MULTIMEDIA PACKAGE: LEARNING HOW TO PROGRAM IN C, PLUS INTERACTIVE TUTORIAL

By

Norsuhana Binti Azman

A thesis submitted to the graduate faculty in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE
IN
COMPUTER SCIENCE

Department of Software Engineering
Faculty of Computer Science and Information Technology
University Malaya

Tables of Contents

List of Figures

List of Tables

Acknowledgement

With the guidance and help from several key people, I was able to complete this project within the time limit. First of all, I would like to express my greatest appreciation to Pn.Azwina for all the precious time spared in order to discuss and exchange creative ideas for the benefit of this project. Many thanks go to Mr. Chiew Thiam Kian who lent his time in giving fair evaluation on the project. And not forgetting, friends who have shared many useful opinions concerning the project. I am also grateful to my family who have given me words of courage and comfort throughout the completion process of this project.

Abstract

Multimedia has grown into a technology which is frequently adopted in most of computer-based educational titles lately. Not only it is implemented in education but it can be seen in business, entertainment and administrative environments. Multimedia has added dimension to the way humans exchange ideas and communicate with machines through a multisensory system. It has provided us with tools and techniques that make communication and comprehension more effective.

Due to the fact that multimedia plays a vital part in the way people communicates with machines and its environment, this project has taken the initiative of applying the multimedia concept in its learning package. More over, programming languages such as C is considered an important programmers basic knowledge before moving on to other programming concepts, that makes it suitable to be conveyed as multimedia based learning tool.

In ensuring a quality software produced, stages of iterative development process has been adopted by implementing the Rapid Application Development methodology (RAD). RAD opts for iterative development process that will help to detect early faults which in turn helps to shorten the development process.

The project focuses on issues such as research methods, reviews made on existing software and surveys on authoring tools. An extensive literature review on subjects that assist in providing information in this project is discussed. Then, moving on to the system design, system implementation, the writing of programs, the testing stage and ending with the delivery process.

CHAPTER 1

INTRODUCTION

Chapter 1 Introduction

1.1 Project Overview

As the term multimedia evolved in our daily lives and the wide spread of its implementation around the globe, its virtue has been known to the information society. This project concentrates on developing a learning package software that incorporates multimedia elements such as text, graphics, images, animation and speech and sound. An effective learning approach could be implemented by making use of the advantages from a mutisensory system. Thus, learning to program in C would become an easy task and a meaningful experience through a multimedia package. This project aims to educate users on how to program in C, the basic of programming language.

The package includes a new concept in educational multimedia which is interactive multimedia. In this software, interactive tutorials and program execution simulation are features of interactive multimedia that could enhance the learning process. As the importance of human-computer interaction (HCI) takes place in the computing environment now, interactivity has become the main aspect of every Computer-Based Education (CBE) software in the market. Due to this fact, the design of interactive contents within a software as well as a user-friendly interface has been taken into consideration.

In any development, information plays an important role in producing a successful project. In this project, various methods are used to obtain relevant information. Methods include reviews, interviews, and internet surfing. Other than that, in ensuring a quality software, the development process of the software are based on good software engineering practices. In this project the Rapid Application Development (RAD) methodology is applied.

1.2 Project Objectives

Below are the objectives that this project aims to achieve:

- Building an educational software that represents information through most of multimedia elements
- Building a multimedia educational software that offers interactivity to the users.
- Introducing C programming language to a wide range of users through effective learning concepts
- Encouraging in-depth understanding through interactive tutorials and line-by-line program execution simulation
- To assist in better navigation by providing an interesting and friendly graphical interfaces
- Providing adequate information and further examples on relevant topics to enhance understanding and knowledge acquiring
- At the end of the learning session, learners will be able to structure and program independently
- To build an educational multimedia software that could convey information in which users could absorb the contents in the most effective way

1.3 Project Scope

The project scope defines the system boundary, explaining what will be included in the system and what will not be included:

- This learning package covers the basic concepts of C programming language which caters for beginners and intermediate users. Also for educators who are teaching the basic programming of C to users that are still new to the programming environment. The topics included are the C fundamentals, program control statements, arrays and strings, pointers and functions.

- This learning package is a local multimedia application software, where the services are provided by local stand-alone systems which is the CD-ROM itself. During run-time it does not make use of any other extra resources than those present on the local system. Local system refers to the personal computer which runs and resources from the CD-ROM itself.

- This package consists of six modules which are:

1. Introduction Module
2. Lessons Module
3. Additional Lessons Module
4. Sample Codes Module
5. Tutorials Module
6. Answers Modules

1.4 Project Schedule

The project has been given a duration of 204 days of completion time. The development process begins on the 11/3/2002 and ended with the completion of the multimedia presentation which is on the 26/9/2002.

Month	March				April				May				June				July				August				Sept			
Week	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Feasibility Study																												
System definition																												
System Analysis																												
System Design																												
System implementation																												
System testing																												
System delivery																												

Table 1.1: Project Schedule

1.5 Report Layout

This report is organized in eight parts to help readers comprehend easily. Here are the brief contents of every chapter contained in this report.

Chapter 1 Introduction

This chapter gives an overview of the whole project. Readers would be able to understand the objectives of the project development and the expected outcome. Besides that, the project scope will give readers the outline of the project's boundary. The project schedule will show readers the duration in which every development activity takes place.

Chapter 2 Literature Review

This chapter explains on the importance of C, the multimedia and interactive multimedia definition. Readers could read on the advantages of Multimedia Educational Software (MES) and Human-Computer Interaction (HCI), the frequently discussed issue presently. Other than that, reviews made on five existing multimedia educational softwares will give readers opportunity to evaluate the systems. Besides, it helps readers to get a picture of how the existing software differs from this software. In this chapter research plans and methods are explained and survey on existing authoring tools are discussed.

Chapter 3 System and Requirements Analysis

In this chapter reviews had been made on two software life cycle methodologies, the Waterfall Model and the Rapid Application Development Methodology (RAD). Here, justifications are made on the preferences for RAD in developing this software. Besides that, this chapter explains the process of capturing requirements from various resources. At the end f the chapter, concentration is given on the functional and non-functional requirements of this project.

Chapter 4 System Design

In this chapter, the focus is on the system's architectural issues such as the conceptual design and technical design. At later part of this chapter, discussions will be made on the system's modeling. In system modeling, different design methods are implemented in order to convey the system's design and properties. This include navigation flow chart, decomposition diagrams and data flow diagram. At the end of this chapter, several system's user interfaces presented.

Chapter 5 System Implementation

This chapter explains the initial point where the real part of developing and building the system occurs. In this phase the structures that have been designed within the System Design phase are given digital form. Concentration is given on the multimedia development softwares that are implemented to develop this multimedia presentation.

Chapter 6 Writing and Testing Programs

This chapter contains explanations on the programming language that is used in developing this multimedia presentation. Marcromedia Director's Lingo scripting language is a very powerful multimedia programming tool. It introduces pre-programmed behaviors, which are a set of combinations of an event and action that will take place if the event is triggered. Other than that, testing these programs is essential in identifying bugs that can effect the quality of the presentation. In this chapter the debugging techniques are discussed.

Chapter 7 Testing and Delivering the System

After unit testing have been done, the next step is to test the overall system. Testing the system is not a technical process. It involves users of the system and the requirements fulfillments. There are four main testing that were undertaken. They are the function testing, performance testing, acceptance testing and delivery testing. After all the testing procedures were done analysis were made to determine the flaws that exist in the system

and this will eventually produces an outline of the system strengths and limitations that the system owns. From here the developer should know the extent of the system objectives that are able to achieved.

1.6 Summary

In this introduction chapter, readers get a general view of the project. The project touches on the evolvement of multimedia in the computing world nowadays and the important part it plays in conveying educational information in an effective manner. The chapter informs readers about the new concept that is applied in this project, which is the interactive multimedia and how it can benefit computer-based education materials. Other than that, the chapter tells of methods used in the project, which include reviews, interviews and internet surfing. The software development process applied in this project is the Rapid Application Development methodology.

Through this chapter readers are informed of the project objectives and scope. The target audience for this learning package are beginners and intermediate users as the software contents concentrates on basic C programming. The project schedule shows the duration accommodated to complete every phase right up to the end of the system's completion.

Layouts of every chapter can be observed from this chapter. Readers will get the basic idea of the contents of every chapter in this report. All information on every chapter can be thoroughly comprehended by accessing these chapters. Explanation on the development at every stage of the software life-cycle are clearly laid out in their respective chapters.

CHAPTER 2

LITERATURE REVIEW

Chapter 2 Literature Review

2.1 C Programming Language

C was invented and first implemented by Dennis Ritchie on a DEC PDP-11 that used the UNIX operating system. C is the result of a development process that started with an older language called BCPL.

Why Learn C?

C is often called a middle-level computer language. This does not mean that C is less powerful, harder to use, or less developed than a high-level language such as BASIC or PASCAL. Rather, C is thought of as a middle-level language because it combines the best elements of high-level languages with the control and flexibility of assembly language. C is a very portable programming language. The use of compiler directives to the pre-processors make it possible to produce a single version of a program which can be compiled on several different types of computer. The function libraries are standard for all version of C so they can be used on all systems. Another reason is this programming language is the fundamental to other languages such as C++, JAVA and UNIX programming. It is the favourite programming language among professional programmers.

2.2 What is Multimedia?

In the computer-based information area, multimedia is defined as "the field concerned with the computer-controlled integration of text, graphics, still and moving images, animation, sound, and any other medium where every type of information can be represented, stored, transmitted, and processed digitally. Thus, multimedia applications may stimulate several human senses; that is multisensory.

The four characteristics of multimedia systems

1. **Multimedia systems must be computer controlled** means that at least the presentation of the information, also called playout, to the end-user is controlled by a computer.

2. **They are integrated,** that is they use a minimal number of different devices, an example is the use of a single computer screen to display all types of visual information.

3. **The information they handle must be represented digitally,** it implies that a single format – that of a succession of binary digits is used to represent not only the usual computer data, but also for all other types of media. Another consequence of digital representation is that the various media, though integrated, may be treated independently.

4. **The interface to the final user may permit interactivity** which allows the end-user to control how and when information elements are presented. Somehow, this is not a mandatory feature for systems to be termed multimedia. It is a frequent feature, but not all multimedia systems are interactive.

2.3 Interactive Multimedia

Interactivity is amount of control the user has over the presentation of information. Interactive multimedia involves people communicating with people, aided by machines. It refers to multimedia that allows for user control. It is more than just people interacting with machines. Certainly interactive multimedia requires a digital intermediate form to be established for content derived from any medium, be it text, data, audio, graphics, video or animation. Interactive multimedia products mainly appear as CD-ROM discs that are

played on personal computers with colour screens, sound cards, hard disk and CD-ROM drives.

Most importantly, interactive multimedia products define both the content and the context in which the user can find, manipulate and interpret informative, educational and entertaining material.

2.3.1 Human-Computer Interaction (HCI)

As interactivity becomes an important component of multimedia, studies on human-machine interaction and interfaces have delivered the term HCI. Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use with the study of major phenomena surrounding them. HCI concentrates on several disciplines, each with different emphases: computer science (application design and engineering of human interfaces), psychology (the application of theories of cognitive processes and the empirical analysis of user behavior), sociology and anthropology (interactions between technology, work and organization) and industrial design (interactive products).

Research in HCI has been spectacularly successful, and has fundamentally changed computing. Just one example is the graphical interface used by Microsoft Windows 95. Another example is that most software written today employs user interface toolkits and interface builders. Even the spectacular growth of World-Wide Web is a direct result of HCI research; applying hypertext technology to browsers allows one to traverse a link across the world with a click of the mouse.

2.4 Multisensory Systems for Knowledge Acquisition

As humans, any type of audio-visual contact that we have in our communication process adds dimensions to our message comprehension.
Taken from Fetterman and Gupta, 1993 :

> *" Humans retain only 20% of what they see, from 20% to 30% of what they hear. But they also retain from 40% to 50% of what they see and hear simultaneously, and in the order of 80% of what they see, hear, and do at the same time. Here, the definition of "retain" is not given, and we shall assume that this refers to long-term memory".*

Practicing while seeing or listening is very essential. Thus, interactive multimedia systems combine the advantage of multisensory channels and active participation. The many advantages that underlie within an interactive multimedia system include:

- Increase in productivity levels
- Higher rates of learning and knowledge acquisition
- Control of the learning pace
- Reaching a wider audience and markets

Due to the fact that an interactive multimedia system plays a vital role in enhancing knowledge acquisition, the implementation of interactive components within this software such as the interactive tutorials and the program execution simulations are part of fulfilling this important criteria.

