional System Design-The Implementation and Delivery Phase

In this chapter, the implementation and delivery details of the project are introduced. Director's developing environment for AIMIT is presented in Section 4.1. In Section 4.2, Window Management of AIMIT is discussed. Section 4.3 describes the process of producing audio files for AIMIT. Section 4.4 examines the technology of Shockwave movies. The Java source code used in AIMIT is presented in Section 4.5. Delivering the product is presented in the last Section—4.6.

### 4.1 Director's Developing Environment for AIMIT

In this section, the development environment of Macromedia's Director will be discussed in detail.

#### 4.1.1 The Basic Components of Macromedia Director Movie

When creating and editing a Director movie, we typically work within the four key windows that make up Director's work area: the Stage, the rectangular area where the movie plays; the Score, where the movie is assembled; one or more Cast windows, where the movie's media elements are assembled.(See Figure 4.1)

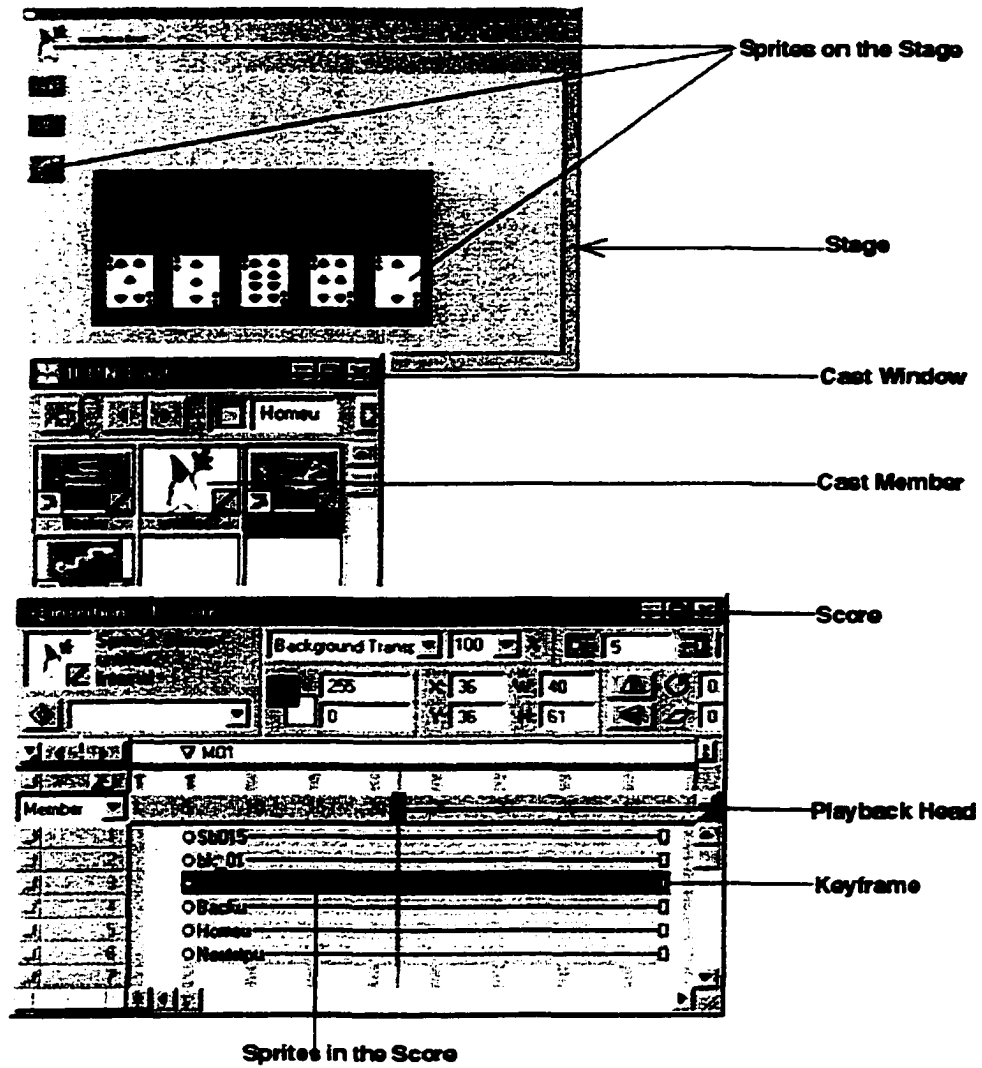


Figure 4.1: Basic Components in Director Movie



- **The Score and the Stage**

The Score coordinates the movie's media, determining when images appear and sounds play. Special channels control the movie's tempo, sound, and colour palettes. The Score also assigns scripts-Lingo instructions that specify what the movie does when certain events occur in the movie. The Stage is the visible portion of a movie. The Stage is used to present where media appear. Working together, the Score's settings and controls create a dynamic, high-quality interactive piece that plays in a Web page or as a stand-alone application directly on the user's local computer.

- **Sprites**

Sprites are objects that control when, where, and how media appear in a movie. The media assigned to sprites are cast members. Director organizes cast members in libraries called casts (See Figure 4.1). Creating a Director movie consists largely of defining where sprites appear on the Stage, when they appear in the movie, how they behave, and what their properties are. We work with sprites on the Stage and in the Score to change where and how cast members appear in the movie. In the Score window, sprites appear as horizontal bars spanning the frames in which the sprite appears. To make sprites appear at different times, move the bars to different frames.

A cast member can be any media element in a Macromedia Director movie. Every media element used in AIMIT—shapes, text, sounds, digital movies, vector images, scripts—becomes part of the cast members. Director organizes cast members into groups called casts. Once we have created the necessary casts, we can import and create cast members to populate them. Cast members are the raw media elements of a movie.

The flow of events is orchestrated in a time line in the score window which is composed of frames. We bring cast members to the score to appear during specified ranges of frames. Cast members become sprites when placed in a channel. The channels tell the sprites where they are and what colour they should be, and so on. The sprite appears on the stage for whatever frames

we designate. The same cast member can be reused throughout the movie in different sections.

To create sprites, drag cast members to the Stage. We can create as many sprites as we need from a single cast member. Sprites are controlled by the Score window.

### **4.1.2 Interactive Action Lingo**

Lingo is referred to as a modified object oriented language. It adds interactivity to a Director movie. We use Lingo to accomplish the same tasks, such as moving sprites on the Stage or playing sounds.

Much of Lingo's usefulness, however, is in the flexibility it brings to a movie. Instead of playing a series of frames exactly as the Score dictates, Lingo can control the movie in response to specific conditions and events. For example, we can use Lingo to create animation, stream movies from the Web, perform navigation, format text, and respond to user actions with the keyboard and mouse.

Writing Lingo also permits us do some things that the Score alone can not do. For example, Lingo's lists allow us to create and manage data arrays, and Lingo operators let us perform mathematical operations and combine strings of text.

Everything created in Director, for example, images, sounds, text is called an object. These objects become cast members and once placed on the stage, each instance is called a sprite. We create Lingo scripts to give the objects the appearance of intelligence to do things. For example, user actions send messages to objects, such as when the mouse button has been pushed down with the pointer sitting over an image. Normally nothing happens. However, we can add a Lingo script to the button image (btn) so that it will respond to the press a button, perhaps, by moving to a certain place in the score and playing a sound. Director can also send messages to itself. For instance, a timer can cause actions to happen after a certain amount of time has passed.

We can use Lingo to turn a sprite into a puppet. A puppeted sprite can then be controlled by Lingo. To create a puppetSprite, use the following Lingo code:

```
puppetSprite 2, TRUE
or
set the puppet of sprite 2 to TRUE
```

This script sets the puppetSprite of the sprite in channel 2. The script can be placed in a movie script, a score script, or in a handler such as on mouseUp or on mouseDown. The puppeted sprite can now be moved to a new location by using locV and locH Lingo commands or changed to another cast member by using the memberNum command.

### **Lingo Syntax of User Action used in AIMIT**

- Movement of the mouse pointer - entering, staying within, and leaving the boundaries of an image sprite: mouseEnter, mouseWithin, mouseLeave
- Clicking the mouse button while pointing at something, double clicking: mouseDown, mouseUp, stillDown, hyperlinkClicked, mouseUpOutside, clickLoc, clickOn
- Location of the mousepointer on the screen: functions called mouseLoc, mouseH (horizontal), and mouseV (vertical)
- Entering text in a text entry field: the text of member “example”
- Which keyboard keys are pressed: keyDown, keyUp. controlDown, optionDown, shiftDown, the key
- Opening, closing, moving, and deactivating windows: openWindow, closeWindow, moveWindow
- The starting or stopping of the movie: startMovie, stopMovie, prepareMovie

## **4.2 Object-oriented programming in Director**

The advantage of object-oriented programming (OOP) is in the way it blends the capability to duplicate code with capacity to customize code. Inheritance is an

OOP term that refers to the capability of one script to incorporate the methods and properties of another script. Inheritance is the way of reusing existing code for similar programming problems, rather than writing code completely from scratch every time.

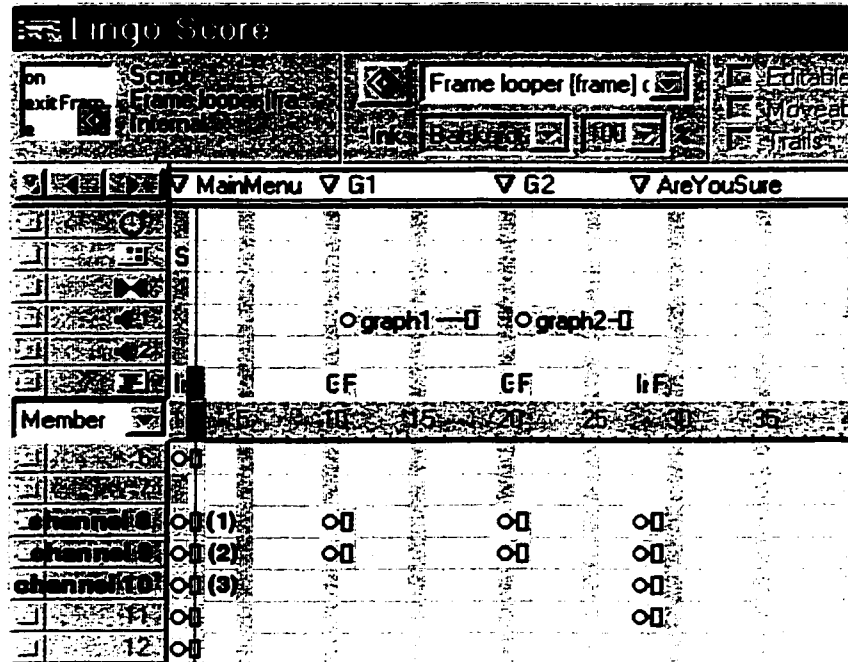


Figure 4.2: Score Window of Navigation Lingo

## Lingo

The following Lingo program is used to change the cursor to a hand when a user moves it onto clickable navigation buttons on screen. Buttons provide feedback when a user clicks on them. To attach the correct Lingo behaviors, customCursorOn to the buttons in channels 8,9 and 10 (See Figure 4.2), the concept of inheritance of OOP is used. (“-” indicates comments of the Lingo code.)

```
--Navigation Lingo source code
global gVisitedG2,gVisitedG1
    global gCustomCursorList,gPointerCursor
    set gCustomCursorList = [ ]
```

```

    set gPointerCursor = [the number of member 'HandCursor',the number of
    member 'HandCursorMask']]
end

```

Declaring Parent global variables makes them available to all Child objects in the movie. To use a global variable in a handler, we must declare it before we use it, as seen in the following: gCustomCursorList (Parent object)- list of sprite numbers that have custom cursors turned on.

gPointerCursor (Parent object)- the custom cursor bitmap and mask

```

on customCursorOn whichSprite

```

```

    -- Parent method

```

```

    -- whichSprite - the sprite number to give a custom cursor to

```

```

    global gCustomCursorList,gPointerCursor

```

```

    set the cursor of sprite whichSprite to gPointerCursor

```

```

    -- keep sprites with custom cursors in a list so we can

```

```

    -- turn the cursor off before going to a new section

```

```

    append gCustomCursorList,whichSprite

```

```

end

```

```

on customCursorOff

```

```

    -- Parent method

```

```

    global customCursorList

```

```

    repeat with listPosition = 1 to count(gCustomCursorList)

```

```

        set the cursor of sprite getAt(gCustomCursorList,listPosition) to 0

```

```

        -- get rid of the custom cursor for each sprite number found on the list

```

```

    end repeat

```

```

    set customCursorList = [ ]

```

```

    -- reset the custom cursor list since nothing has a custom cursor any more

```

```

end

```

```

on mainCursors
--Child object, give the buttons on the MainMenu screen a hand cursor
  repeat with buttonSprite = 8 to 10
    customCursorOn(buttonSprite)
  end repeat
end

```

```

on g1Cursors
-- Child object, give the buttons on the G1 screen a hand cursor
  global pointerCursor
  customCursorOn(8)
  customCursorOn(9)
  customCursorOn(10)
end

```

```

on g2Cursors
-- Child object, give the buttons on the G2 screen a hand cursor
  global pointerCursor
  customCursorOn(8)
  customCursorOn(9)
  customCursorOn(10)
end

```

From the above, we can conclude that, by using OOP, we only need to declare the methods on customOn/Off script for the Parent object once, then we can use it repetitively on the multiple Child objects-mainCursors, g1Cursors and g2Cursors in channel 8,9 and 10.

### **4.3 Window Management (User Interface) in AIMIT**

In AIMIT, a unified window management, supporting a training environment of multimedia, is incorporated to provide the means for text or image display under

different windows and for the user to interact with applications via buttons, animation, and so on. Different-sized windows are made available to suit presentations of different purposes such as Home Index window, background, or pop-up window. An optimal window size is chosen to suit the information being presented.

### 4.3.1 Home Index Window

- **Animation:** 12 images of 3-D fonts “Step By Step” (Figure 4.3) created in Corel Draw 8.0 were imported into the Cast Window, and inserted into the Score Window by using the animation technique “Cast to Window”. Use Cast to Time to move a series of cast members to the Score as a single sprite. This is one of the most useful methods for creating animation with multiple cast members. Typically, we can create a series of images and then use Cast to Time to quickly place them in the Score as a single sprite. Director’s onion skinning feature is also useful for creating and aligning a series of images for use in animation.
- **Text:** This slide shows the index used to navigate from scene to scene in AIMIT. It shows:
  - Introduction to Arrays
  - Introduction to For/While Loops
  - Exit
- **Interactivity:** A simple mouseclick continues. The Exit button quits the program if the user clicks on it. Index topic fonts’ color will change to red when the mouse rolls over them, the following Lingo programming shows how to achieve the interactivity:

```
On startmovie
  global L
  -- reset the last image position
  set L = 24
```

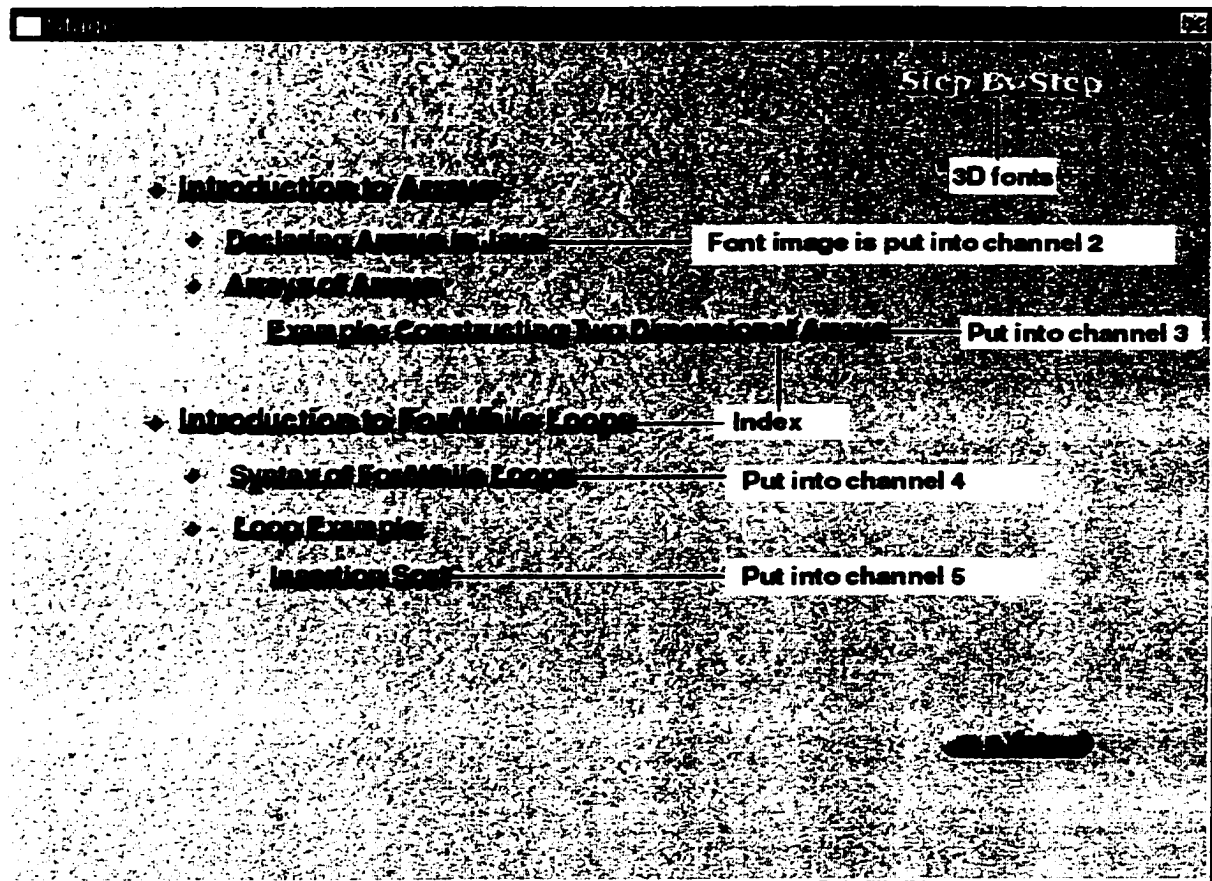


Figure 4.3: Home Index Window

```

-- make the map the only thing which acts
-- tell the stage to set the modal of window 'Index Window' = TRUE
-- Prepare highlighting channels (Font Image) for rollover (Figure 4.3)
repeat with ch = 2 to 5
  --Set ch to channel 2 to channel 5
  puppetsprite ch,true
end repeat

-- Prepare switch images to indicate where we are
repeat with ch = 7 to 10
  puppetsprite ch,true

```



```

        set the ink of sprite ch = 7
    end repeat

on activatewindow
--Parent method
    global L
    set the ink of sprite L = 7
    repeat with ch = 7 to 10
        puppetsprite ch,true
        set the ink of sprite ch = 7
    end repeat
end

on stopmovie
--Parent Method
    repeat with ch = 2 to 10
        puppetsprite ch,true
    end repeat
end

on idle
--Child objects
    global L
    -- ink 39 is darkest
    -- ink 7 is not ghost
    repeat with ch = 2 to 5
        if rollover(1) then
            set the ink of sprite 1 = 7
            set the ink of sprite 2 = 39
            set the loc of sprite 2 = point(196,101)
        else

```

```

    set the ink of sprite 1 = 39
    set the loc of sprite 2 = point(-175,194)

end if
...

if rollover(4) then
    set the ink of sprite 4 = 7
    set the ink of sprite 10 = 39
    set the loc of sprite 10 = point(4,179)
else
    set the ink of sprite 4 = 39
    set the loc of sprite 10 = point(-187,272)
end if

end repeat

end

```

### 4.3.2 Tutoring Window

The most important animation technique, Tweening of Director, is used widely for the tutoring window in AIMIT.

We can tween an image (a sprite) directly on the Stage by editing its path. Director displays the path of the selected sprite directly on the Stage. We can adjust the path by dragging keyframe indicators (See Figure 4.4).

**Here are the steps to tween the path of a sprite:**

1. Select a sprite on the Stage where we want the path to start. This places the start frame of the sprite in the proper location. The start frame is also the first keyframe of the sprite.

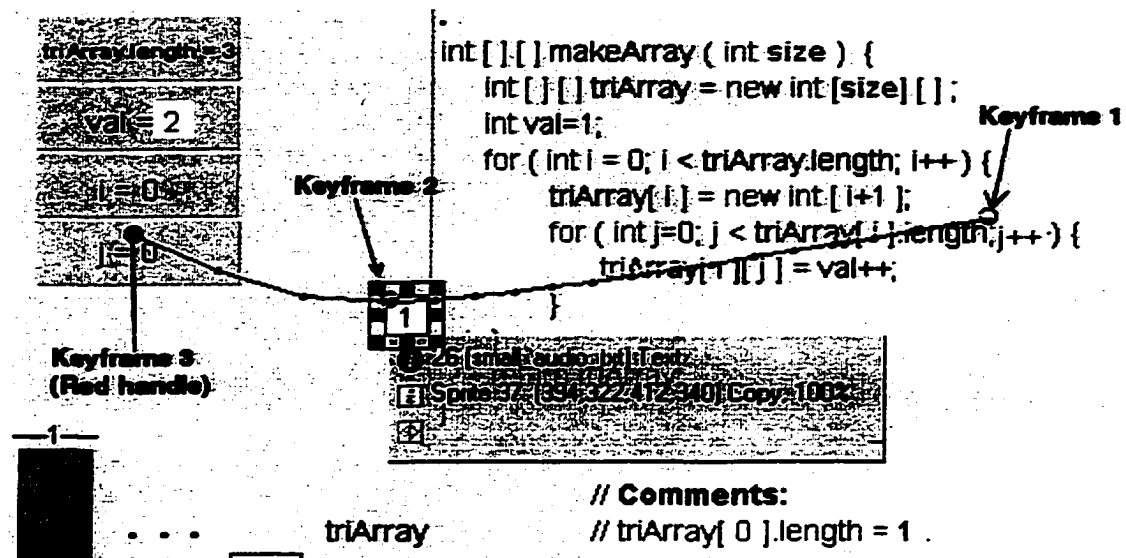


Figure 4.4: Tweening the Sprite

2. If necessary, choose View > Sprite Overlay > Sprite Paths.  
With “Show Paths” option turned on (See Figure 4.4), Director displays the paths of moving sprites on the Stage. Keyframes appear as hollow circles. Small tick marks show the sprite’s position in tweened frames.
3. Drag the red handle within the sprite to the place on the stage where we want the sprite to move. The red handle represents the sprite’s location in the end frame. For bitmaps, the red handle is usually in the center of the image. For vector shapes and other media types, the handle is often in the upper-left corner. Director displays the path the sprite will follow. The tick marks along the path show the sprite location in each frame in between.
4. To make the sprite curve between more points, hold down the Alt key (Windows) or Option key (Macintosh) and move the pointer on the Stage over a tick mark. When the pointer changes color, drag the tick mark to a new location.
5. To make the property changes defined by a keyframe occur at a different time, drag the keyframe in the Score to a new frame within the sprite.
6. To change the degree of curvature between keyframes, choose Modify > Sprites

> Tweening and adjust the Curvature slider.

7. To make the sprite move in the same direction at the beginning and at the end, turn on **Continuous at Endpoints** in the Sprite Tweening dialog box. This creates a circular motion.

The use settings in the Sprite Tweening dialog box can be used to to create more natural motion in tweened sprites. **Ease-In** makes a sprite move more slowly in the beginning frames; **Ease-Out** makes the sprite slow down in the ending frames. These setting makes the sprite move more like an object in the real world. **Ease-In** and **Ease-Out** control how a sprite moves from its start frame to its end frame, no matter how many keyframes are in between.

The **Speed** settings control how Director moves a sprite between each keyframe. The **Sharp Changes** option is the default setting. Using this option, Director calculates how to move the sprite between each pair of keyframes separately. If a sprite's keyframes are an unequal number of frames apart from each other in the Score, or different amounts of space apart from each other on the Stage, abrupt changes in speed may occur as the sprite moves between keyframe locations. Smooth out these speed changes by choosing the **Smooth Changes** option.