2.5 Applying Multimedia to Education

As mentioned earlier, this package focuses on educating users on how to program in C. Thus, this software is not only known as a multimedia software but it can be categorize as a multimedia educational software. It can serve the needs of university students as well as a teaching tool used by lecturers or educators while introducing a new programming concept.

2.5.1 What is Multimedia Educational Software (MES)

MES refers to software programs specifically developed for educational purposes, which use a variety of different media (text, graphics, animation, video, sound, or speech) to present information.

This software is usually called "interactive" implying that the content or direction of the program changes in response to some input from the student. The levels of interactivity range from "object interactivity", where the system responds to a mouse click by playing a sound or displaying an image, to a form of interactivity which allows the user to play a role in a simulated environment, and where input from the user provides the opportunity to experience changes in the environment that result from the manipulated of conditions.

2.5.2 The Educational Advantages of MES

Below are the significant advantages of multimedia educational software:

- MES provides representations in multiple modalities (e.g 3D, auditory, graphic, text). Comprehension and retention are remarkably increased when more senses are used in acquiring information.
- MES can drill students on basic concepts until they reach mastery.

- MES can facilitates collaborative activity amongst students.
- MES can help students see interconnections among concepts through the use of the hypertext mode.
- MES can repeat the lesson as many times as users like.
- Attention can be firmly held, thanks to sophisticated interaction devices and game-like scenarios.
- MES has the flexibility to meet the individual needs and abilities of each user, allowing branching options and different learning paths.
- The user interacts directly with the software (exploring, answering questions, performing activities and inserting data) and receives immediate feedback.
- MES provides a "friendly" learning environment, removing barriers such as fear of failure, embarrassment and peer pressure.

2.6 Existing Software Review

Review on existing multimedia softwares is essential in helping software developers to evaluate different components and contents of various software available in the current market. The criterias that are taken into consideration include interface design, contents and structure as well as availability or non-availability of navigational tools. Reviews have been done on three educational software:

1. **Inside the Unified Modeling Language (UML)**
   By Rational Software Corporation

Figure 2.1 The main menu of Inside the UML learning software.

- The target audience for this software are students and professionals, including software developers.
- It has full characteristics of a multimedia system. It incorporates texts, video, narration, animation and graphics.
- The interface designs are attractive and consistent. The introduction to UML includes live video appearance by important figures.
- Contents and information are well organized and carefully structured to avoid information overloading.
- Navigational features include "Continue" and "Exit". Back and previous functions are not needed as all the topics and sub-topics could be accessed without these functions.
- The learning package conveyed step by step example of constructing the UML diagrams clearly, making the learning experience a very effective process.
- In this software, hyperlinks to web sites with relevant information and free downloads are available.
- Narration is done throughout the package. A clever way of avoiding boredom is to change the narration voice from a man's to a woman's half way through the package. This way users will stay focussed and still have the interest to proceed.

2. Java Multimedia Cyber Classroom

By Harvey M. Deitel and Associates

Figure 2.2 The main menu of Java How to Program learning package.

This is a learning package which accompanies a reference book titled Java How to Program, Third edition written by Deitel & Deitel. These are the results:

- This package does not incorporate voice/narration, animation and video. Still, it have the criteria of an interactive multimedia with live-code execution, hypertext representations and live hyperlinks to web sites.
- The cyber classroom package only acts as a partial learning package as there is no explanations on programming concepts. Users rely on the lessons contained in the reference book. The main page is equipped with hypertext links, which gives access to installation instructions of the CD-ROM and the Java 2 SDK compiler, complete code examples and web site's links for Java Demos, Java Resources and many other Java components. The resources and information provided is sufficient and extensive. Thus, making it very helpful for Java users who are seeking for Java resources.
- One obvious advantage is that it offers a live World Wide Web link to the most up-to- date code examples with just a click of the mouse on the hypertext web site address.
- The package comes with a very detailed installation instructions and system requirements that cover different platforms.
- The structure of components on the main page is quite cluttered but somehow files of executable programs are very well organized in folders according to its chapters and lesson topics. Associations can be easily made among the programs from the reference book and the ones in the software, as filenames are very meaningful, for example Welcome1.java, Circle.java.
- As for navigational tools, the software takes advantage from the Microsoft Internet Explorer browser. Yet, there is one tool that is included, which is return to the top page.

3. The Logic Tutor

By David Abraham, Liz Crawford, Leanna Lesta, Agathe Merceron, Kalina Yacef, University of Sydney.

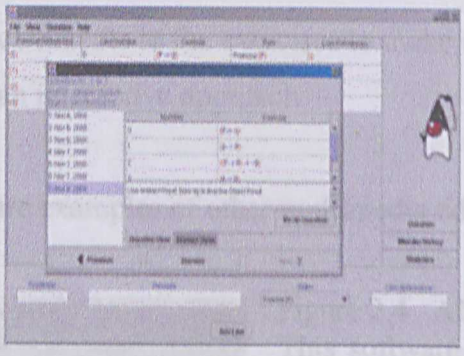


Figure 2.3 An interface taken from The Logic Tutor.

The Logic Tutor is a tool to support computer science students in their learning of logic and more specifically in their learning of formal proofs.

- This learning package does not incorporate voice, video, sound or animations.
- The interface design is very design and plain. The structure of components within the interface is very cluttered.
- The software has a modular design for easy modifications especially when there are addition of logics or formulas.
- The learning package is based on practices and tutorials concept. Students are able to enter formulas and the system will checked the validity of the operation performed against a database of mistake patterns and return feedbacks.
- The software apply interactivity in the sense of giving quality feedback to users informing of them of the mistake done, what type of mistake and advises for correction.
- The software provides Help function that gives explanation on the basic functionalities of the Logic Tutor.

- The user history module records every exercise the user attempts to allow continuation on unfinished exercises. This is considered a good software design for flexibility in accessing previous exercises.
- The software allows tutors and lecturers to create files of exercises and associates comments to the exercises to make it more interactive. This could be described as an interactive approach.

Below are examples of other multimedia educational softwares:

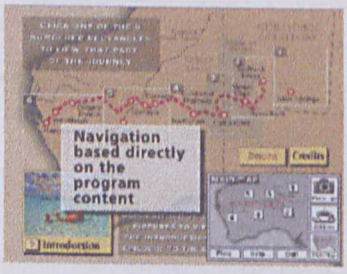


Figure 2.4 **Alice to Ocean**
This software allows the user to travel with an adventurer across the Australian outback. The journey can be followed linearly or users have the choice to go to different parts of the trip by placing the mouse on the chosen destination. This CD provides a visually stunning experience that blends narrative and interactivity.

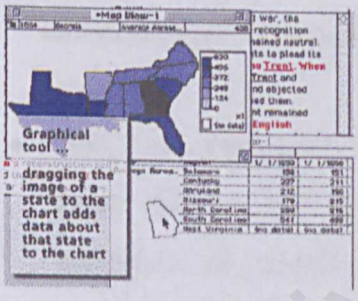


Figure 2.5 **Point of View**
This is a history tool that allows for students and teachers to add to existing databases of information. There are powerful timeline and data manipulation tools included.

Figure 2.6 **How Computers Work**
This is a multimedia computer delivered instruction program. It contains lessons and activities for learning about computing and its history.

2.7 Research Plans and Methods

In order to obtain sufficient information and relevant materials on the subject, various techniques had been consulted. They include:

- **Brainstorming and discussion with lecturers**

  Brainstorming helps to produce creative ideas on how the system should be developed. The discussion offers an opportunity to clarify and validate problems and solution. It assured the whole project is on the right track.

- **Internet resources**

  It offers the fastest and easiest way of accessing information. With the help of Google, the frequent used search engine in this project, most of the information concerning the latest technology to be implemented in this project could be obtained. Relevant texts concerning the subject had been abstracted from electronic journals and electronic encyclopaedias through the internet.

- **FSKTM's Document Room**

  From the faculty's document room, past years thesis had been reviewed and considered as guidelines. It gives an overview of how the thesis should look like and what it should contain.

- **University Malaya's Main Library**

  This facility provides books and publications that are relevant to the project. It offers a wide range of reliable and useful information. Besides that, the reviewed multimedia educational software was also obtained from here.

- **Reviews on existing educational software**

  Through the reviews made on the existing multimedia software, a brief idea of the software contents and the interface layout had been benefited. Evaluation of each software helps to significantly highlight the important components that were included and its variety interface designs.

2.8 Authoring Tools

Authoring tools have the advantages of collecting all the separate multimedia elements and integrating them with the power of interaction so that the application becomes a powerful communicator. In the past, incorporating such features into an application required tremendous programming codes but nowadays with the existence of many authoring tools developers save effort and time. These are the features that could be expected from an authoring tool :

- Provide graphics and text function, which allow developers to design, create and edit more text and graphics
- Posses importing capabilities to allow developers import media elements that have been created externally.
- Provide media integration function that allows developers to sequence, link, synchronize and even script the application in an interactive manner.

Types of authoring tools

There are many different types of authoring tools in the market, these are the popular categories:

- Slide-based
- Object-oriented or Icon-based
- Frame-based
- Script-based

2.9 Survey on Multimedia Authoring Tools

1. Astound Inc.'s Astound

- Astound is a slide-based authoring tool which have the capabilities to incorporate multimedia and interactivity. The metaphor that this authoring software follow is that of a traditional slide show presentation sequence, where the screen appears as a slide and is navigated sequentially.
- This software is multimedia capable because they are able to import a variety of multimedia file types, from graphics to video to sound and animation.
- It also has OLE (Object Linking and Embedding) capabilities that allow developers embed or link media elements in each slide.
- Besides that, it provides a user-friendly toolbar and a slide sorter that enables developers to rearrange slides as they want them to appear. The software comes with an assortment of templates yet give developers the choice of creating their own master slides and own customized templates.
- Astound offer an intuitive sequencing method called a timeline. A timeline is a horizontal bar that keeps track of the objects' order of appearances on each slide.

2. Macromedia Authorware

- Authorware is an object-oriented or icon-based tool. This object-based packages utilise the traditional method of the flow-chart metaphor.
- The media elements are represented by different icons, to create an application simply drag a particular media icon and drop it onto the flowline.
- But since these packages are primarily used to importing media elements, their graphics editing and creation capabilities are somewhat limited.
- Authorware provides over 1,000 Smart Clips which includes buttons, sliders, and bullets that can be imported directly into the presentation.

- Authorware also allows Director movies file imports into the presentation.
- Icon-based authoring tools offer a lot more interactive capabilities. It provide users with hypermedia and hypertext functionalities.
- Users are able to create links, such as hotspots or hot objects, not just between text items, but between the other multimedia elements such as graphics, sounds, animations, and digital videos. All these without any scripting.

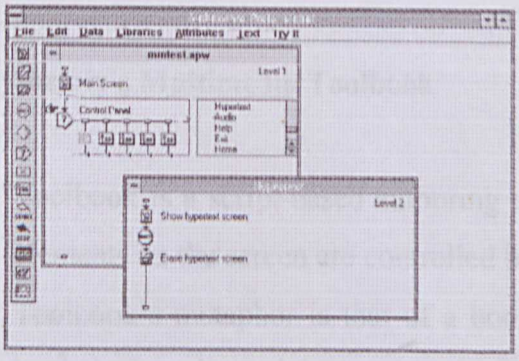


Figure 2.7 Authorware's Flowline

3. Macromedia Director

- The Macromedia Director is a frame-based authoring tool, where the use of cells, frames and channels are the basis on which a multimedia application is built.
- Director is currently the standard and developers prime authoring tool for authoring CD titles, interactive games, kiosks, and product demonstrations.
- The metaphor that Director adopts is that of theatre and movies. This package combines a powerful animation engine with interactive capabilities.
- Director's files are saved as "movies", the screen is called the Stage, all the multimedia elements that are imported or created are called Casts and the control of the movie is done on the Score.

- It is similar to that of the presentation packages' timelines that it runs from left to right. Director's timeline is based on frames or cell.
- Due to its function as an authoring tool that has an animation engine, Director's graphics utilities are powerful. It comes with a Paint window that allows creations of dazzling designs and graphics.
- If developers require to add interactivity to the animations that have been created, they would have to utilise Director's scripting language called Lingo.