#### **To change the acceleration or deceleration of a sprite:**

1. Use one of the tweening methods to create a moving sprite.
2. Turn on **View > Sprite Overlay > Show Paths** to see how far the sprite moves between each frame.
3. Select the sprite and choose **Modify > Sprites > Tweening**.
4. Use the **Ease-In** and **Ease-Out** sliders to specify the percentage of the sprite's path through which the sprite should accelerate or decelerate.
5. Choose one of the following speed settings:

Sharp Changes Moves the sprite between keyframe locations without adjusting the speed. Smooth Changes Adjusts the sprite's speed gradually as it moves between keyframes.

### **Creating a Keyframe**

To make the sprite's motion more complex, we need to define an additional keyframe.

To create a new keyframe, click the frame where we want the keyframe located and choose Insert > Keyframe.

We can view and change a sprite's keyframes directly on the Stage. Choose View > Sprite Overlay > Show Paths.

Move the sprite on Stage to define a new location.

When we play the movie, the sprite moves in a curve between the three keyframe positions (Keyframe1, Keyframe2, Keyframe3 on Figure 4.4).

By default, Director automatically tweens sprite positions. To tween other sprite properties, choose Modify > Sprite.

Move keyframe2 all the way to the left to make the sprite move in a straight line between the keyframe1 and keyframe2 position.

When we play the movie, the sprite moves linearly between the keyframe positions.

We can move keyframes in the Score when a sprite is closed. This makes the keyframe's property changes occur in a different frame.

Use tweening to make sprites shrink or grow, fade in and out, or change color. All types of tweening involve the same process of defining properties in keyframes.

## **4.4 Handling Sound in AIMIT**

This section explores the procedure of using sound in developing Director movies—AIMIT. Two broad categories of sound are used with Director: internal sounds, which are imported into the cast and become integral with a file, and external sounds, which are linked to a cast while remaining external to the Director movie or cast file.

Director supports a number of sound file formats: .AIFF, .WAV, .SWA, .MIDI and QuickTime.

#### 4.4.1 Recording Sound File by using Cool Edit Pro

When planning and developing our audio requirements, we have three fundamental technical choices on a sound file's format: bit depth resolution, frequency, and whether to treat the files as internal or external.

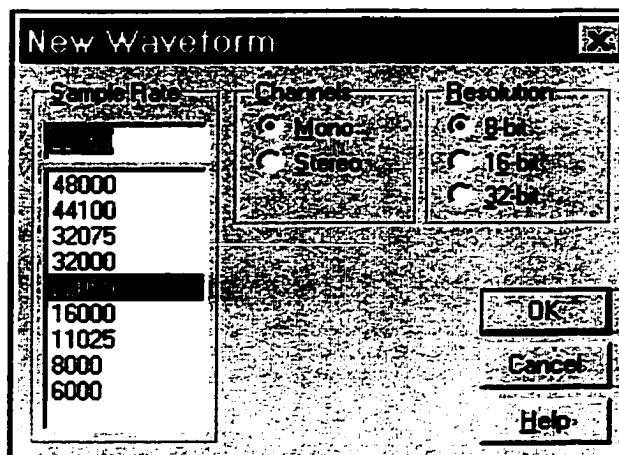


Figure 4.5: Format of Sound File

**Bit Depth Resolution:** The bit-depth resolution of a sound establishes its dynamic range, or signal-to-noise ratio measured in decibels (dB). The main reason that we use 8-bit is its compatibility, and the fact that it can be loaded into RAM faster than 16-bit and 32-bit format.

**Frequency:** Although the best quality sound is undoubtedly found with 44100 Hz (or 44.100 kHz) audio, it is difficult to use this frequency for playback in average machines because it takes too long to load into RAM. Professional developers still prefer 22050 Hz (or 22.050 kHz) because this has low noise level on built-in speakers, lots of high frequency response for clarity. It also loads fairly quickly into RAM, and is generally compatible with a wide range of computers. The average computer can process this sound and still enable the loading of additional cast members from a CD-ROM without too long of a wait. Hence, the most popular choice for Macintosh

and Windows is 22.050 kHz, for it allows good quality music and narration, similar to a strong AM radio broadcast.

Internal sounds are saved as data inside the Director movie. This makes the audio file hidden and secure when the movie is protected. External files remain outside the Director movie. In AIMIT, all sound files are imported into the Director movie, and work as internal sounds.

Sound editing application software is a necessity for all developers who need to prepare sound files for Director productions. It works as a creation tool to modify sounds and create new effects, and as a utility to ensure clean sound in the correct format. Cool Edit Pro is used as a sound editor for developing AIMIT.

In Cool Edit Pro, Play and Record, as well as other transport functions like Stop and Rewind, function as they do on “real-world” tape machines (See Figure 4.6). A Transport toolbar can be found at the bottom left of each of the main viewing windows. Click on Play to play the portion of the waveform that is currently being viewed, or the portion that is highlighted. With /Options/Loop Mode enabled, the waveform or selection will playback in a continual loop.

Click on Record to start recording at the current insertion point. Any waveform data after that will be recorded over. The button will light up bright red when recording.

Click on Pause to temporarily pause the playback or recording of audio. The button turns a bright yellow when audio is paused.

Click on Stop to end waveform playback or recording. When a waveform is being played, a vertical yellow bar shows the current playing position.

We will also find buttons for Fast Forward, Rewind, Go to Beginning/End, and Play to End and Play Looped. See Figure 4.6 for details on the Transport buttons.

#### **4.4.2 Controlling sound in the Score Window**

We control sounds in the Score in much the same way as we control sprites. We can place sounds in one of the two sound channels at the top of the Score and extend the sounds through as many frames as required. (Figure 4.7)

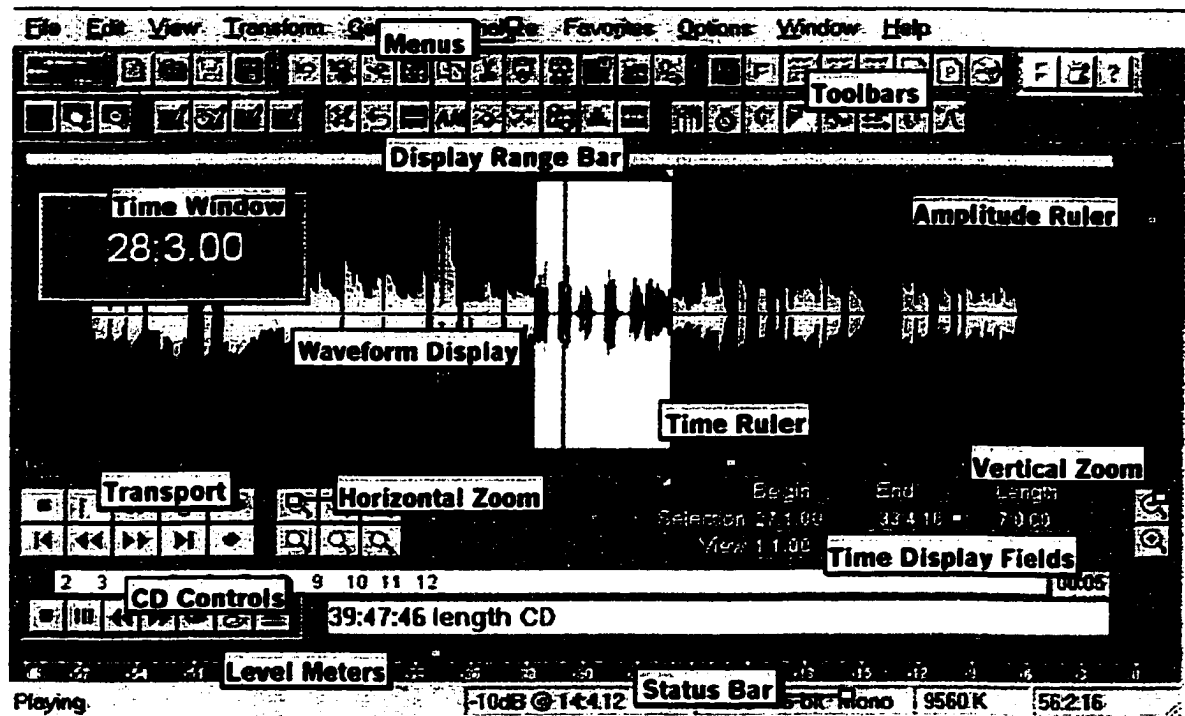


Figure 4.6: Cool Edit Pro's Editor Window

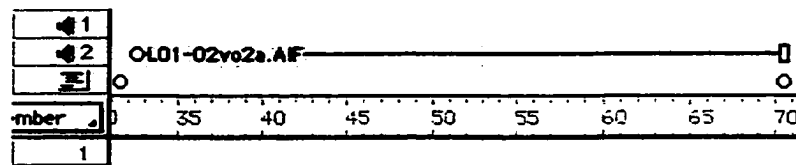


Figure 4.7: Sound Channels in Score Window

Unless we use a behavior or other Lingo scripts to override the Score's sound channels, sounds play only as long as the playback head is in the frames that contain the sound. After a sound begins playing, it plays at its own speed. Director cannot speed up or slow down sounds. If a sound is not set to loop, it stops playing at the end, even if the sprite specifies a longer duration.

In addition to the two sound channels in the Score, Director can use additional sound channels simultaneously. However, the additional channels are accessible only from Lingo, from behaviors, or by using sounds in digital videos. Available RAM and the computer's speed are the real constraints on the number of sounds Director can



use effectively.

To place a sound in the Score:

1. If the sound channels are not visible, click the expander at the right side of the Score.
2. Do either of the following: Drag a sound cast member from a Cast window to a frame in one of the sound channels. Double-click a frame in the sound channel and then choose a sound from the Frame Properties: Sound dialog box. We can also preview any sound cast member in the movie from this dialog box.
3. Extend the sound through as many frames as necessary. New sounds are assigned the same number of frames as set for sprites in the Sprite Preferences dialog box. We may need to adjust the number of frames to make the sound play completely, or change a tempo setting to make the playback head wait for the sound to finish.

Note: Sound in the last frame of a movie continues to play or loop until the next movie begins or we exit the application. This sound can be a useful transition while Director loads the next movie. We can stop the sound using the `puppetSound` command.

### **4.4.3 Synchronizing Audio in AIMIT**

Director offers a variety of ways to synchronize audio. The choice of techniques depends on our content and what we'd like to achieve.

The traditional method of synchronization is to use short sounds, and trigger them on specific frames. If we have a long narrative for a slideshow, we can then break it up into smaller files in our audio editor. Attach each phrase to a frame where the new slide starts. As the animation proceeds, it will trigger new sounds in turn.

If we are given a long sound file that cannot be cut, then we will usually have to stream it, and will usually be using SWA (Shockwave format) compression. We can embed cue points in this file to trigger movement of the score.

Perhaps the lowest-tech method of all is to simply set a low frame rate, and make sure that our animation and audio are synchronized on a range of low-end playback machines. Because Director will not exceed the frame rate we set, we can be assured that faster machines will not play back faster.

## 4.5 Shockwave Movie

AIMIT in Director movies can be saved as another kind of Macromedia format, Shockwave movie. As we mentioned in Chapter 3, Shockwave movies can be posted on Internet Web sites, and can be accessible 24 hours a day.

### 1) Creating a Web page to run a Shockwave movie: HTML code

A Director movie requires HTML code to play on a Web page. In Macromedia Dreamweaver, use the Insert > Shockwave Director command to insert the necessary code. In other HTML editors, use the code listed below.

The code for inserting a movie makes use of EMBED or OBJECT tags to play our movie. EMBED is the original tag defined by Netscape Navigator. OBJECT is the Microsoft Internet Explorer tag. All Shockwave-compatible browsers support EMBED; newer browsers support the added functionality of OBJECT. To ensure that a movie plays on as many compatible browsers as possible, use both tags.

The HTML code that follows runs a movie from both Internet Explorer and Netscape Navigator. It is intended for use with Netscape Navigator versions 2.0 or later and Internet Explorer version 3.0 or later.