4. Asymetrix's Multimedia Toolbook

- Toolbook is a script-based authoring tool, where the actions of the multimedia elements on the screen are controlled by scripts and are executed line by line.
- Toolbook's metaphor is that of a book. The completed application is called a book and is divided into pages (or the screen). These pages can have fields (text), buttons and graphics and can be shared among pages throughout the book.
- It does allow importation of media elements saved in popular file formats.
- It also provides a toolbox to create graphical elements like shapes, fields and buttons for the pages.
- Toolbook has limited animation capabilities. Therefore, any animation, video or sound files have to be created in third-party software like Adobe Premier or Creative Labs' SoundBlaster.
- Once they have been created or imported, they have to be scripted in order to be functional. The scripting language used in Toolbook is called OpenScript. These scripts handle the actions to be performed by buttons and other multimedia elements and control the branching structure of the application.

2.10 Summary

At the beginning of this chapter, readers are told of a brief explanation on the C programming language and the reasons why this multimedia software offers a learning package on C programming. C is a powerful programming language that is used frequently by programmers and should be a fundamental knowledge for beginners.

The multimedia term is defined as the field concerned with the computer-controlled integration of text, graphics, still and moving images, animation, and any other medium where every type of information can be represented, stored, transmitted, and processed digitally. There are four characteristics of multimedia; they must be computer controlled, they are integrated, all information is represented in a digital form and lastly they may offer interactivity. Interactivity leads to interactive multimedia which involves people communicating with people, aided by machines. It refers to multimedia that allows for user control. Human-Computer Interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use with the study of major phenomena surrounding them.

Multimedia Educational Software refers to software programs specifically developed for educational purposes, which use a variety of different media (text, graphics, animation, video, sound, or speech) to present information. MES provides a friendly environment, removing barriers such as fear of failure, embarrassment and peer pressure.

Existing software review is done on three software; Inside the Unified Modeling Language, Java How to Program and The Logic Tutor. Inside the UML is a very interesting and complete software. Java How to Program does not incorporate narration and animation in the software as it only holds live program execution codes, Java 2 SDK compiler and Java resources material. The Logic Tutor is a software with databases that

CHAPTER 3

SYSTEM AND REQUIREMENTS ANALYSIS

Chapter 3 System and Requirements Analysis

3.1 Overview

In any software development project, there is a sequence of steps taken to accomplish a set of tasks. This set of ordered tasks is called a process. The software development process is sometimes called the software life cycle. There are various types of life cycle models. In this chapter two different models will be discussed. The general steps in a software development process usually involves the following stages:

- Requirements analysis and definition
- System design
- Program design
- Writing the programs (program implementation)
- Unit testing
- Integration testing
- System testing
- System delivery
- Maintenance

There are several reasons for modeling a process:

- The model reflects the goal of the development, such as building high-quality software and finding faults early in development.
- Description of development process forms a common understanding of the activities, resources, and constraints involved in software development.
- It helps the discovery of inconsistencies, redundancies, and omissions in the process and in its constituent parts.

In this chapter, discussions on software process models and techniques of eliciting requirements from the right resources are done. There will be a list of the system

requirements definitions as a result of the requirements eliciting process done earlier. Included in this chapter are the functional and non-functional requirements of the system.

3.2 Waterfall Model

This model is one of the first models to be proposed, illustrated in Figure 3.1, where the stages are depicted as cascading from one another. As the figure implies, one development stage should be completed before the next begins. The waterfall model presents a very high-level view of what goes on during development, and it suggests to developers the sequence of events they should expect to encounter.

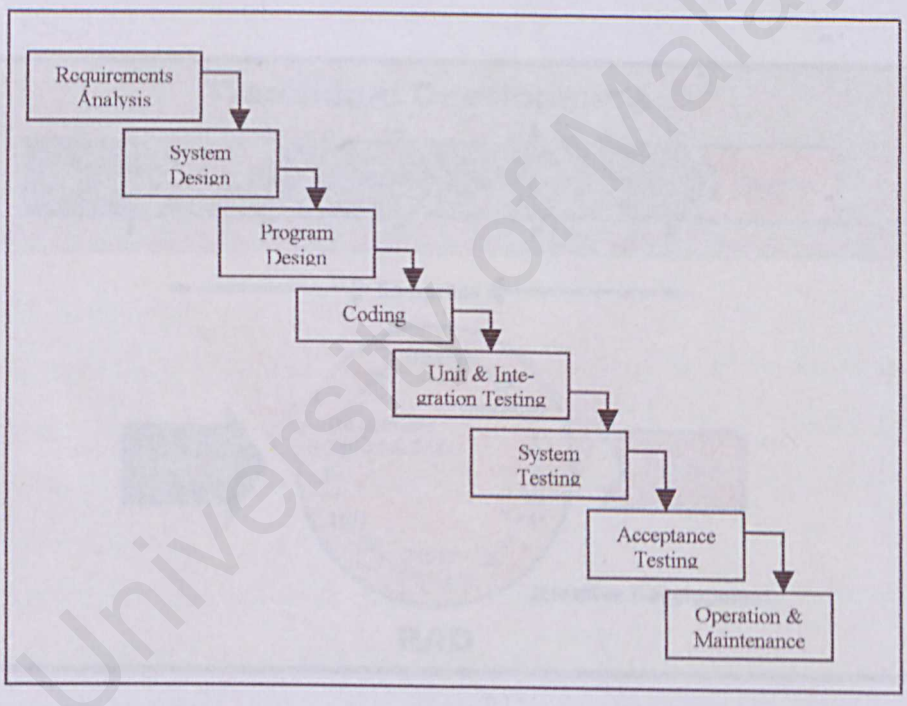


Figure 3.1 The Waterfall Model

The waterfall model can be very useful in helping developers lay out what they need to do. Its simplicity makes it easy to explain to customers who are not familiar with software development; it makes significance which intermediate products are necessary

in order to begin the next stage of development. Aside from the advantages mentioned, it has received many critics ever since it was introduced. The biggest problem with the waterfall model is that it does not reflect the way code is really developed, software is usually developed with a great deal of iteration. Other than that the model fails to treat software as a problem solving process and it does not provide guidance to developers on how to handle changes to products and activities that are likely to occur during development. Many other, more complex models are really just modifications and improvements of the waterfall model, incorporating feedback loops, extra activities and prototyping.

3.3 Rapid Application Development Methodology (RAD)

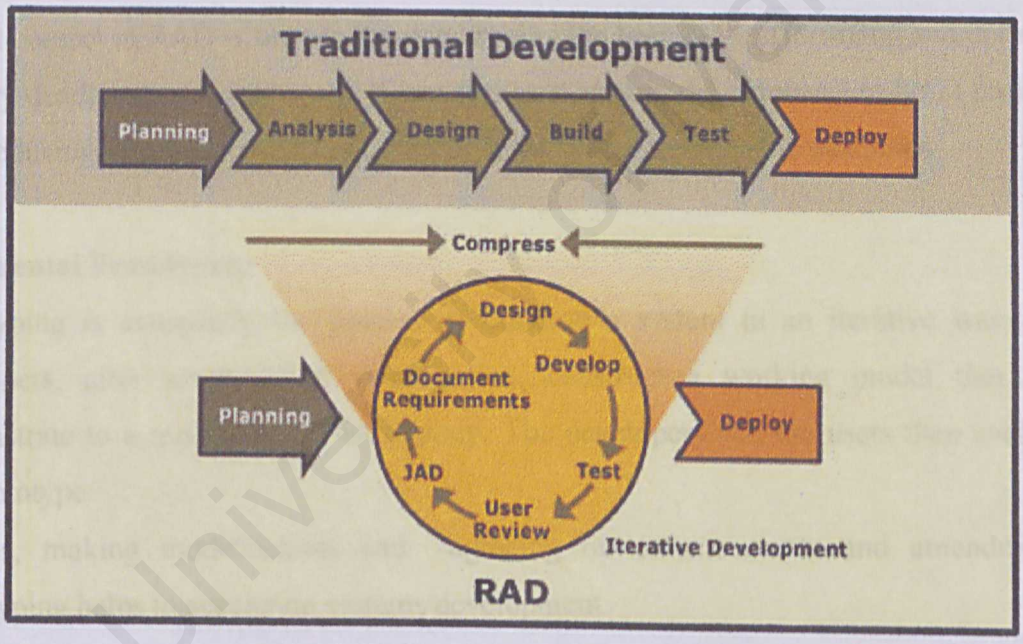


Figure 3.2 Comparison of RAD and the traditional Waterfall model

Rapid Application Development (RAD) is a methodology for compressing the analysis, design, implementation/build and test phases into a series of short iterative development cycles. The key objectives of RAD is high quality systems, fast development and

delivery, and at a low cost. The types of development to which RAD is especially seen as suitable are those applications where:

- The system to be build is of a small-scale and of short duration (2-6 months)
- The application is interactive
- The functionality is clearly visible at the user interface
- The user group is clearly defined
- The functionality of the system is not computationally complex

These are the common components of RAD:

Timeboxing

Project control in RAD is seen to involve scoping the project by prioritizing and defining delivery deadlines or 'timeboxes'. If projects start to slip, the emphasis in RAD projects is on reducing the requirements to fit the timebox, not increasing the deadlines.

Incremental Prototyping

Prototyping is essentially the process of building a system in an iterative way. The developers, after some initial investigation, construct a working model that they demonstrate to a representative user group. The developers and the users then evaluate the prototype designs, making modifications and agreeing on enhancements and amendments. Prototyping helps to accelerate systems development.

Rapid Development Tools

Projects implementing RAD should not involve complex programming. This normally means some combination of fourth generation languages (4GLs) and graphical user interface builders (interface design software). The project does not include hard-core programming but more to the assistance of application software.

Joint Application Design (JAD)

Small development teams usually characterize RAD. The teams are made up of both developers and users. JAD gives the developer an opportunity to elicit requirements and discuss on systems design with users. This is important in order to produce high quality software.

The Advantages of RAD

- Iteration allows for effectiveness and self-correction. The development process is much more flexible and efficient, because system developers can quickly readjust the development direction toward the required functionality.
- Shorter development period and an early delivery date is expected. By using RAD, the complete version of the system is available much earlier than with a conventional model. Due to the iterative cycles done on important stages, early detection of faults and modifications on requirements saves time and effort of altering the system at a costly stage.
- High-quality systems can be delivered as developers work closely with users to extract requirements of the highest priority and modifying the design many times until the satisfied design is established.

The Disadvantages of RAD

- RAD is not suitable for real-time or safety-critical systems such as the flight monitoring system.
- It is also not suitable for very large infrastructure systems, computationally complex systems and applications in which the functional requirements have to be fully specified before any programs are written.

- RAD is not appropriate in a system where many new technologies are to be introduced in the scope project, as too many new technologies require more time for familiarization and would effect the delivery stage.

3.4 Justifications In Choosing Rapid Application Development Methodology

Choosing the right software development model to suit a particular system is very important in ensuring the whole development process flows smoothly and is successfully delivered within the deadline. The preferred process model for the implementation of this multimedia package is the Rapid Application Development Model. These are the justifications:

- The iterative cycles and prototyping within the model would facilitate in modifications on the system design or amendments of user requirements that are due to occur later throughout the development process, making the whole process very flexible to changes.
- With the iterative procedure, time and effort could be saved as changes are done gradually during development and not at the final stage. Thus, resulting in a shorter development time.
- Another reason is due to the characteristics of RAD that is meant for developing systems such as this multimedia package. This package is suitable for RAD as it is of a small-scale, given a short duration of four to five months, implementing interactivity, and does not involved much of complex programming. Due to the short duration given, the package could benefit from 'timeboxing'.

- As for JAD, in this project, assumptions have to be made that the users where the developers work with to obtain requirements are peers and the lecturer (supervisor). It is true that to acquire information relating to interface designs or

- system requirements in this project a small Joint Application Development session need to be held with the project's supervisor and peers.

3.5 Capturing the Requirements

Requirements analysis is always the first step in any software development process. It helps to identify important system and users requirements in order to produce software of acceptable quality and utility. Therefore, it is considered as a very important step and always come first in any software life cycle. It is considered as a critical part of software development process. Variety of techniques is used to determine what the users and customers really want. These are the techniques used in order to obtain requirements for this system:

Brainstorming and discussion with customers

In this software development project, the supervisor acted as the customer of which the developer is building the system for. Assuming that this package is developed for the customer, it is important to know what kind of features that the customer would like to see on the system. Thus, it is important to work closely with the supervisor to extract the requirements. Weekly meetings are done to validate the requirements. This is part of an iterative process done to assure that the software will meet the customer's exact requirements.

Discussions and interviews with potential users

As for peers, they can be assumed as the potential users of this system. Users are as equal important as customers because they are the ones that will be utilizing the software. Through discussions and interviews their requirements could be obtained. Most of their views reflect on the interface designs and navigation features. This group of resources helps in providing opinions on the user-friendliness criteria that the software should posses.

Reviews on current existing software

This evaluation technique is essential in giving ideas of the characteristics that a good multimedia learning software should present. Besides that, it gives a picture of how the software's contents should be structured and presented to the users. As the reviewing process is conducted, assuming we are normal users, we could determine the advantages and the disadvantages of the software reviewed. Indirectly, it helps to convey the important criterias that should not be overlooked in this software.

3.6 Approach to the Development of the System

Multimedia development, like any production process that involves the coordination of multiple skills, can be complicated. Careful planning and designing of the overall system can help to ensure that the project moves along smoothly and efficiently. The overall objective of the planning process is to accurately assess the time, resources and costs (commercial purpose) associated with a project. It can also help identify and avoid any obstacles to completion.

The planning and designing process occurs in stages. The first stage is the development of a project description, or ***proof of concept***, which is a written description of the project. A good proof of concept is useful for consolidating and directing the ideas for a project before any complex computer work is started. The ***proof of concept*** for this project consists of:

1. **System requirements definition :** the system's criteria
2. **Functional requirements:** the system's interaction with its users
3. **Non-functional requirements:** the system's behaviour
4. **Project scope:** the target audience and modules involved

The next important stage of project planning is the development of a **storyboard** and a **logic flowchart**, which will be explained in Chapter 4 under System Design.

3.7 System Requirements Definition

A requirements definition document is written in terms that the customer can understand, the requirements definition is a complete listing of everything the customer expects the proposed system to do. In this software development, the system will apply the 'timeboxing' feature of RAD methodology, requirements might be added or reduced depending on the capability of accomplishing the requirements within the deadline. Below are the requirements of the system that were successfully elicited through the requirements capturing process:

1. The learning package is presented in English
2. The medium of teaching to be incorporated must include:
   - Text
   - Voice / narration
   - Graphics
   - Animation
3. The learning package should include an explanation on the C programming language.
4. The software should provide relevant topics on C programming language.
5. For each topic or sub-topic, a user is given an explanation and a choice for further explanation, which will explain the topic in more detail.
6. The software should provide methods to help users in better understanding of program reading during program executions.
7. It should be able to teach users on how to read a program.
8. The package must make the whole learning process an effective experience, which helps to increase the retention and comprehension rates.

3.8 Functional Requirements

A functional requirement describes an interaction between the system and its environment. Here are the lists of functional requirements for this software:

1. Narration and texts will be conveyed in one version which is English.
2. There will be an Introduction module to give a brief explanation about C programming language.
3. Relevant topics that will be discussed in this learning package will be placed in the Lesson Module.
4. For further explanation on the topics discussed the system will provide an Additional Lesson Module where more explanation could be accessed.
5. For in-depth understanding of program execution and its output there will be line-by-line program execution examples with outputs. All the sample codes included in the package can be obtained from the Sample Code Module.
6. In order to make the learning package effective, interactive tutorials and practices are included at the end of every sub-topic within every chapter. Tutorials are made available in the Tutorial Module. In addition to that, tutorials that are provided at the end of every sub-topic consist of two varieties; interactive multiple choice questions and a programming question with an interactive button to view the answers. Answers can only be accessed via the Tutorial Module to encourage effective learning.

3.9 Non-functional Requirements

Non-functional requirements focus on the user-friendliness, interactivity, usability and efficiency of the software.

1. User friendly

User friendliness of an application is always associated with the interface design. As for this software, it provides a simple, consistent and informative interfaces. The functions and sections within the software are carefully layout for easy accessed and recognition. Besides that, the components on every interface are minimized to reduce information overload and screen cluttering, which could confuse users. Other than that, navigational controls are provided to assist users with the information flow. Navigational buttons such

as "Previous", "Forward", "Continue Lesson", "Sound Toggle", "Main" and "Exit" as well as buttons to access every lesson individually will be provided. Another friendly feature is the ability to link sub-modules from the main menu. For example, the Tutorial Module within the Lesson Module can be access directly at the main menu without having to pass through Lesson Module.

2. Interactivity

The main feature of interactivity within this learning package is the tutorials and the line-by-line program execution simulation. The tutorials provide practices for users to test their knowledge and answers will be provided interactively with a press of a button. The program execution simulation will help users reinforce their understanding of the program execution sequence and the expected outputs. Meanwhile, the tutorials provided test user's knowledge regarding the sub-topic and gives answers in real-time where a window with remarks are presented with every right or wrong answer chosen by the user. An addition to that, a programming question with button for answers is also made available as a means of effective learning. Other than these features, the navigational controls provided is part of the software's interactivity capabilities, which give users control over their own learning pace.

3. Usability

The usability rate of this software is high as the interface design and the contents are clearly defined. Users will not face difficulty in accessing modules within the learning package because hypertext representations and icons are provided. The navigation tools will enhance the user's movement around the software contents. Thus, providing a fully interactive environment where all choices are made by users. Apart from that, the Additional Lesson Module within the Lesson Module provides detail information for extra knowledge on every sub-topic that are discussed in this presentation.. Users will discover that with these helpful features this software provides the highest ease-of-use.

4. Efficiency

The efficiency of linking to a page when the function is prompted lies within the hypertext feature and navigational controls of the software. With these features, the users learning experience will be made efficient. Not only the package is easy and effective it also allows fast and efficient learning.

3.10 Summary

This chapter concentrates on the important aspects of a software life-cycle. Basically a software process involves requirements analysis and definition, system design, program design, implementation, unit testing, integration testing, system testing, system delivery and maintenance. Two examples of software life-cycles are the Waterfall Model and the one adopted in this project the Rapid Application Development methodology. Waterfall is a very linear life-cycle where as RAD compresses the development processes into series of short iterative cycles.

A few of RAD components include 'timeboxing', incremental prototyping, rapid development tools and Joint Application Design. One of its advantage is that the development process is much more flexible and efficient as iteration allows for effectiveness and self-correction. On the other hand, RAD is not suitable for real-time or safety-critical systems and large infrastructure systems.

The preferences towards applying RAD in this project are due to many reasons. Compared to the rigid and linear Waterfall model, RAD is much more flexible and reflects the real software development process that is mainly consist of iterative processes. Iterations within the software life cycle encourage early fault detection and a system that is very close to the users' expectations. RAD is very suitable to be implemented on small-scale software development projects such as this one. It saves time and effort.

The most critical part of a software life cycle is the requirements capturing stage. It has to be done in many effective ways in order to get the right information from the right resources. In this project brainstorming and discussions with the project's supervisor and peers are among the techniques involved. They are very important resources as they act as the system's customer and users. Other than that, reviews on current existing softwares help in providing a clearer picture of the latest trends and technologies implemented in software development nowadays.

The most integral part of a system is its requirements definitions. This is a document written in terms that the customer can understand and it reflects on the customer requirements towards the system capabilities. The full list of definitions can be found in the system requirements definition topic at 3.6. Besides the requirements definition, every system should have functional and non-functional requirements that describe the interaction between the system and its environment and the system user-friendliness, interactivity, usability and efficiency respectively. The complete lists are in the sections 3.7 and 3.8.

CHAPTER 4

SYSTEM DESIGN

Chapter 4 System Design

4.1 Overview

The next stage to be encountered, upon the completion of the system and requirement analysis process is the design stage. In design stage, the structure and contents are layout. This stage is the platform where all of the system problems are transformed into solutions. An important issue in system design is the design styles and techniques used in conveying the architecture of the whole system in the most understood concept.

4.2 Conceptual Design

A conceptual design tells the customer of the overall system functions and purpose in a language that the customer can understand, rather than in computer jargon and technical terms. It helps customer to understand every different component of the software and its functions.

Basically, the overall system consists of six modules which are the Introduction module, Lesson module, Additional Lesson module, Sample Code module, Tutorial module and Answer module. Here are the descriptions of each module and its functions.

1. Introduction Module

In this module users are given a brief introduction on C programming language and its importance. Other than that, in this section users are also informed of the reasons as why to learn the C programming language.

2. Lesson Module

The Lesson module gives access to all the programming topics and its sub-topics. The topics and sub-topics are listed as below:

i) C Fundamentals

- Components of C program
- Create and Compile a Program
- Variables
- Constants
- Arithmetic Operations
- Comparison Operators
- Order of precedence

ii) Conditionals

- The if Statement
- The ? Operator
- The Switch Statement

iii) Loops and Iteration

- Increment and Decrement Operators
- The for Statement
- The while Statement
- The do-while Statement
- Break and continue

iv) Arrays and Strings

- Single and Multidimensional Arrays
- Strings

v) **Pointers**

- Pointers to Functions
- Pointers and Arrays
- Array of Pointers

vi) **Functions**

- Function Parameters
- Void Functions
- The return Statement
- Function Prototypes

In every sub-topic there will be sample codes execution simulation. With a click of the mouse these sample codes can be accessed. These sample codes are available within the Sample code module. Besides the codes execution simulation, at the end of every topic and sub-topic users will be tested through tutorials. A hypertext or button will be placed at the end of every topic and sub-topic that will transfer users into the Tutorial module where all tutorials reside.

3. Additional Lesson Module

This module provides additional information to the topics contained in the Lesson module. It provides users with further explanation on the relevant topics which they need more concentration. As for beginners, a brief explanation might not satisfy their understanding so they would need to go deeper into the subject for better comprehension. This is a very helpful feature as there will be more sample codes for users to observe and evaluate. Users will also be provided with more tutorial questions.

4. Sample Code Module

The sample code module holds samples of programs that can also be linked from the Lesson module. It provides interactive program execution simulation. There will be a line-by-line explanation on how the programs are executed. This is to educate users on how to read programs execution sequence. The sample programs are labeled carefully and meaningfully to avoid confusion and for ease at locating relevant programs. This module is also can be accessed via the main screen. Sometimes, advance learners they would like to make a quick reference by just evaluating a program execution sequence.

5. Tutorial Module

In this module a database of tutorial questions are kept and can be accessed through the Lesson module and other parts of the learning package. It is important to place a hypertext link from the main screen to the tutorial questions as there are users who's concept of effective learning is by testing on themselves these tutorial questions at the very end of all the lessons. The tutorial questions are placed at the end of topics and sub-topics. This tutorial module is equipped with interactivity as it supplies answers at the end of the tutorial questions. This module provides more tutorial questions that can also be accessed from Additional Lesson module.

6. Answer Module

The Answer module can only be accessed via the Tutorial module. Part of providing an effective learning concept, its intention is to encourage users to try out the tutorial questions. Somehow it is easier to comprehend answers when the questions are provided first. By just looking at the answers it is difficult to relate to the learning experience. Due to this fact the module access is not provided at the main screen.

4.3 System Modeling

The system is designed with modularity that partitioned every component within the system into well-structured modules. Modularity in a system design helps developers to easily maintain the software contents. In order to model the system's design, structured system modeling techniques are adopted. Structured modeling is a process-oriented technique that breaks up the overall system into hierarchy of structure chart modules that consist of the system's decomposition diagram.

Decomposition is the act of breaking the system into its component subsystem. It shows the hierarchy of the system contents. It gives a high-level description of the functions that are to be implemented and builds lower-level explanations of how each component will be organized and related to other components within the system.

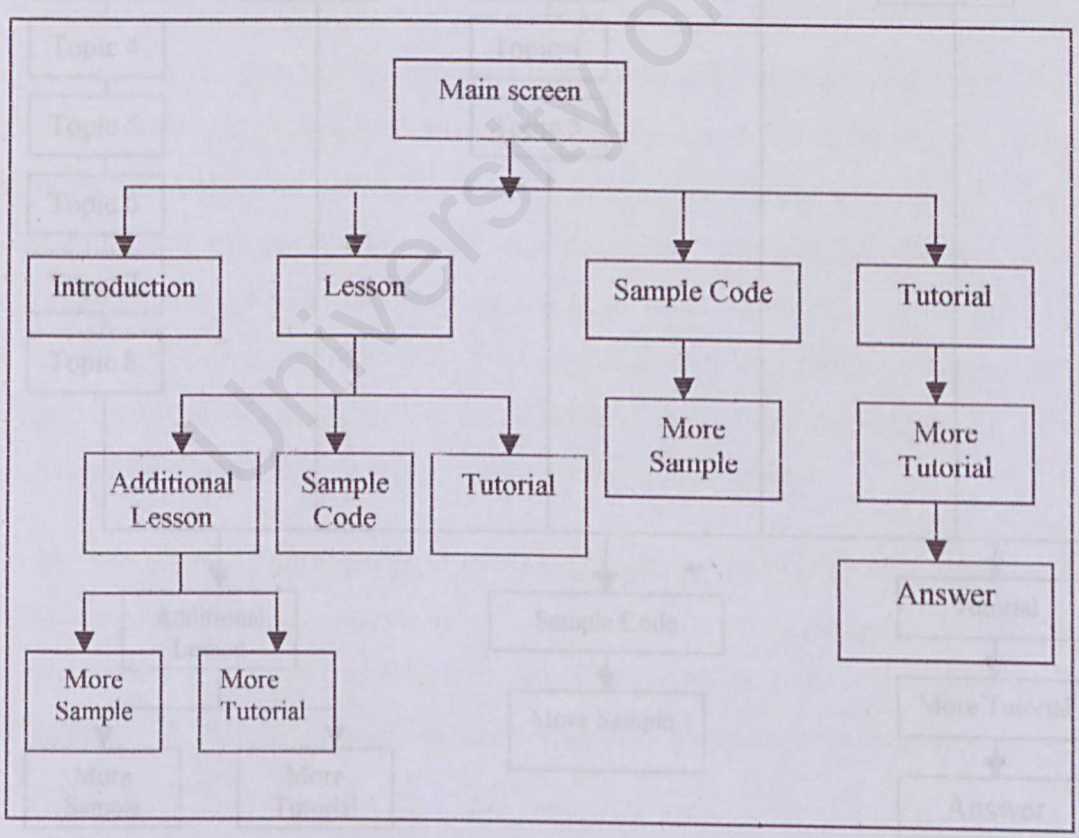


Figure 4.1 Decomposition diagram for the main screen of Learning How to Program in C learning package

Due to the enormous contents of the Lesson Module, an individual decomposition diagram for the Lesson Module has been constructed.