For most Shockwave applications, we can enter the code as shown and simply substitute our own values for the location and size of the movie. For more sophisticated applications, we may have to use additional parameters.

```
<OBJECT CLASSID='clsid:166B1BCA-3F9C-11CF-8075-444553540000'  
CODEBASE='http://download.macromedia.com/pub/shockwave/cabs/director/sw.c  
ab\#version=7,0,0,0' WIDTH='512' HEIGHT='480' NAME='MovieName'>
```

```
<PARAM NAME='SRC' VALUE='MYMOVIE.DCR'>
<EMBED SRC='MYMOVIE.DCR' HEIGHT=480 WIDTH=512 NAME='MovieName'>
</OBJECT>
```

For WIDTH and HEIGHT, enter the Stage size of our movie in pixels. For MYMOVIE.DCR, enter the URL of our movie. For NAME, enter a name to identify the movie for browser scripting. NAME is optional, but required if we want to control our movie with a browser scripting language. In the following subsection, syntax to embed Shockwave movie is presented in detail.

## 2) Parameters for OBJECT and EMBED tags

This table includes the definitions and syntax for OBJECT and EMBED parameters.

- HTML Syntax for Shockwave Movie: CLASSID

**Definition:** The CLASSID parameter specifies the universal class identifier for the Shockwave ActiveX control. Enter it exactly as shown in the next column.

**OBJECT syntax:** CLASSID="clsid:166B1BCA-3F9C-11CF-8075-444553540000"

**EMBED syntax:** None

- HTML Syntax for Shockwave Movie: CODEBASE

**Definition:** The CODEBASE parameter of the OBJECT tag specifies where the Director Shockwave ActiveX control can be obtained if the user doesn't already have it installed in the browser, or if the user has a previous version installed. Enter the parameter exactly as shown in the next column.

**Object Syntax:** CODEBASE="http://download.macromedia.com/pub/shockwave/cabs/director/sw.cab#version=7,0,0"

**Embed Syntax:** None

- HTML Syntax for Shockwave Movie: WIDTH and HEIGHT

**Definition:** Use the WIDTH=width and HEIGHT=height parameters to specify the width and height of the image in pixels. The browser crops the image to the size we specify. Enter the exact Stage size of the movie.

**Object Syntax:** WIDTH="356" HEIGHT="128"

**Embed Syntax:** WIDTH="356" HEIGHT="128"

### 3) Shockwave browser compatibility

The Shockwave player is a system-level component that plays Director movies. Shockwave uses a plug-in to work with Netscape Navigator, and an ActiveX control to work with Microsoft Internet Explorer for Windows 95, 98, and NT. Shockwave also works with browsers that are compatible with the plug-in architecture of Netscape Navigator 3.0 or later, including America Online.

- **HTML Syntax for Shockwave Movie: NAME**

**Definition:** The NAME parameter can be used by the browser or objects in the document to find and communicate with the movie. The NAME parameter also provides a way for browsers that support FORMs to determine whether an object within a FORM block should participate in the "submit" process.

**Object Syntax:** NAME="MovieName"

**Embed Syntax:** NAME="MovieName"

- **HTML Syntax for Shockwave Movie: SRC**

**Definition:** Use the SRC parameter to specify the URL of the movie. The file's extension should be .dcr. The file name is the name of our movie. The path is the path to the movie.

**Object Syntax:** <PARAM NAME="SRC" VALUE="movie url">

**Embed Syntax:** SRC="movie url"

If Shockwave has not been installed, Internet Explorer for Windows 95, 98, and NT prompts the user for permission to download Shockwave. If the user approves, it downloads and installs Shockwave from the Macromedia Web site.

Browser	Version
Netscape Navigator for Windows or Macintosh	3.0 or later
Microsoft Internet Explorer for Windows	3.0 or later
Microsoft Internet Explorer for Macintosh	4.01 or later
America Online for Macintosh	4.0 or later
America Online for Windows	3.0 or later

Table 4.1: Browsers for Shockwave Movie

## 4.6 Java Sample Codes

Two basic programming concepts, Array and Loop are covered in AIMIT. To make examples used in AIMIT easy to understand to the students, samples encountered in real world are used to illustrate the programming concepts. For the concept of “Array”, two examples, “Temperature” and “Blue Jay Stadium” are used; for the concept of “Loop”, other examples, such as “Weight of Persons”, “Gas Station” and “Playing Cards” are used.

### 4.6.1 Temperature

The following Java sample program is used to show the students the concept of “Array.” As an example of an array, we construct an array called “dayhigh” which will store the daily high temperature for each day in a week. At the end of the week, a function called getWeekHigh returns the highest recorded temperature for the week. Sample code of the “Temperature” is as following:

```

Class Temperature {
    int index = 0;
    int dayhigh [ ] = new int [7];
    public void setDailyHigh( int temp ) {
        dayhigh [index++] = temp;
    }
    public int getWeekHigh( ) {

```

```

        int max = 0;
        for ( int i = 0; i < dayhigh.length; i ++ )
            max = max > dayhigh[ i ] ? dayhigh[ i ] : max;
        return max
    }
}

```

## 4.6.2 Two-Dimensional Array

The following is an example of an array of arrays. The method `makeArray` allocates and returns a triangular array of integer arrays. The number of integers in each component array corresponds to that array's position within the composite array. Furthermore, each component has been initialized with a sequential value, starting with 1. Sample code of the “`makeArray`” program is as following:

```

Class Array {
...
    int [ ] [ ] makeArray ( int size ) {
        int [ ] [ ] triArray = new int [size] [ ] ;
        int val=1;
        for ( int i = 0; i < triArray.length; i++ ) {
            triArray[ i ] = new int [ i+1 ];
            for ( int j=0; j < triArray[ i ].length; j++) {
                triArray[ i ][ j ] = val++;
            }
        }
        return triArray;
    }
    public static void main (String[ ] arg) {
...
        makeArray (size); // calling the above makeArray method
    }
}

```

```
...  
}
```

### 4.6.3 For-While Loop

- Java offers several different ways to repeat a block of code. One way is the for loop, which enables us to execute a set of instructions a specified number of times. In this example, we will add five people's weight together. The instructions will be executed five times, once for each person. The following is the sample code of "For Loop" program:

```
int sum = 0;  
for ( int index = 0; index < 5 ; index++)  
{  
    sum = sum + WeightOfPerson [ index ];  
}
```

- The while statement is one of the looping constructs in Java. It allows us to continue to execute a block of code while a condition is true. For example, if our program is controlling a gas pump, we want to continue to add gas to a car while the tank is not full. Sample code of the "While Loop" is as following:

```
class pump {  
    ...  
    public void fillTank( Car c) {  
        while ( !c.tankFull ) {  
            addgas ( c );  
            updateAmount ( );  
        }  
    }  
}
```

- Insertion Sort

We examine an example of Insertion Sort by using for/while loop and array. The purpose here is to demonstrate complication of loop control. The method `sortArray` sorts a sample array of integers, and shuffles the position of the integers in the array. Finally, we will have a new array in ascending order. Sample code of the "Insertion Sort" is as following:

```
static void sortArray (int [ ] array) {
    for ( int i = 1; i < array.length; i ++ ) {
        int j = i ;
        int tmp = array [ j ] ;
        while ( ( j > 0 ) && ( array [ j -1 ] > tmp ) ) {
            array [ j ] = array [ j -1 ] ;
            j -- ;
        }
        array [ j ] = tmp ;
    }
}
```



## **4.7 Final testing of AIMIT**

When our AIMIT production is complete, we still have decisions to make and work to do to deliver it to the students who will use it for Java programming. The specifications of their machines, and the delivery platform we decide to target, help to determine how we author our project, whether we are creating a Shockwave piece or a full CD-ROM title.

From a technical standpoint, delivering a Director project such as AIMIT takes practice and attention to many details. Some of the most important tips for testing and delivering AIMIT will be discussed in the following paragraphs.

### **4.7.1 Testing the Director project-AIMIT**

John Dowdell is a senior technical support engineer at Macromedia and has taken many Director authors through troubleshooting issues related to delivering a project. When we developed AIMIT, we followed his famous quote, “Test early; test often; test on all of our target machines” [31].

**Test early-**From the beginning of our development, all media elements imported into AIMIT were tested to see whether they would work well within Director or not. If any element did not perform or appear to our satisfaction within Director, we took back the sound editing, or graphics editing tools it came from for further editing.

On a large project like AIMIT, creating small test movies or modules is necessary. It is easier to work with small files. They take less time to save; they make troubleshooting easier; and they also perform better on lower-end machines. AIMIT is divided into a series of small files, and navigation among them is implemented by using Lingo. The best reason for creating small files is that we can save the files for later use. If we would like to use the same functionality in our later project, we can reuse the code.

**Test often-**Testing often is an important step throughout the development of AIMIT. That is, we did not wait until the last minute to test our production, but consistently tested our work. Testing along the way is also most important for cross-platform development. Since AIMIT is developed for Windows, Macintosh and the

Internet, we need to test our project on all of the platforms each step of the way.

Testing on All the Target Machines-Testing AIMIT on all the target machines is very important. Since we are authoring for CS170 students, we define the minimum requirements for our project based on the system of machines use in the Computer Science Department laboratories. We also tested our project on as many different configurations as our students will have. In addition, we tested it on low-end and high-end machines. The playback system requirements for AIMIT for Microsoft Windows are as follows:

- 386/33 or faster
- Windows 95, Windows 3.1, 3.11, NT 3.51, or higher
- 4 MB RAM on Windows 3.1, 8 MB or more required on Windows 95 and Windows NT.

The playback system requirements for AIMIT for Macintosh are as follows:

- 680 processors or faster
- System 7.0 or higher
- 4 MB RAM (8 MB or more recommended)

## **4.7.2 Creating the Projector of AIMIT**

Like any software application, staging the delivery of AIMIT onto the end users' machines takes planning and testing to ensure that the users experience what we intended. Creating a Director projector is not difficult. What takes more thought is how we should structure AIMIT and how the projector works with the series of Director movies in AIMIT that we have created.

Structuring Movies and Resources. When we create a Director projector for AIMIT, the Create Projector and Projector Options dialog boxes are presented with several options (See Figure 4.8). One or many movies can be added in the Create Projector dialog box.

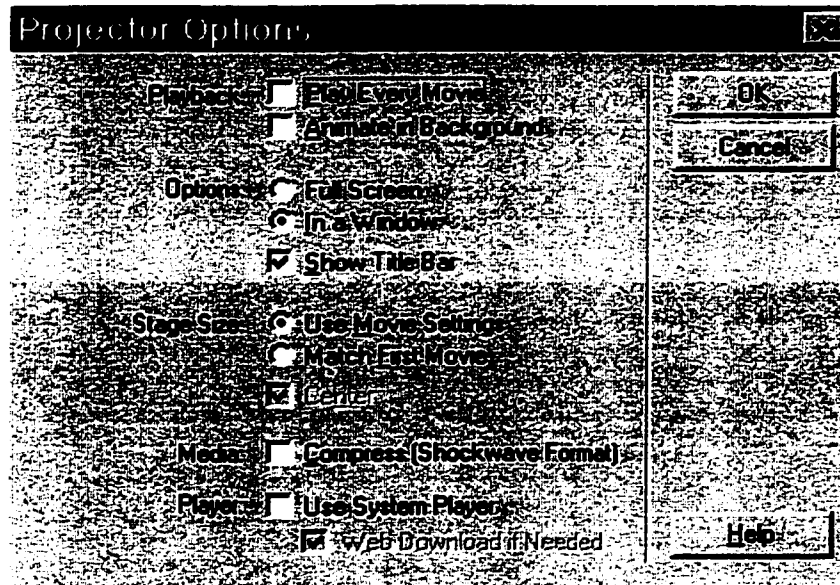


Figure 4.8: Projector Options Box

Deciding which movies to add to a projector is a crucial part of finalizing our project. Since AIMIT is a large-scale, interactive project, we add only one file to the project and store the remaining movies externally to the projector. This one file is the “Index Window” of AIMIT, and it launches the first movie in AIMIT with the go command. We then can call seamlessly between our first and subsequent Director movies with Lingo programming.