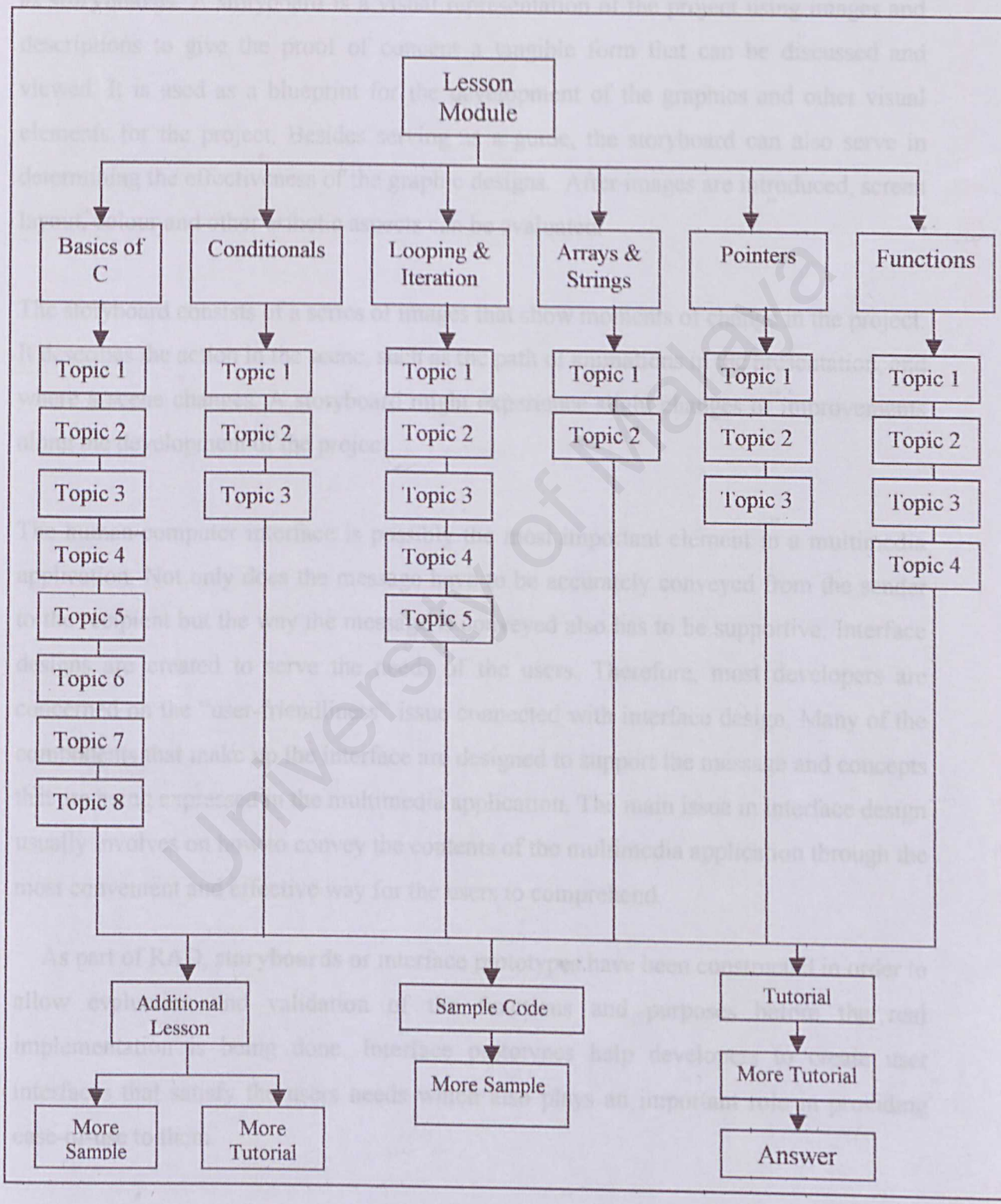


Figure 4.2 Decomposition diagram for Lesson Module

4.4 Storyboards (User Interface Design)

User interface design in a multimedia environment using Macromedia Director is known as **storyboards**. A storyboard is a visual representation of the project using images and descriptions to give the proof of concept a tangible form that can be discussed and viewed. It is used as a blueprint for the development of the graphics and other visual elements for the project. Besides serving as a guide, the storyboard can also serve in determining the effectiveness of the graphic designs. After images are introduced, screen layout, colour and other esthetic aspects can be evaluated.

The storyboard consists of a series of images that show moments of change in the project. It describes the action in the scene, such as the path of animations in the presentation, and where a scene changes. A storyboard might experience slight changes or improvements along the development of the project.

The human-computer interface is possibly the most important element in a multimedia application. Not only does the message have to be accurately conveyed from the sender to the recipient but the way the message is conveyed also has to be supportive. Interface designs are created to serve the needs of the users. Therefore, most developers are concerned on the "user-friendliness" issue connected with interface design. Many of the components that make up the interface are designed to support the message and concepts that are being expressed in the multimedia application. The main issue in interface design usually involves on how to convey the contents of the multimedia application through the most convenient and effective way for the users to comprehend.

As part of RAD, **storyboards** or interface prototypes have been constructed in order to allow evaluation and validation of the functions and purposes before the real implementation is being done. Interface prototypes help developers to create user interfaces that satisfy the users needs which also plays an important role in providing ease-of-use to them.

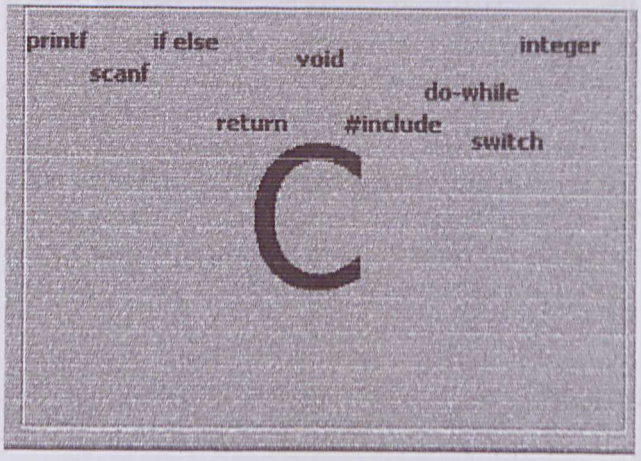


Figure 4.3: Part 1, Opening Animation

Description

Nine icons of programming keywords animate by falling from the top part of the screen. In the background is a synthesized drum beat music.

Animation

Nine icons of programming keywords animate in two cycles pattern. The icons are: keyword printf, scanf, if else, return, void, #include, do-while, switch and integer.

Transition

None

Interactivity

None

Audio

Synthesized drum beat

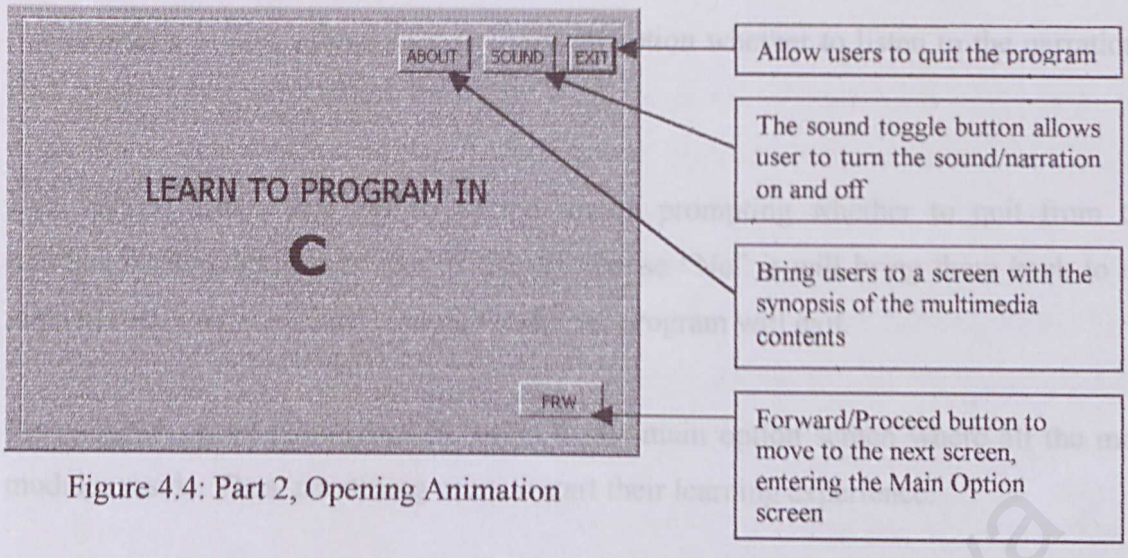


Figure 4.4: Part 2, Opening Animation

Description

All of the nine keyword icons disappear at the lower part of the screen. The word "LEARN TO PROGRAM IN " animates from the center part of the screen to full size. The alphabet C is at the background. At the same time four buttons appear on the static interactive screen and a welcoming speech is presented. At upper right part of the screen there are three buttons About, Sound and Exit. At the lower right one button which is the Forward/Proceed button.

Animation

The word "LEARN TO PROGRAM IN" animates from small to large from the center of the screen. Button rollover and pressed states.

Transition

Dissolve Bits upon clicking button Forward/Proceed

Interactivity

About button: takes the viewer to an information screen with a synopsis of the overall multimedia presentation, the software used to develop this presentation.

Sound toggle button: allows user to make an option whether to listen to the narration / background music or to switch them off.

Exit button: brings users to an option screen prompting whether to quit from the multimedia presentation or stay. If the user chose "No" it will bring them back to the main option screen. If "Yes", automatically the program will exit.

Forward/ Proceed button: Access users to the main option screen where all the main modules reside. Thus, permitting users to start their learning experience.

Audio

A welcome speech, no background music

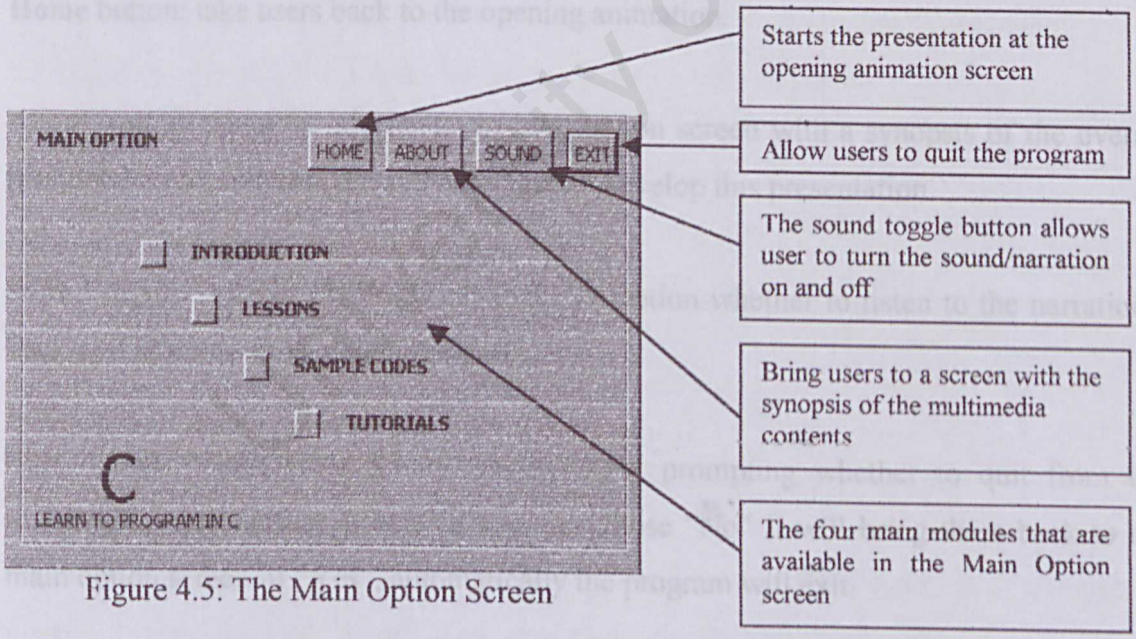


Figure 4.5: The Main Option Screen

Description

This is the main option screen that gives users access to four main modules in this presentation; the Introduction Module, Lesson Module, Sample Code Module and Tutorial Module. The background texture and colour are as similar as the opening animation background, which is a textured purple background graphic. This screen provides four buttons which Exit, Sound toggle, About and Home buttons. At the lower left part of the screen there is an embossed C logo with the title of the multimedia CD underneath the C logo. All word labels such as "Main Option" and "Lessons" have a 3D effect and shadowed.

Animation

Button rollover and pressed states, transitions

Transition

Dissolve Bits on initial entry.

Interactivity

Home button: take users back to the opening animation.

About button: takes the viewer to an information screen with a synopsis of the overall multimedia presentation, the software used to develop this presentation.

Sound toggle button: allows user to make an option whether to listen to the narration / background music or to switch them off.

Exit button: brings users to an option screen prompting whether to quit from the multimedia presentation or stay. If the user chose "No" it will bring them back to the main option screen. If "Yes", automatically the program will exit.

Audio

None

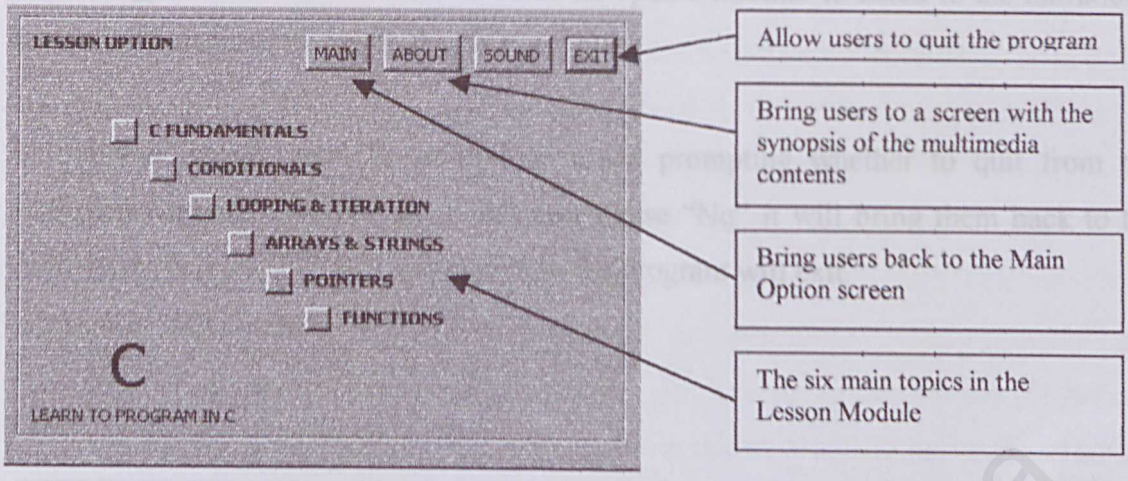


Figure 4.6: The Lesson Option Screen

Description

This interactive screen provides access to the six main topics that will be covered in this learning package. On the upper right of the screen is four important navigational buttons. At the lower left side of the screen is the embossed C logo and underneath is the title of this multimedia package. All word labels are in the form of shadowed 3D graphics. The background graphic is similar to the one in the Main Option screen.

Animation

Button rollover and pressed states, transitions.

Transition

Dissolve Bits on initial entry

Interactivity

Main button: this button helps users to return to the Main Option screen.

About button: takes the viewer to an information screen with a synopsis of the overall multimedia presentation, the software used to develop this presentation

Sound toggle button: allows user to make an option whether to listen to the narration / background music or to switch them off.

Exit button: brings users to an option screen prompting whether to quit from the multimedia presentation or stay. If the user chose "No" it will bring them back to the main option screen. If "Yes", automatically the program will exit.

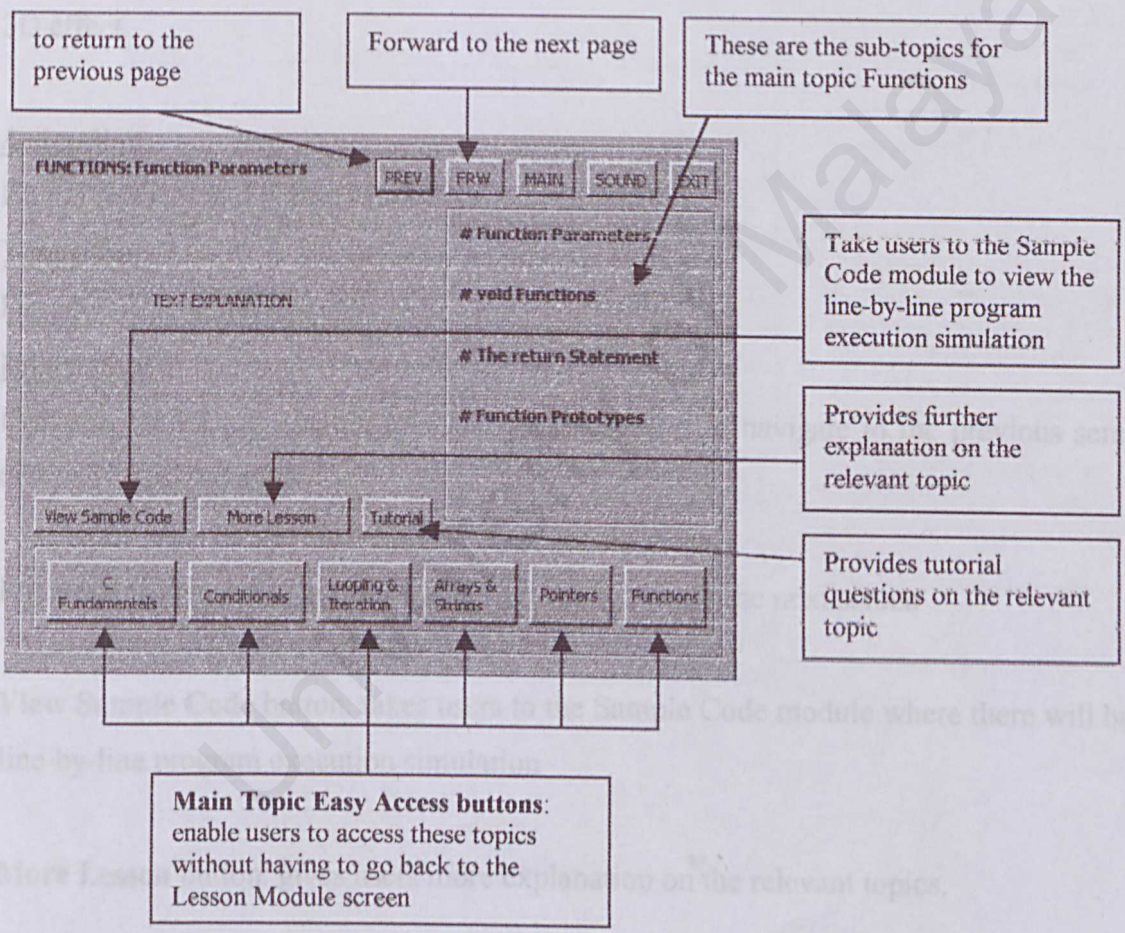


Figure 4.7: The Lesson module screen

Description

This screen resides in the Lesson Module. It contains text explanation and narration on the relevant sub-topics. The explanation might span to more than one page. There are many navigational buttons provided within this screen. There are five buttons on the upper right part of the screen, which has two new button features; the "Previous" and "Forward" button. These buttons are round in shape and have a 3D effect. Underneath the explanation text there are three buttons; "View Sample Code", "More Lesson" and "Tutorial" button. At the lower part of the screen, six navigational buttons are provided to help users access the other main topics easily. These buttons are oval in shape and have a 3D effect.

Animation

Button rollover and pressed states.

Transition

Dissolve Bits on initial entry.

Interactivity

Previous (left-arrow) button: this button helps users to navigate to the previous screen that had been visited.

Forward (right arrow) button: this button takes users to the next screen

View Sample Code button: takes users to the Sample Code module where there will be a line-by-line program execution simulation

More Lesson button: gives users more explanation on the relevant topics.

Tutorial button: this button will present users with multiple choice questions and a programming question with given answers.

Main Topic Easy Access buttons: the Main Topic buttons are located at the bottom of the screen and they consist of six buttons **C Fundamentals, Conditionals, Looping & Iteration, Arrays & Strings, Pointers** and **Functions.** Each of the buttons holds access to the topic as stated on each button. This way it is convenient for users to access to any other topic from any point of the lesson.

Audio

Narration on every lesson page.

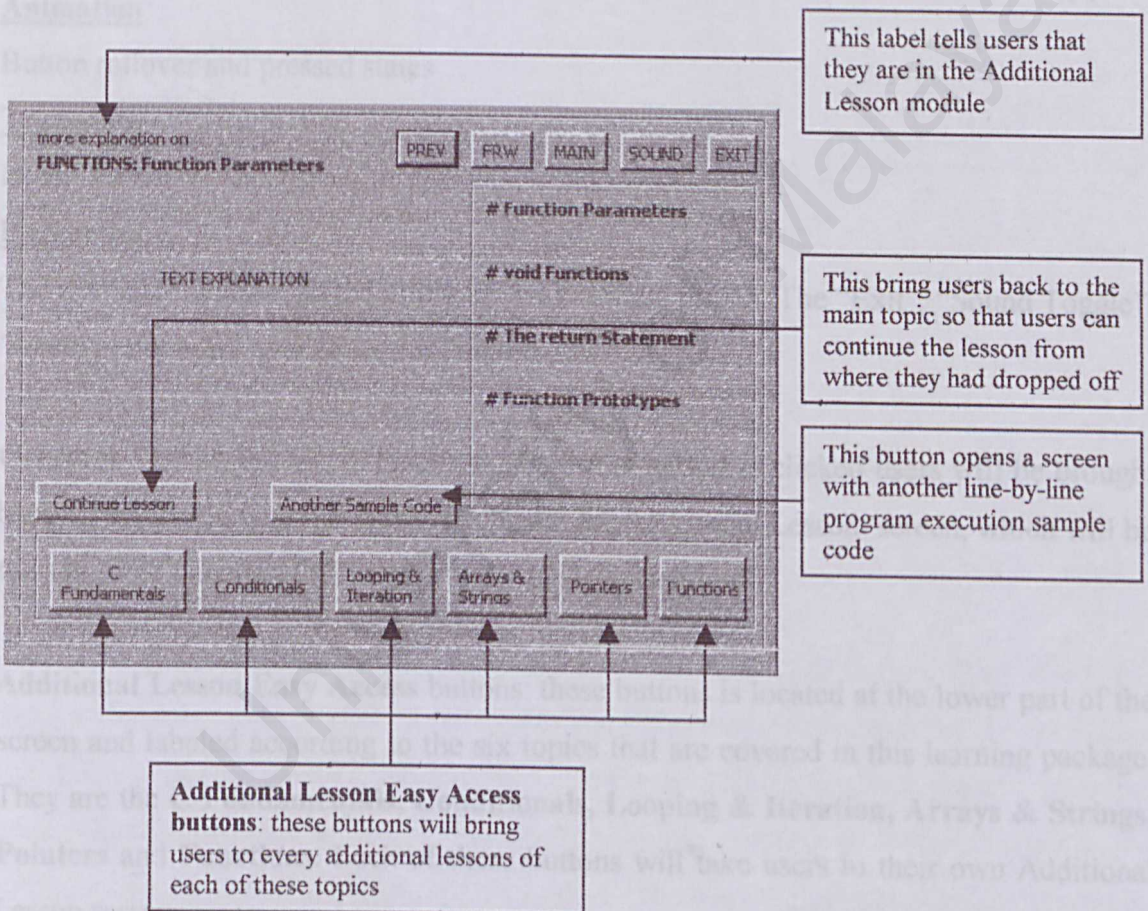


Figure 4.8: The Additional Lesson module screen

Description

This screen belongs to the Additional Lesson Module and is accessed via the Lesson Module. It elaborates on the sub-topics from the Lesson Module. This module is essential for beginners who need more resources of information on the programming topics. This module is part of an effective learning concept. This screen contains the five main buttons on the upper right side of the screen that are similar to the ones in the Lesson module. Other additional buttons are the "Continue Lesson" and "Another Sample Code" button.