Protecting movies was the last thing we did before we distributed AIMIT. To protect our project, we select the Update Movies command from the Xtras menu. We compressed the disk size of AIMIT by two thirds when protecting it. Shockwave’s .DCR file format was used when we distributed AIMIT on the Internet, and .DXR file format is used when we distributed AIMIT on CD-ROM by using the Protected Movies command in Director. Protected movies load more quickly from the CD-ROM than Shockwave movies do because Protected movies do not need to be decompressed upon playback.

## **Chapter 5**

# **The Evaluation of the Experiment with AIMIT**

For a computer-based instructional tool, effectiveness is a major concern. This chapter will describe the evaluation of the multimedia instructional tool, AIMIT. Section 5.1 introduces the general methodology used for the evaluation. In Section 5.2, the evaluation methods used for AIMIT are discussed. The general criteria and expert review criteria applied for formative evaluation are presented in detail in Section 5.3. In Section 5.4, we discuss the methods used for summative evaluation, and examine the results from the experiments.

### **5.1 Evaluation Methodology**

Evaluation of teaching/learning software consists of two types: formative and summative. Formative methods of evaluation are used when a project's outline has been decided and work has begun on the design and development of the various parts. It can be deliberate and can consist of a series of criteria to determine whether the project works as planned. Hayden and Speedy[32] found that, although formative evaluation was a project requirement, many developers either only paid lip service or simply ran out of time before they could implement it. These authors suggest that the developers did not understand the main purpose of such evaluation and so considered it an "add-on." Alexander and Hedberg, in noting the level of academic

effort which goes into developing educational software, state,

“Given the high expectations of technology to provide more cost-effective learning and to improve the quality of the learning, together with the need to gain recognition for academics undertaking such development projects, the time has come for a re-examination of the role of valuation in the development and implementation cycles”[33].

Yet Moses and Johnson, found that “some projects were funded despite the proponent’s lack of expertise in evaluation and of knowledge about learning theories and practices”[34]. Thus, although formative evaluation should occur during the process of developing an instructional program, whether it consists entirely of software or also has other components, it appears that many multimedia instructional projects reach completion without the benefit of data that has the potential to inform and improve them. Northrup, writing about formative evaluation for multimedia, states that it is “an ongoing process conducted along every step of program development”[35] and finds that, if a first draft or version of a product is created before a formative evaluation is conducted, then major modifications will not occur even when they appear to be required. Too much money, effort and time will have gone into the product to allow a major rework to take place. To help prevent this unfortunate situation, she offers guidelines for the development team which include the need for all the people to be involved, and support for, and enforcement of, formative evaluation at all stages. She also discusses how data can be collected and used. However, Biraimah[36], Barker and King[37], Reiser and Kegelmann[38] and Henderson[39] all agree that users should help carry out the formative evaluation in a number of ways, even if their main function is seeing whether the program will actually load or not. Indeed, Reiser and Kegelmann[38] note that user evaluation of software is necessarily subjective and should be supplemented by that of subject matter experts, media specialists and administrators.

In comparison, summative evaluations can be much wider in scope. They occur when the finished product is examined and can benefit from hindsight. Thorpe, an open learning specialist, defines summative evaluation as “the collection, analysis and interpretation of information about any aspect of a program education and training,

as part of a recognized process of judging its effectiveness, its efficiency and any other outcomes it may have ”[40]. She notes that a number of characteristics belong with this definition, such as inclusiveness, the search for both intended and unintended effects, and the capability of the activity to be made public. She emphasizes that evaluation is not synonymous with assessment.

## **5.2 Evaluation Methods for AIMIT**

The purpose of the evaluation process was to establish whether the identified learning problem was overcome as a result of using the multimedia instructional tool. Questions such as, “Do they like it?” or “Do they find it useful?” while important, are inadequate. The more fundamental evaluative questions should be “What is being learned by users?”, “How will they demonstrate this learning?” and “Does the multimedia instructional tool support users in achieving the learning objectives that we want them to achieve?”

According to Beattie[41], a good evaluation needs to establish whether a learning problem is being overcome, and needs to attempt to investigate deeply the specific learning experiences users are having as a result of the teaching intervention. In other words, in the context of this thesis, it is important to discover whether or not the multimedia instructional tool had helped to solve the learning problem that we are concerned with, that is, difficulty in visualizing the dynamic process of Java programming in our introductory-level students. The evaluation of AIMIT will be based on the methodology discussed in the first section.

The methods we used for formative evaluation are general criteria and expert review criteria.

The methods we used for summative evaluation are control/experimental group test, pre- and post- test and user evaluation.

General criteria recommended by Azarmsa[42] were applied for the formative evaluation. These general criteria are described in detail in Section 5.3.1, and were followed by AIMIT developers through the design and development phase prior to the users’ evaluation. The general criteria can help the developers to ensure that each of

the objectives we have defined for the multimedia product is accompanied by more reliable and valid measures. All too often, development teams wait until a product is almost complete before “making up a test” intended to measure its outcome. That is generally too late. By using these criteria in the design stage, there is a much greater chance of designing an effective and attractive multimedia instructional program.

After the general criteria were applied, anything that was perceived as inappropriate, unclear, or incorrect, was eliminated and changed as considered necessary by the developers. This resulted in a revised version of the multimedia instructional program ready for expert review.

The reason that expert review must be involved in both of the methods is that, expert review is one of the primary evaluation strategies used in formative evaluation (How can this multimedia program be improved?) It is often a good idea to provide experts with some sort of instrument or guide to insure that they critique all of the important aspects of the multimedia program that you want reviewed. You would employ different sorts of Expert Review Checklists with different types of experts such as a content expert or a human computer interface expert[43].

Expert review criteria for this thesis are described in detail in Section 5.3.2. Two Computer Science professors, Dr. Symes and Dr. Yao, who have years of experience in teaching programming classes, used them for formative evaluation on AIMIT.

Expert review criteria was designed for this thesis, based on the “Expert Review Checklist” recommended by the Multimedia in Manufacturing Education Lab at the Georgia Institute of Technology. The “Expert Review Checklist” is a tool that will help to ensure that the experts reviewing the multimedia program focus on the variables of most interest to the multimedia program development.

The control/experimental group test was recommended by Dr. Maguire, who has years of experience in multimedia teaching, and the test was once applied by Kumar Anadan[54]. The control/experimental group test method has been adopted by lots of CAI system developments. There were sixty students taking the Computer Science Java Programming class in the fall semester of 2000, at the University of Regina. Sixteen of them voluntarily participated in the experiment outside their normal class time. They were randomly assigned into two groups: a control group

and an experimental group for both experiments. The control group students were requested to do the test without using AIMIT, while the experimental group students did the test with AIMIT. Details of this test are discussed in Section 5.4.1.

The pre- and post-tests were conducted to measure what the students had learned, and thus to measure the teaching effectiveness of AIMIT. The tests were used to determine whether AIMIT could help the students to comprehend Java programming concepts. The same tests were given to two students in the control group. Two students from the control group found some of the test questions hard to answer were willing to take the pre- and post-tests. These two students took the control group test first. Then they went through the multimedia instructional tool (AIMIT), and after this they then answered the same test kit again.

The purpose of the pre- and post-tests was to ascertain the impact of AIMIT on students' learning, in particular, their understanding of the dynamic change in Java programming. The research design involved mainly a comparison of multimedia instruction versus no multimedia instruction, and a measure of the difference. Though it might seem limited, it was considered useful to focus on the value of AIMIT for teaching Java programming concepts.

For user evaluation, a questionnaire is administered to a group of users who are given the multimedia program to work through on their own. Questionnaires are undoubtedly the single most frequently used type of evaluation instrument. Poorly designed questionnaires are often administered at the close of a course or training session as a "smile meter" or "happiness indicator." "If the only thing you find out about your interactive multimedia program with a questionnaire is whether the users liked it, you are not making good use of this strategy." As shown in the "Questionnaire", a well-designed questionnaire can provide a wealth of information.

The evaluation questionnaire is designed very carefully so that we can get the information we need without requiring the users completing it to spend too much of their time. In the questionnaire, questions were directed towards the users' satisfaction and their perceived usefulness of the AIMIT in helping them to learn Java programming concepts presented. Details are presented in Section 5.4.3.



## **5.3 General Criteria and Expert Review Criteria**

Users are no longer satisfied merely because a program will load and run without error. Technical excellence is widespread today, and anything less must be rejected as unacceptable. It is our responsibility to demand excellence in the project we build for our users. In this section, the general criteria and expert review criteria used for formative evaluation are presented in detail.

As you might expect, there are many different ideas about how to categorize general criteria. Church and Bender[53] suggest that the following be evaluated: (1) instructional content, (2) utilization of computer capabilities, (3) goals and objectives, (4) software presentation variables (pacing, ability levels, and the like), (5) production information, and (6) documentation. We prefer to base evaluation on seven similar criteria recommended by Reza Azarmsa, that attention should be given to (1) instructional objectives, (2) accuracy and fairness of content, (3) Approach to content, (4) Instructions for operating software, (5) screen formatting, (6) user response and (7) technical quality.

### **5.3.1 General Criteria**

#### **Instructional Objectives**

A good multimedia instructional program will include clearly defined instructional objectives, and suggested activities for using the package to achieve them. The objectives should be part of the written documentation, usually in the teacher's manual.

- Does the program documentation have defined objectives and goals that clearly state expected learning?
- Are the objectives clear to the user and perceived as relevant to instructional goals?
- Are the objectives important to the Java programming teaching?
- Does the program focus on a very specific and limited topic or objective?

- Is the program appropriately divided into smaller segments?
- Do the objectives make it clear whether the program is designed for tutoring, demonstration, testing and placement, or for some other purpose?
- Does the program achieve its stated objectives?

### **Accuracy and Fairness of Content**

It is still necessary to review carefully the factual content of each program for accuracy. Also the instructional content of the program should be checked for relevance to Java programming teaching, and to determine whether it is appropriate at the grade level(s) for which it is being considered.

- Does the content fit well into CS170– Java programming teaching?
- Does the program have instructional values and/or other valid objectives for the users who will be using it?
- Is all information factually correct?
- Is all information current and free of slang phrases or poorly selected examples?
- Are the graphs, charts, and other visual aids included in the program well designed and technically accurate?
- Is the content appropriate to the interests and the abilities of users ?
- Is the presentation logical and well organized?
- Is the content free of negative comments regarding ethnic background?

### **Approach to Content**

Creative programs present the usual material in unusual ways and often involve the user in experiences not otherwise available. Many concepts can be explored in more exciting ways than having users respond with “right” or “wrong” answers.

Whenever possible the programs should be open ended, allowing the user to control the instructional content and to select the method of presentation. Such creative use of the computer is one of the most important factors to consider in evaluating courseware.

- Does the program present information in a way not easily duplicated with textbooks, workbooks, or other traditional classroom materials?
- Does the interaction between the user and the program significantly involve the user in the learning process?

### **Instructions for Operating Software**

Good instructions make a program easy to use and therefore attractive to the user.

- Are the instructions clear, complete, concise, and well formatted on the screen?
- Does the user have the option of bypassing the instructions?
- Are there instructions for ending the program (e.g. Exit) if the user wants to stop?
- In the event that the program “crashes,” or ends unexpectedly, are there instructions for continuing rather than forcing the user to return to the beginning?
- Are there instructions to inform the user how to control the speed and sequence of paging?
- Is the reading level of the instructions appropriate to the intended users?

### **Screen Formatting**

Users should be able to control the screen by advancing or returning to any desired page at will. Special attention must be paid to avoid the frustration of having a page advance before the user has finished reading it, and to avoid the frustration of waiting for a program to continue.

- Is each separate screen neat and uncluttered?
- Is the amount of information displayed at one time appropriate rather than confusing?
- Is there a limited number of choices at one time?
- Can the user access specific screens, advancing or returning to earlier material as desired?
- Does the user control the timing of screen advancement?
- Are spelling, punctuation, and grammar correct in each screen display?
- Are maps, graphs, and other illustrations clear and simple to interpret?
- Are all graphics displayed effectively?
- Is colour used effectively on systems that offer this option?

### **Users Response**

Well designed courseware is “user friendly,” making it easy for the user to work through the program without being inhibited by the mechanics of computer operation.