Animation

Button rollover and pressed states

Transition

None

Interactivity

The five main buttons on the upper right side of the screen. The "Exit", "Sound Toggle", "Main", "Previous" and "Forward" button.

Continue Lesson button: in this case, when this button is clicked users will be brought back to the screen where they had accessed this Additional Lesson screen, which will be the sub-topic Function Parameters of the Lesson Functions.

Additional Lesson Easy Access buttons: these buttons is located at the lower part of the screen and labeled according to the six topics that are covered in this learning package. They are the **C Fundamentals, Conditionals, Looping & Iteration, Arrays & Strings, Pointers** and **Functions.** Each of these buttons will take users to their own Additional Lesson screens.

Audio

Narration

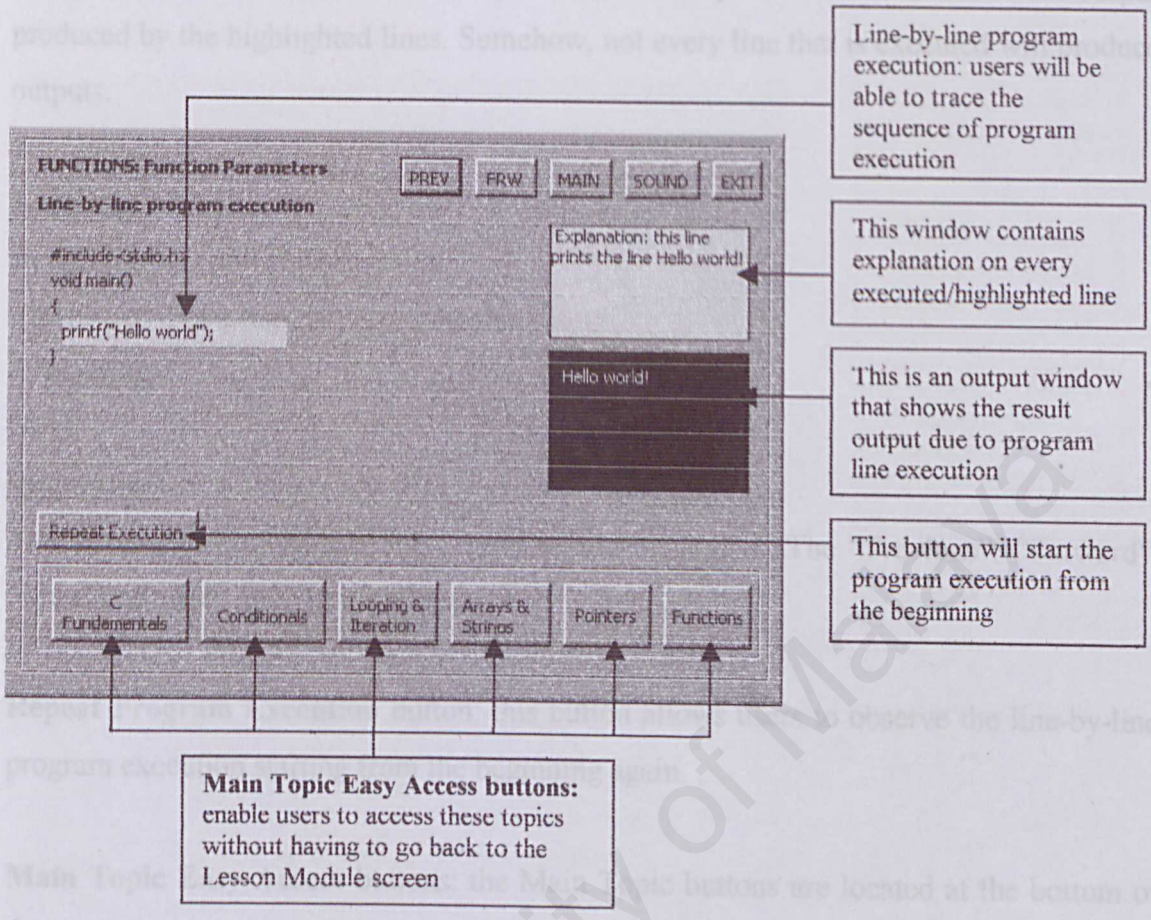


Figure 4.9: The Sample Code module screen

Description

This is the Sample Code screen. There will be one sample code at the end of every sub-topic and another sample code at the end of the additional lesson sub-topic. Overall there will be two sample codes provided for a sub-topic. This screen will present a sample code, which allows users to observe the line-by-line program execution sequence. At every executing line a highlight bar will appear to focus the attention on the particular line. At the same time, two windows will appear at the right side of the screen. The window with brown background will have explanations on the particular highlighted line.

The second window with black background is the output window that shows the output produced by the highlighted lines. Somehow, not every line that is executed will produce outputs.

Animation

Button rollover and pressed states.

The line-by-line program execution simulation.

Transition

None

Interactivity

The five main buttons that have been explained earlier. The "Previous", "Forward", "Main", "Sound" and "Exit" button.

Repeat Program Execution button: this button allows users to observe the line-by-line program execution starting from the beginning again.

Main Topic Easy Access buttons: the Main Topic buttons are located at the bottom of the screen and they consist of six buttons **C Fundamentals, Conditionals, Looping & Iteration, Arrays & Strings, Pointers** and **Functions.** Each of the buttons holds access to the topic as stated on each button. This way it is convenient for users to access to any other topic from any point of the lesson.

Audio

Narration

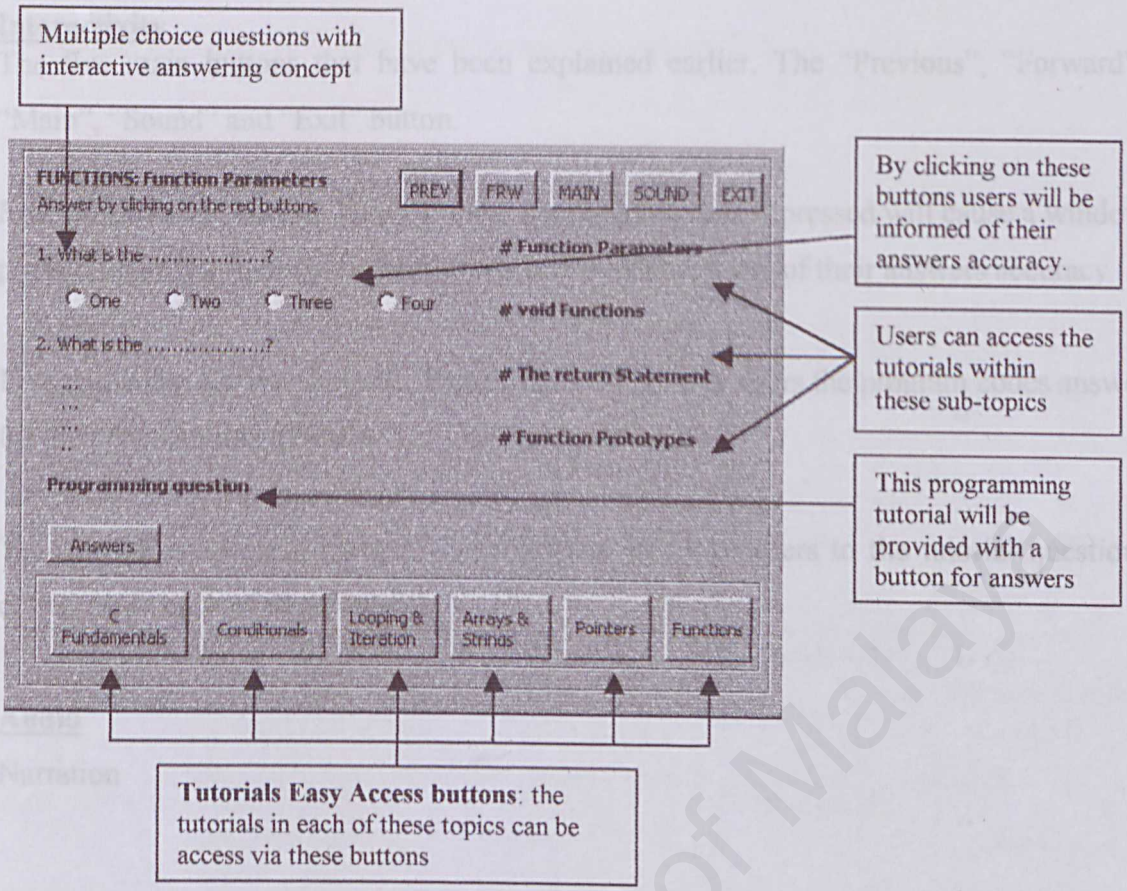


Figure 4.10: Tutorial module screen

Description

Within this screen are four multiple choice tutorial questions with interactive answering approach where answers could be obtained with just a click of the button. The tutorial will interact with the users by presenting a window with remarks of the answer's accuracy. Other than that, every tutorial will include one programming question. The answers for this programming question could be viewed interactively by clicking on the "Answer" button provided. Links to other sub-topic tutorials are provided on the right side of the tutorial screen.

Animation

Button rollover and pressed states.

Transition

None

Interactivity

The five main buttons that have been explained earlier. The "Previous", "Forward", "Main", "Sound" and "Exit" button.

Multiple Choice Answer buttons: these small red buttons if pressed will cause a window popped up on the right side of the screen informing the users of their answers accuracy.

Programming Answer buttons: This button will provide users the program codes answer for the programming question.

Tutorials Easy Access buttons: these buttons will take users to the tutorial questions contained in each of these topics.

Audio

Narration

4.5 Inventory

Once all of the planning and designing has been made for the design of the project, attention can be placed on planning and creating the elements that will actually be included. This not only helps organize the project, but also helps determine the software needed to develop the elements. The types of graphics or images that are chosen to be included in the project determine the type of graphics or image editing and creation software that will be used.

Part of the process of organizing the media for a project is developing the ***inventory,*** a list of all of the media needed. An inventory typically contains the name, media type, and location in the project of each item used in the project. Besides serving as an organizational tool, the inventory can also serve to put into full perspective the scope of what is required to create the project. The inventory is not only a list of file names, it is a

production tool. Below, is part of the project's inventory list.

Inventory List for Main Option Screen:

Name	FileName	CastName	Type	Purpose
3rd background	background3.psd	background3	graphic	Back drop (725 x 525)
Home button	home1.psd home2.psd home3.psd	home1 home2 home3	Button(graphic)	Go to opening animation
About button	abot1.psd abot2.psd abot3.psd	abot1 abot2 abot3	Button(graphic)	Go to presentation synopsis
Sound button	audi1.psd audi2.psd audi3.psd	audi1 audi2 audi3	Button(graphic)	Toggle narration/sound on and off
Exit button	quit1.psd quit2.psd quit3.psd	quit1 quit2 quit3	Button(graphic)	Go to quit option screen
Main option screen title	utam1.psd utam2.psd utam3.psd	utam1 utam2 utam3	graphic	Title for main option screen
Introduction button	intr1.psd intr2.psd intr3.psd	intr1 intr2 intr3	Text button (graphic)	To access introduction module screen
Lessons button	Less1.psd Less2.psd Less3.psd	Less1 Less2 Less3	Text button (graphic)	To access the lesson option screen
Sample Codes button	samp1.psd samp2.psd samp3.psd	samp1 samp2 samp3	Text button (graphic)	To access the sample codes option screen
Tutorials button	Tuto1.psd Tuto2.psd Tuto3.psd	Tuto1 Tuto2 Tuto3	Text button (graphic)	To access the tutorials option screen

Table 4.1: Main Option Screen Inventory list

After developing the inventory lists for the elements, the next step in the process of multimedia development is the actual execution of the project.

4.6Summary

In this chapter focus is given on the design issues of the software. This is the stage where the structure and contents of the system is being layout. This stage is where developers, customers and users need to make agreements and validations on important design issues relating the system. The first important aspect to be determined is the conceptual design of the system that is the description of the whole system functions and purpose in a language that the customer can easily interprets.