- Are data entry routines consistent and easy for the user to use?
- Does the program use common conventions and symbols for data entry?
- Is there an indicator to show where the responses are to go?
- Is the user given an opportunity to correct errors?

### **Technical Quality**

The wide range of technical qualities available today represents one of the most innovative and creative aspects of using the computer for instruction. Yet it is essential that we make the realistic assessment of these technical qualities an important part of our evaluation.

**GRAPHICS** should use illustrations as valuable additions to the program, helping to achieve the objectives by demonstrating, explaining, or otherwise clarifying the material.

- Are the graphics appropriate for intended users?
- Are the graphics embedded in content?
- Do the graphics add interest?
- Are the graphics clear?
- Do the graphics enhance program quality?

**COLOUR** can be a powerful motivational tool and can also be used to explain and clarify information. Colour cannot be counted automatically as a plus in a program, and should be critically evaluated for its effect on the total presentation.

- Is colour used effectively?
- Is colour a motivating factor in this program?
- Does colour make the text difficult to read?
- Is the program more interesting to use because of colour?

**SOUND** can be either essential or supplemental in a program.

- Can the sound be turned off completely by the user if necessary?
- Is the sound used effectively?
- Is the sound used for instruction?
- Is the sound used only for reward?

**USER CONTROL** can be effective when properly used. Otherwise it can lead to user frustration by presenting material too quickly or, for the able user, far too slowly.

- Are there opportunities for the user to choose among content topics?
- Can the user review instruction?
- Can the user exit the program at any time?
- Can the user alter the rate of presentation (e.g. text rolling)?

### **5.3.2 Expert Review Criteria**

Expert review is one of the primary evaluation strategies used in formative evaluation (e.g. how can this multimedia program be improved?). It is often a good idea to provide experts with some sort of guide to ensure that they critique all of the important aspects of the multimedia program that you want reviewed.

#### **EXPERT REVIEW CRITERIA**

The “Expert Review Criteria” is a tool that will help assure that the experts reviewing AIMIT focus on the variables of most interest to us.

#### **SECTION 1 - INSTRUCTIONAL DESIGN REVIEW**

- Does AIMIT provide learners with a clear knowledge of the program objectives?
- Are the instructional interactions in AIMIT appropriate for the objectives?
- Is the instructional design of AIMIT based on sound learning theory and principles?
- Is the feedback in AIMIT clear?
- Is the pace of AIMIT appropriate?
- Is the difficulty level of AIMIT appropriate?

## **SECTION 2 - COSMETIC DESIGN REVIEW**

- Does the screen design of AIMIT follow sound principles?
- Is colour appropriately used in AIMIT?
- Are the screen displays easy to understand?

## **SECTION 3 - PROGRAM FUNCTIONALITY REVIEW**

- Does AIMIT operate flawlessly?

Following the above criteria, the expert review was conducted via e-mail and personal interview with Dr. Symes and Dr. Yao from the Computer Science Department. Dr. Symes went through AIMIT carefully and noted his comments, as well as asking some questions relating to each section of AIMIT. Based on his years of teaching programming classes in Computer Science, he suggested what kind of examples in real life we should use for interpreting the programming concepts. Dr. Yao also gave an excellent suggestion on that “Insertion Sort” algorithm sample. He suggested to use Poker cards to show the student the dynamic process of Insertion Sort. The evaluation afterwards by the students who used AIMIT proved that, their comments and suggestions are insightful and invaluable to this research work.

## **5.4 Summative Evaluation**

### **5.4.1 Control/Experimental Group Test**

#### **Introduction**

As we mentioned earlier in Section 5.2, sixteen of class CS170 voluntarily participated in the experiment outside their normal class time. Each student signed a letter of consent before the experiment. Appendix A of this thesis shows a photocopy of the letter of consent, a copy of questionnaire (Part I), a copy of evaluation form (Part

II), and also a photocopy of the University of Regina ethics committee approval for conducting this experiment.

The questionnaire (Part I) used in the AIMIT Control/Experimental Group Test contains multiple choice questions and written questions involving the writing of programs. The questionnaire was designed to have two parts. Part I (A) consists of questions involving “For/While Loops” (first experiment); Part I (B) consists of questions involving “Arrays” (second experiment). Part II concerns the evaluation of AIMIT. For Part I, the students were required to answer five questions in the Java programming language for both experiments. Students in the Experimental Group, who used AIMIT were requested to answer Part II of the questionnaire.

### Procedure

Sixteen students were randomly assigned into two groups: a control group and an experimental group for both experiments. One group (Group I), with eight students, was requested to answer the questionnaire, Part I, without using AIMIT, but rather using their textbook and notes. Another group (Group II), with eight students, was requested to answer the same questionnaire, but answer Part I plus Part II using AIMIT. The first experiment on Part I (A) “For/While Loops” was organized as shown in Table 5.1:

Phase	Group I	Group II
I	Solving 5 questions of Part I (A) of the questionnaire without using AIMIT	Solving 5 questions of Part I (A) of the questionnaire using AIMIT
II	–	Completing Part II of the questionnair

Table 5.1: Organization of First Experiment

For the second experiment on Part I (B) “Arrays”, one group (the control group) with eight students, was assigned to answer the questionnaire without using AIMIT,



while another group (the experimental group), with eight students, was assigned to answer the questionnaire using AIMIT. Table 5.2 illustrates how the second experiment was organized.

In Phase I (Table 5.1, 5.2) of both experiments, the students were requested to answer all the questions of Part I of the questionnaire (shown in Appendix A). In Phase II, only students using AIMIT were requested to answer Part II of the questionnaire. The students decided which group they would like to join before the start of the experiment. The time limit was a maximum of half an hour for writing the test.

Phase	Group I	Group II
I	Solving 5 questions of Part I (B) of the questionnaire without using AIMIT	Solving 5 questions of Part I (B) of the questionnaire using AIMIT
II	–	Completing Part II of the questionnaire

Table 5.2: Organization of Second Experiment

### First Experiment (“For/While Loop”) Results

All students finished the Part I (A) (Loops) of the questionnaire within 30 minutes. Part I (A) was designed to have two multiple choice questions on the “For Loop,” one multiple choice question on the “Do-While Loop,” one short answer question on the “While Loop” and one question involving writing a Java program using a “Do-While Loop.”

The results of all the five questions of Control Group involving eight students without using AIMIT are shown in Table 5.3.

Question	S1	S2	S3	S4	S5	S6	S7	S8	Average	Percentage
Q1	0	5	5	5	5	5	5	5	4.375	87.5
Q2	5	0	5	5	5	5	5	5	4.375	87.5
Q3	5	3	5	4	1	5	5	5	4.125	82.5
Q4	5	5	5	0	5	5	5	5	4.375	87.5
Q5	5	0	5	0	5	5	5	0	3.125	62.5
Overall	4	2.6	5	2.8	4.2	5	5	4	4.075	81.5

Table 5.3: Results of the Control Group on the “For/While Loop”

- Average = Sum/8;
- Overall Average = Sum/5;
- Average Percentage = (Average /5) \* 100.

“S1, S2, ..., S8” refers to the students and “Q1, Q2, Q3, Q4, Q5” refers to the questions in Part I (A) of the questionnaire (Appendix A). Each of the five questions has 5 marks, and the total mark is 25. The column “Average” refers to the average mark obtained for each question among 8 students without using AIMIT, and its equivalent value in percentage is shown in the “Percentage” column.

The results of Part I (A) of the questionnaire with the experimental group (Group II) involving eight students using AIMIT are shown in Table 5.4.

Question	S1	S2	S3	S4	S5	S6	S7	S8	Average	Percentage
Q1	5	5	5	5	5	5	5	5	5	100
Q2	5	5	5	5	5	5	5	5	5	100
Q3	5	5	5	5	5	5	5	5	5	100
Q4	5	5	5	5	5	5	5	5	5	100
Q5	5	5	5	5	5	5	5	5	5	100
Overall	5	5	5	5	5	5	5	5	5	100

Table 5.4: Results of the Experimental Group on the “For/While Loop”

### **First Experiment (“For/While Loops”): Part I (A) Result Summary**

From Tables 5.3 and 5.4, it is clear that students in control group had considerable difficulty with the programming concepts. The comparative results from Table 5.4 indicate that the overall percentage of the experimental group (Group II) is much higher than the control group (Group I), which tells us that students using AIMIT were able to perform better than students not using AIMIT.

### **Second Experiment (“Array”) Results**

All students finished Part I (B) (“Array”) of the questionnaire within 30 minutes. Part I (B) was designed to have two multiple choice questions on the “Array Index”, one multiple choice question on the “two dimensional array”, one short answer question on “multiple dimensional arrays,” and one question involving the writing of a Java program on using a “multiple dimensional array.”

#### **Part I (B) (Questionnaire)**

The results of all five questions of the control group (Group I) involving eight students without using AIMIT are shown in Table 5.5.

Question	S1	S2	S3	S4	S5	S6	S7	S8	Average	Percentage
Q1	5	3	5	5	5	3	5	5	4.5	90
Q2	0	5	0	5	5	5	5	0	3.125	62.5
Q3	0	5	5	0	5	5	5	5	3.75	75
Q4	5	5	5	3	0	1	5	5	3.625	72.5
Q5	3	3	5	0	0	3	2	5	2.625	52.5
Overall	2.6	4.2	4	2.6	3	3.4	4.4	4	3.525	70.5

Table 5.5: Results of the Control Group on the “Array”

“S1, S2, ..., S8” refers to the students and “Q1, Q2, Q3, Q4, Q5” refers to the questions in Part I (A) of the questionnaire (Appendix A). Each of the five questions has 5 marks, and the total mark is 25. The column “Average” refers to the average

mark obtained for each question among eight students without using AIMIT, and its equivalent value in percentage is shown in the “Percentage” column.

The results of Part I (B) of the questionnaire with the experimental group (Group II) involving eight students using AIMIT are shown in Table 5.6

Question	S1	S2	S3	S4	S5	S6	S7	S8	Average	Percentage
Q1	5	5	5	5	5	5	5	5	5	100
Q2	0	5	5	5	5	5	5	5	4.375	87.5
Q3	5	5	5	5	5	5	5	5	5	100
Q4	5	5	5	5	5	5	5	5	5	100
Q5	5	4	5	4.5	5	4.5	4.5	5	4.6785	93.75
Overall	5	4.8	5	4.9	5	4.9	4.9	5	4.8125	96.25

Table 5.6: Results of the Experimental Group on “Array”

### **Second Experiment (“Array”): Part I (B) Result Summary**

From the above Tables 5.5, 5.6, it is clear that students in Control Group had considerable difficulty with the programming concepts. The comparative results from Table 5.6 indicate that the overall percentage for the experimental group (Group II) is much higher than the control group (Group I), which tells us that students using AIMIT were able to perform better than students not using AIMIT.

#### **Pre- and post-tests results**

Section 5.2 has already presented an introduction to pre- and post-tests method.

The results (see Figure 5.1) from the pre- and post-tests were particularly pleasing. The difference between the pre- and post-test results shows that students who worked through AIMIT improved their test results rather significantly.

### **5.4.2 Evaluation Questionnaire**

Evaluating multimedia instructional tool is a time-consuming but extremely important task if we are going to secure the best quality for our users. Software must

Student	Test	Part I (A) "For/While Loops"					Part I (B) "Arrays"					Total
		Q1	Q2	Q3	Q4	Q5	Q1	Q2	Q3	Q4	Q5	
S4	Pre-	5	5	4	0	0	5	5	0	3	0	27
S4	Post-	5	5	5	5	5	5	5	5	5	5	50
<b>Difference</b>		<b>0</b>	<b>0</b>	<b>+1</b>	<b>+5</b>	<b>+5</b>	<b>0</b>	<b>0</b>	<b>+5</b>	<b>+2</b>	<b>+5</b>	<b>+23</b>
S5	Pre-	5	5	1	5	5	5	5	5	0	0	36
S5	Post-	5	5	1	5	5	5	5	5	5	5	46
<b>Difference</b>		<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>+5</b>	<b>+5</b>	<b>+10</b>

Figure 5.1: Results of Pre- and Post- Tests

be evaluated to ensure quality and compatibility.

The experimental group students who used AIMIT before they wrote the test were requested to give their opinion in Part II, after they finished Part I of the experiment. They were asked the following questions and opinions the multimedia tool kit:

1. The multimedia tool kit is easy to use.
2. The multimedia tool kit provides effective and useful messages in the process of teaching programming concepts.
3. The multimedia tool kit helps in understanding the concepts.
4. The multimedia tool kit helps in learning the concepts quickly.
5. I will recommend this tool kit to other students.
6. I recommend the development of such a tool kit (Multimedia tool kit) in the future.
7. Do you think it will be helpful to have a system like this running on the Computer Science Department's Web site to allow you access at any time, 24 hours a day?
8. Comments or suggestions for improvement of this multimedia tool kit.
9. What do you like about this multimedia tool kit?
10. What do you not like about this multimedia tool kit? What are your suggestions for improving it?
11. If you have used another kind of multimedia system/tool, are there any comparisons you would like to make?

Students provide their answers for questions 1 to 7 by circling a number from 1 to 5 in the following order:

- 1=strongly disagree
- 2=disagree
- 3=no opinion
- 4=agree
- 5=strongly agree

The questionnaire was designed in this manner following the recommendations of The Multimedia in Manufacturing Education Lab at the Georgia Institute of Technology. The students' response to the first seven questions is summarized in Table 5.7.

### Part II Evaluation Questionnaire Result Summary

Question	S1	S2	S3	S4	S5	S6	S7	S8	Average	Percentage
Q1	4	5	5	4	5	4	4	5	4.5	90
Q2	4	5	5	4	4	4	5	4	4.375	87.5
Q3	5	4	5	4	5	5	5	5	4.75	95
Q4	4	4	5	3	4	5	4	5	4.25	85
Q5	4	3	5	4	5	4	5	5	4.375	87.5
Q6	5	5	5	5	5	5	5	5	5	100
Q7	5	5	5	5	4	5	5	4	4.75	95
overall	4.43	4.43	5	4.143	4.571	4.571	4.71	4.71	4.571	91

Table 5.7: Experimental Group Evaluation Questionnaire Results

All students in the experimental group expressed positive (agree or strongly agree) comments with two exceptions (See Table 5.7). The two exceptions were a “no opinion” for questions 4 and 5 (Q4, Q5).

Students answered questions 8 to 11 in the following manner. Suggestions for question eight indicate that, a “Replay” button needs to be placed, not only on some long-time running slides, but also on each slide. When asked about what they liked about AIMIT (question nine), six of the eight students mentioned that they liked the

visual effects and animation which made it easy to use and easy to understand. One student indicated that she liked the “Insertion Sort” example using cards and other visuals, since they made learning much easier. While responding to question ten, two students mentioned that more Java programming concepts should be covered. At present, we only have two basic concepts, “Loops” and “Arrays”. Seven of the eight students gave no comments regarding question eleven. This indicates that a multimedia instructional tool, such as AIMIT, is not popular on the campus for novice programming class teaching. Only one student mentioned that she had used one multimedia instructional program for Physics 30, but said that AIMIT has more animation and short slides, which made it easier to use and easy to understand.

### **Experimental Result Summary**

Based on the above results from two experiments we are able to find that:

1. Students using AIMIT were able to perform better than the students not using AIMIT.
2. AIMIT provides assistance for teaching Java programming concepts to students.
3. Students have indicated that AIMIT is easy to use and also indicates that it provides effective and useful messages.
4. Students like the visual effects and graphics section of the animation in AIMIT.
5. All students are in favour of the development of such a multimedia tool kit in the future.
6. Finally, most of the students are in favour of having such a multimedia tool available 24 hours a day on the Internet.

The users concerns regarding the improvement to AIMIT will be discussed in detail in Section 6.2.

## Chapter 6

# Conclusions and Future Work

### 6.1 Conclusions

In the previous five chapters, a basic introduction to multimedia enhanced instruction tools was presented. The historical review of computer-assisted instruction (CAI) was conducted. The importance of interactivity and multimedia for introductory computer science programming courses was presented. AIMIT (An Interactive, Multimedia Instructional Tool) was developed based on the concept of using interactivity in multimedia technology. With the constantly increasing popularity for using the WWW (World Wide Web), there is a wide scope for delivery of programming courses over the WWW. AIMIT, which can be embedded on Internet, is an example of an application for teaching programming concepts on the WWW.

Two experiments among the CS170 students were performed at the University of Regina. These experiments were performed to assess the value of AIMIT in aiding students' ability to learn basic Computer Science programming concepts. Experimental results indicate that introductory-level students using AIMIT were able to perform better than the students not using it, validating the design of AIMIT and this approach in general as an aid in teaching basic programming concepts.



## 6.2 Contributions

At the present time, AIMIT assists users in learning “Array” and “Loop” concepts in Java programming language. Two experiments and students’ evaluation on AIMIT give us clear and positive messages:

- AIMIT allows students to learn programming concepts interactively in a non-linear way, and multimedia techniques used in AIMIT clearly illustrate the dynamic process in programming.
- Using Macromedia Director as the development tool is not only cost effective, but also allows students to access AIMIT through the Internet 24 hours a day.
- AIMIT can be run on different operating systems, can be burned on CD-ROM, and required little learning to use it.
- By using Lingo in Director, AIMIT is designed in a modular style that allows for future extension.

## 6.3 Future Work

Several extensions can be made to AIMIT. One desirable extension is to make AIMIT support other programming languages, such as C and C++. As to the students’ negative comments in Chapter 5, I find the primary concern is covering more programming concepts in AIMIT. This problem can be improved by setting up a multimedia project team to have more people working on it. Another direction for AIMIT would be providing multimedia-supported exercises, tests and feedback on students’ performance and progress in an interactive and adaptive manner. The purpose of the further research is to investigate the effect of an overall multimedia course in introductory computer science programming education. However, this would require extensive Lingo knowledge and significant effort, and this is the ultimate goal of multimedia supported instructional tool such as AIMIT.

## Bibliography

- [1] Kalmbach, J. A. Just in time for the 21st century: Multimedia in the classroom. *Tech Trends*, 39(6), pp29-32, 1994.
- [2] *The Business Week Guide to Multimedia Presentation* by Robert L. Lindstrom, Osborne McGraw Hill, San Francisco, 1994.
- [3] Reynolds, A., & Anderson, R. H. Interactive multimedia. In *Selecting and Developing Media for Instruction* [3rd ed.], pp. 195-202. New York: Van Nostrand Reinhold, 1992.
- [4] Paivio, A. *Imagery and verbal processes*. New York: Holt, Rinehart & Winston, 1971.
- [5] Clark, J. M., & Paivio, A. Dual coding theory and education. *Educational Psychology Review*, 3, pp149-210, 1991.
- [6] Mayer, R. E., & Anderson, R. B. Animations need narrations: An experimental test of a dual-coding hypothesis. *Journal of Educational Psychology*, 83, pp484-490, 1991.
- [7] Paivio, A., & Csapo, K. Picture superiority in free recall: Imagery or dual coding? *Cognitive Psychology*, 5, pp176-206, 1973.
- [8] Soloway, E. Reading and writing in the 21st century. With Mark Guzdial and Kenneth E. Hay. *EDUCOM Review*, 28(1), pp26-29, 1993.
- [9] Livergood, N. D. A study of the effectiveness of a multimedia intelligent tutoring system. *Journal of Educational Technology Systems*, 22, pp337-344, 1994.

- [10] A New Paradigm for University Teaching & Learning By Howard Kaplan, Sequence: Volume 32, Number 1. Release Date: January/February 1997.
- [11] Strauss, R. The development tool fandango: Deciding the authoring system versus programming language question. CDROM Professional, 8(2), pp47-55, 1995.
- [12] Barron, A. E., & Orwig, G. W. Multimedia technologies for training. Englewood, CO: Libraries Unlimited, 1995.
- [13] Bangert-Drowns, R. L.; Kulik, J. A.; and Kulik, C. C. "Effectiveness of Computer-Based Education in Secondary Schools." Journal of Computer-Based Instruction 12/3: pp59-68, 1985.
- [14] Batey, A. Building a Case for Computers in Elementary Classrooms: A Summary of What the Researchers and the Practition-ers Are Saying. Paper presented at the Second Leadership in Computer Education Seminar, Seattle, WA, December 1986.
- [15] Grimes, D. M. Computers for Learning: The Uses of Computer Assisted Instruction (CAI) in California Public Schools. Sacramento, CA: California State Department of Education, 1977.
- [16] Kinzer, C.K., Sherwood, R.D., and Bransford, J.D. Computer strategies for education. Columbus, OH: Merrill Publishing Co., pp26, 1986.
- [17] Pagliaro, L.A. The history and development of CAI: 1926-1981, an overview. The Alberta Journal of Educational Research, 29(1), pp75-84, 1983.
- [18] Woolley, D.R. PLATO: The emergence of on-line community. Computer-Mediated Communication Magazine, 1(3), 5, 1994.
- [19] Alessi, S.M., Trollip, S.R. Computer-based instruction: methods and development. Englewood Cliffs, NJ: Prentice-Hall, 1985.
- [20] Chambers, Jack A. & Sprecher, Jerry W. Computer-Assisted Instruction: Its Uses in the Classroom. Englewood Cliffs, NJ: Prentice-Hall, 1983.

- [21] Alessi, Stephen M. & Trollip, Stanley R. *Computer-Based Instruction: Methods and Development*. Englewood Cliffs, NY: Prentice-Hall, 1991.
- [22] Kinzer, C.K., Sherwood, R.D., and Bransford, J.D. *Computer strategies for education*. Columbus, OH: Merrill Publishing Co, 1986.
- [23] Saettler, Paul. *The Evolution of American Educational Technology*. Englewood, CO: Libraries Unlimited, 1990.
- [24] Niemiec, R. P., & Walberg, H. J. From teaching machines to microcomputers: Some milestones in the history of computer-based instruction. *Journal of Research on Computing in Education*, 21, pp272-273, 1989.
- [25] Johnson W.L. and Soloway E. *PROUST Byte Vol. 10 No.4*, 1985.
- [26] Anderson J.R. and Reiser B.J. *The LISP Tutor Byte Vol. 10 No. 4*, 1985.
- [27] Anderson J.R., Boyle C.F., and Yost G. *The Geometry Tutor Proceedings of IJCAI-85*, 1985.
- [28] Anderson J.R., Boyle C.F. and Reiser B.J. *Intelligent Tutoring Systems Science* 228, pp456-462, 1985.
- [29] Macromedia Support Center: *Managing multimedia production*, <http://www.macromedia.com/support/director/how/expert/manage/managemm01.html>.
- [30] Lee Allis, *Inside Director 6.0 with Lingo*, pp17, New Riders Publishing, 1997.
- [31] Lee Allis, *Inside Director 6.0 with Lingo*, pp70, New Riders Publishing, 1997.
- [32] Hayden, M. & Speedy, G. *Evaluation of the 1993 National Teaching Development Grants*. Project commissioned by the Committee for the Advancement of University Teaching. Lismore, Australia: Southern Cross University, 1995.
- [33] Alexander, S. & Hedberg, J. *Evaluating technology-based learning: Which model?* In K. Beattie, C. McNaught & S. Wills (Eds.) *Interactive Multimedia in*

- Education: Designing for Change in Teaching and Learning. Holland: Elsevier Science B.V, 1994.
- [34] Moses, I. & Johnson, R. Review of the Committee for the Advancement of University Teaching. URL <http://uniserve.edu.au/caut/report/finrep.html>, 1995.
- [35] Northrup, P.T. Concurrent formative evaluation: Guidelines and implications for multimedia designers. *Educational Technology* 35 Nov/Dec, pp24-31, 1995.
- [36] Biraimah, K. The non-neutrality of educational computer software. *Computers & Education* 20(4), pp283-290, 1993.
- [37] Barker, P. & King, T. Evaluating interactive multimedia courseware – a methodology. *Computers in Education* 21 (4), pp307-319, 1993.
- [38] Reiser, R.A. & Kegelmann, H.W. Evaluating instructional software: A review and critique of current methods. *Educational Technology, Research & Development* 42(3), pp63-69, 1994.
- [39] Henderson, L. Instructional design of interactive multimedia: A cultural critique. *Educational Technology, Research & Development* 44(4), pp85-104, 1996.
- [40] Thorpe, M. *Evaluating open & distance learning*. Harlow, Essex: Longman Open Learning, 1988.
- [41] Beattie, K. How to avoid inadequate evaluation of software for learning. In K. Beattie et al (ed.), *Interactive Multimedia in University Education: Designing for Change in Teaching and Learning*. Elsevier Science B.V., 1994.
- [42] Reza Azarmsa. *Educational Computing, Principles and Applications*, California State University, Bakersfield. Educational Technology Publications, Englewood Cliffs, NJ 07632, pp46-55, 1991.
- [43] MULTIMEDIA DEVELOPMENT TOOLS,  
[http://mime1.marc.gatech.edu/MM\\_Tools/ERC.html](http://mime1.marc.gatech.edu/MM_Tools/ERC.html).

- [44] Paivio, A. *Mental representations: A dual-coding approach*. New York: Oxford University Press, 1986.
- [45] Paivio, A. Dual coding theory: Retrospect and current status. *Canadian Journal of Psychology*, 45, pp255-287, 1991.
- [46] Shortcliffe E.H. *Computer-based Medical Consultations: MYCIN* American Elsevier, 1976.
- [47] Fletcher, J.D. Effectiveness and cost of interactive videodisc instruction in defense training and education. ERIC microfiche ED326194. July, 1990.
- [48] O'Neil, H.F., Slawson, D.A., & Baker, E.L. Design of a domain-independent problem-solving instructional strategy for intelligent computer-assisted instruction, Hillsdale, NJ:Erlbaum, pp69-104, 1991.
- [49] Stevens, A.L., Roberts, B. & Stead, L. The use of a sophisticated interface in computer-assisted instruction. *IEEE Computer Graphics and Applications*. 3(2), pp25-31, 1983.
- [50] Johnson, W.L., & Soloway, E. PROUST: Knowledge-based program understanding (Report No.285), New Haven, CT:Yale University, Computer Science Department, 1987.
- [51] Anderson, J.R., & Reiser, B.J. The LISP tutor, *Byte*, 10(4), pp159-175, 1985.
- [52] Clancey, W.J. Knowledge-base tutoring, the GUIDON program, Cambridge, MA:MIT Press, 1987.
- [53] Church, G., & Bender, M. *Teaching with computers: A curriculum for special educators*. Boston:Colledge-Hill Press, 1989.
- [54] S.K. Anandan, Animation Tool Kit for Computer Science Education on the Internet. Computer Science Department, University of Regina, 1999.
- [55] WebCT Support Center, <http://www.webct.com>.

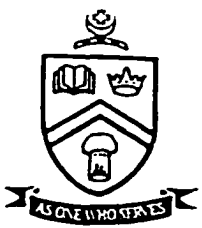
## **Appendix A**

This appendix section includes:

1. **Ethics Committee Approval for Experiments**
2. **Letter of Consent for Student Volunteers**
3. **Questionnaires**

## **A.1 Ethics Committee Approval**





UNIVERSITY OF REGINA  
OFFICE OF RESEARCH SERVICES

---

DATE: September 5, 2000

TO: Chi Song  
Computer Science


FROM: K. McNaughton  
Research Ethics Board

Re: **An Interactive Multimedia Instructional Tool For Computer Science Students**

---

Please be advised that the committee has considered this proposal and has agreed that it is:

1. Acceptable as submitted.  
(Note: Only those applications designated in this way have ethical approval for the research on which they are based to proceed.)
2. Acceptable subject to the following changes and precautions (see attached):  
**Note:** These changes must be resubmitted to the Committee and deemed acceptable by it prior to the initiation of the research. Once the changes are regarded as acceptable a new approval form will be sent out indicating it is acceptable as submitted.  
**Please address the concerns raised by the reviewer(s) by means of a supplementary memo.**
3. Unacceptable to the Committee as submitted. Please contact the Chair for advice on whether or how the project proposal might be revised to become acceptable (4775.)

  
K. McNaughton

cc: Dr. L. Symes  
Computer Science

GM/ab/ethics2.dot

## **A.2 Letter of Consent**



**UNIVERSITY OF REGINA**  
**DEPARTMENT OF COMPUTER SCIENCE**

---

**Letter of Consent**

---

**Survey on Multimedia Tool Kit for Computer Science Students**

---

1. Thank you for your time. Please note:
  - ◆ Participating in this project is not part of the requirements for this course (CS170).
  - ◆ We would like to ask you to test a multimedia tool kit, take a sample test on programming concepts (loops and arrays) and then provide some feedback regarding the multimedia tool kit.
  - ◆ Your participation is **COMPLETELY VOLUNTARY** and you may withdraw from the study at any time without **PENALTY**.
  
2. Volunteers will be divided into two groups, one group(A) will employ an multimedia tool kit to assist programmers in learning fundamental programming concepts, the other group (B) will not. When the tool kit is fully developed, we hope that it will be a useful tool for assisting students of introductory level programming courses.
  
3. Participation will include:
  - For group A:**
    - ◆ Approximately 1 to 1.5 hours in a computer lab outside of your normal class time.
    - ◆ Solving 10 problems with the use of multimedia tool kit.
    - ◆ Filling out a questionnaire that provides us with feedback for improving the multimedia tool kit.
  - For group B:**
    - ◆ Approximately an hour in a computer lab. This is outside of your normal class time.
    - ◆ Solving 10 problems without the use of multimedia tool kit.
  
4. Goal of study – to evaluate the effectiveness of the multimedia tool kit and obtain feedback for improving the tool kit.

5. No personal information will be asked in this study. Only your input to the problems (loops and arrays) and your opinion on the effectiveness of the software is needed. Your comments on using the software will not be released to the public and your name will not be mentioned under any circumstances. After the completion of the study, only survey results with no identification will be published in the Master of Science thesis.
6. This project was approved by the Research Ethics Board, University of Regina. If research subjects have any questions or concerns about their rights or treatment as subjects, they may contact the Chair of the Research Ethics Board at 585-4775 or by email: [research.ethics@uregina.ca](mailto:research.ethics@uregina.ca),
7. Contact persons:  
If you have any questions regarding the survey please contact us.

Chi Song  
Office: CL135.1  
Phone: 585-4836  
Email: [chisong@cs.uregina.ca](mailto:chisong@cs.uregina.ca)  
[songlch@uregina.ca](mailto:songlch@uregina.ca)

My thesis advisor :  
Dr. Larry Symes  
Office: AH505.4.1  
Phone: 585-5318

8. Please sign this form indicating your agreement to participate in this study.
9. I acknowledge the receipt of this consent letter and am willing to participate in this survey.

Name: \_\_\_\_\_

Date: \_\_\_\_\_

### **A.3 Questionnaires**

## Multimedia tool kit for Computer Science Education Project

### Questionnaire

#### Note:

1. Please answer **ALL** questions. Your information will be used for scientific research and this information will remain confidential.
2. Please provide an honest answer.
3. Please circle appropriate answer(s) for the questions with possible answers provided.
4. This questionnaire has **Part I and II**. **Part I** is about programming concepts (Loops). Part II is about the tool kit and please answer the questions in Part II **only** if you use the multimedia tool kit.

### Part I(A)

1.

```
public class Loop_01
{
    /* main */

    public static void main(String[ ] args)
    {
        int sum = 0;

        for (int i = 0 ; i < 10 ; i++)
            sum = sum + 1;

        System.out.println("The value of the sum is " + sum );

    } // end of method main

} // end of class Loop_01
```

What will be the output of the above program?

- a. sum = 1
  - b. sum = 0
  - c. sum = 10
  - d. sum = 11
  - e. None of the above answers
2. How many times the above program(Question 1.) will be executed if the for loop is modified as :  
For( int i = 10; i >0 ; i-- ) ?
- a. 0
  - b. infinite
  - c. 1
  - d. 10
  - e. None of the above answers

3.

```
public class Loop_03
{
    /* main */
    public static void main(String[ ] args)
    {

        System.out.println("Table of Squares " );
        int i = 0 ;
        While ( i < 6 )
        {
            System.out.print("the square of " + i) ;
            System.out.println(" is " + (i * i));
            i++;
        }
        System.out.println("**end of table**");
    } // end of method main

} // end of class Loop_03
```

What will be the output of the above program?

4. How does do while loop execute? Please select the right answer.
- a. do while loop executes the body of the loop first and then evaluates the condition
  - b. do while loop does not execute the condition at all
  - c. do while loop evaluates the condition first and then executes body of the loop
  - d. do while loop does not execute body of the loop
  - e. None of the above

5. Modify the program in Question 3.. to use **do while** loop ( You only need to modify the **loop** part ).



## Multimedia tool kit for Computer Science Education Project

### Questionnaire

#### Note:

1. Please answer **ALL** questions. Your information will be used for scientific research and this information will remain confidential.
2. Please provide an honest answer.
3. Please circle appropriate answer(s) for the questions with possible answers provided.
4. This questionnaire has **Part I and II**. **Part I** is about programming concepts (Array). **Part II** is about the tool kit and please answer the questions in **Part II only** if you use the multimedia tool kit.

### Part I (B)

1.

```
import java.lancs.* ;

public class Question2
{
    /*main */
    public static void main(String[] args) throws Exception
    {
        int [ ] i = { 1, 2, 3, 4, 5, 6 } ;

        char [ ] c = { 'A', 'B', 'C', 'D', 'E' };

        String [ ] names = { "John", "Mary", "Tom" };

        System.out.println("The second number is " + i [1] );
        System.out.println("The third character is " + c [2] );
        for (int k = 0; k < names.length; k++)
            System.out.println(names[k] );
    }
}
```

What will be the output of the above program?

2.

```
import java.lancc.*;
class Average {
    public static void main (String [ ] args) throws IOException {
        int [ ] i= { 1, 2, 3, 4 } ;
        int sum 0 ;
        for ( int k = 0; k < 4 ; k++ )
            sum = sum + i[k] ;
        float average = sum/4 ;
        System.out.println (" The averagge is " + average );
    }
}
```

What will be the output of the above program ?

- a. 1
- b. 1.5
- c. 2
- d. 2.5
- e. None of the above answers

3. The declaration of 2 dimensional array with 2 rows and 3 columns would be as below.

```
Private int [ ] [ ] board = { {CROSS, BLANK, BLANK},
                              {BLANK, CROSS, CROSS}} ;
```

Please select the correct answer:

- a. True
- b. False

4. `int [][] triangleArray = { {1}, {1, 1}, {1, 2, 1}, {1, 3, 3, 1}, {1, 4, 6, 4, 1} }`  
`/* print the array */`  
`for ( int i =0; i < triangleArray.length; i++ ) {`  
`for (int j = 0; j < triangleArray[i].length; j++)`  
`System.out.print (triangleArray[i][j] + " " );`  
`System.out.println( );`  
`}`

what is the output of the above program?

5. Rewrite the program in Question 4, and try to get the following output:

```
1 4 6 4 1
1 3 3 1
1 2 1
1 1
1
```

## Part II

**Note:**

1. Please provide an honest answer.
2. Please circle the appropriate answer(s) for the questions with possible answers provided.

### Opinion about the multimedia tool kit:

Please rate the following statements by circling the number that represents the response closest to your opinion:

1=strongly disagree

2=disagree

3=no opinion

4=agree

5=strongly agree

1. The multimedia tool kit is easy to use.

1            2            3            4            5

2. The multimedia tool kit provides effective and useful messages in the process of teaching programming concepts.

1            2            3            4            5

3. The multimedia tool kit helps in understanding the concepts.

1            2            3            4            5

4. The multimedia tool kit helps in learning the concepts quickly.

1            2            3            4            5

5. I will recommend this tool kit to other students.

1            2            3            4            5

6. I recommend development of such tool kit (Multimedia tool kit) in the future.

1            2            3            4            5

7. Do you think it will be helpful to have a system like this running on Computer Science Web site to allow you to access at anytime 24 hours a day?

1            2            3            4            5