This system is easily explained by structuring the system's contents in modules. This software is consist of six modules, the Introduction Module; brief explanation about C programming and its importance, the Lesson Module; focuses on six main topics and each consists of several sub-topics, Addditional Lesson Module; provides further explanation about the topic or sub-topic and also more program examples and tutorials, the Sample Code Module; grouped all the programs which have a simulation that could be viewed by users in a line-by-line explanation, the Tutorial Module; holds all the tutorial questions and the Answer Module that could be accessed via the Tutorial Module.

A system is well understood through visual explanations. Thus, in system design system modeling is essential in describing a system's structure in an easy way to be comprehended by both the customer and the users. As for this project, the contents are best explained through decomposition diagram, storyboards and inventory. The components can be observed in detail without complexity of understanding, as it does not involve technical drawings.

CHAPTER 5

SYSTEM IMPLEMENTATION

Chapter 5 System Implementation

5.1 Overview

This is the stage where all of the plans and ideas are given digital form. In order to develop the system it is important to work based on the proof of concept and the system designs specifications created earlier. The implementation stage involves creating functions and multimedia elements that are to be incorporated into the system. In this stage the focus will be on the system development softwares that are utilized in order to produce the multimedia presentation and multimedia elements.

5.2 Macromedia Director 8.5 Shockwave Studio

This is the multimedia authoring software used to develop "Learn to Program In C" multimedia learning package. It is a popular multimedia presentation development software in the market. Macromedia Director 8.5 is the latest version of Macromedia Director software and it offers many powerful functions that are useful to develop interactive multimedia presentations. It is a tool originally designed by artists for artists that later had programming elements incorporated into the interface. The scripting language that is part of Director, named Lingo, is a sophisticated multimedia scripting language and is simple to master. Director is an excellent option for anyone interested in developing interactive presentations. Director's interface is based upon three central windows: the ***Stage***, the ***Cast*** window, and the ***Score***.

The Stage

The Stage is the display area of the presentation and will contain any movie activity that should be visible to the end user. It is essentially a blank projection screen provided for authoring and placing the multimedia elements.

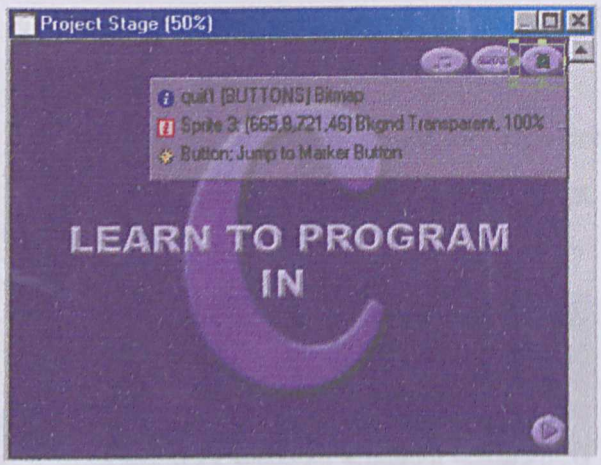


Figure 5.1: Project Stage

The Cast

The *Cast* is the storage location for all the media in this Director movie. When images and other media, such as text, are created or imported into Director, they become *Cast members.* Cast members are then integrated into the movie as frequently as necessary simply by clicking and dragging them to the desired location on the stage. Each Cast window can contain up to 32,000 Cast members. Each Director movie has an internal Cast window by default. As in this learning package there are eight Cast categories for convenience purposes. The Cast categories are as below:

1. Internal Cast: contains background images, opening animation graphics, option screens
2. Introduction Cast: contains all of the Introduction module elements
3. Lessons Cast: contains all of the Lessons module graphic and text elements
4. Sample Codes Cast: contains all of Sample Code module graphic and text elements
5. Tutorial Cast: contains all of Tutorial module text and graphic elements not including
6. Buttons Cast: contains all of the graphic and text buttons that are used in all of the screens within the presentation.

7. Narration Cast: contains all of the sound files and voice narrations in wave (wav.) format
8. Scripts Cast: contains all scripts that are used to produce animations and other functions scripts such as the button script.

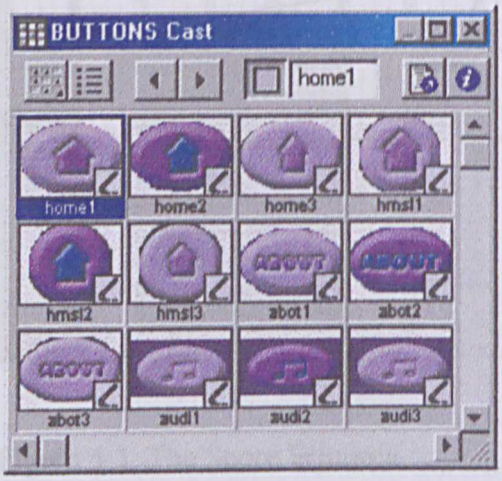


Figure 5.2: The Cast Window

The Score

The *Score* is Director's construction window. This is the place to set up different scenes in the presentation, similar to the scenes in a film production. Time passes from left to right and is measured in frames.

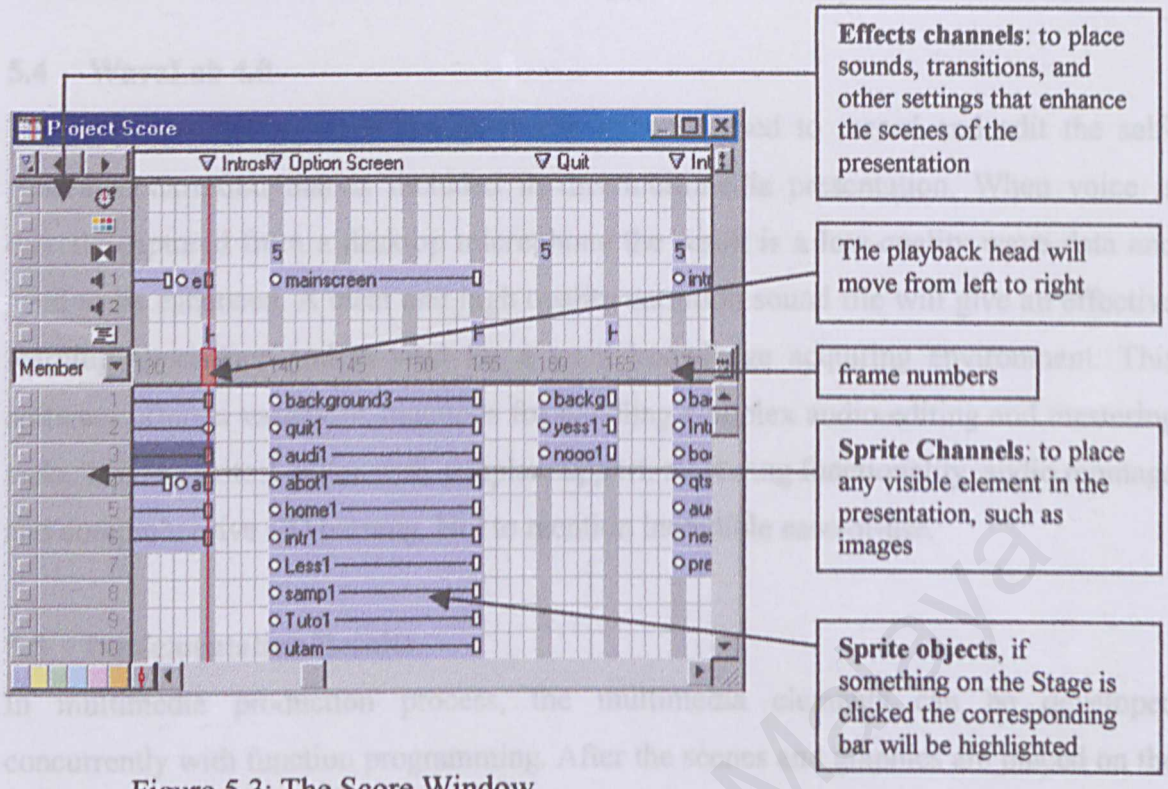


Figure 5.3: The Score Window

5.3 Adobe Photoshop 7.0

Adobe Photoshop 7.0 is the latest Adobe Photoshop program. It is a professional image-editing software and the main graphic software used for developing the images contained in this multimedia presentation. It is fully utilized to create all of the graphics within this learning package. Besides editing image, it can be utilized to create original images. It is also recognized as the most powerful and most amazing version of Photoshop. This software plays a vital role in producing high quality image and graphics for the learning package. By using this software the graphic production were made efficient and effortless. It provides many graphic effect functions, styles as well as pattern makers to give more creative choices when creating images or graphics for all occasions. Other than that, it offers sophisticated and up to date graphic tools, which are an advantage compared to other graphic software.

5.4 WaveLab 4.0

This is a sound design and audio editing software used to record and edit the self-generated narration that is included in this multimedia presentation. When voice is directly captured from a desktop microphone the result is a low-quality wave data and need to be enhanced. A clear and high quality narration sound file will give an effective learning experience and is vital for a good knowledge acquiring environment. This software offers a variety of functions for handling complex audio-editing and mastering tasks. It is the fastest editor with sampler support, mastering functionality, audio montage and comprehensive CD burning. Not to mention incredible ease-of-use.

5.5 Implementation Results

In multimedia production process, the multimedia elements can be developed concurrently with function programming. After the scenes and graphics are placed on the Stage according to the storyboards design, the programming scripts are assigned to the particular graphic in order to equipped it with the appropriate function. For example, for a graphic button to act as a button with rollover and pressed states it has to be assigned with a button script. Scripting or programming of multimedia elements will be discussed in the next chapter. Utilization of the softwares mentioned earlier is the final interface result of the multimedia presentation.

Below are a few examples of the final interface created in Director:

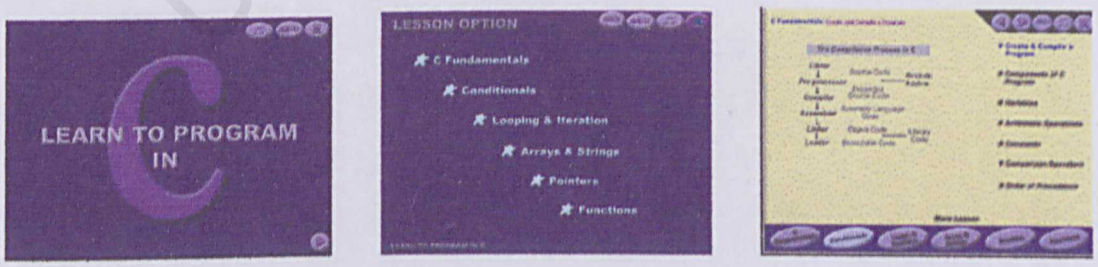


Figure 5.5: Examples of The System Final Interface

5.6 Summary

In this chapter, explanations were made on the softwares that are used in the development of this multimedia presentation. The multimedia authoring tool chosen for this project is Macromedia Director 8. Shockwave Studio. This software is chosen as it offers many advanced functions in developing a multimedia presentation. Director is a powerful authoring tool that offers fast and simple animation creation. The graphic elements are developed in Adobe Photoshop 7.0 the latest version that offers varieties of filters and effects to enhance graphic presentation. Wave files for the narration is being edited in WaveLab 4.0.

CHAPTER 6

WRITING AND TESTING THE PROGRAMS

Chapter 6 Writing and Testing the Programs

6.1 Overview

Two of Director's most significant features are scriptless authoring and the pure Lingo programming. In multimedia development, programming is important to add interactivity and animations to the presentation elements. Interactivity is highly expected within multimedia learning presentations as it is part of an effective learning concept. As for this leraning package it is an essential part of the system requirements. In this chapter, the focus is on programming with Lingo and testing the accuracy of thescripts.

6.2 Behavior Inspector

The behavior inspector introduces scriptless authoring, it is a simple Lingo script generator that is used to add scripts to the movie in a very quick and intuitive manner without requiring the knowledge of Lingo programming. The behavior inspector provides commonly used Lingo scripts. It provides settings that are accurate to create a particular behavior. These settings can be made just with a few clicks of the mouse. It means that Lingo can be added to a presentation without typing a single line of code. The behavior allows a combination of settings to be chosen according to the behavior that is required. It is almost similar to pre-defined functions. The settings are based on selecting an ***event*** (the action that will activate the behavior, such as a mouse click or a keypress) and an ***action*** (the result produced by the behavior, such as moving a sprite or playing a sound). Each event can have multiple actions associated with it. One or more of these event-action group are what define a behavior. A behavior setting that have been chosen can be saved and reapplied elsewhere if appropriate. Basically, the behavior inspector has made Lingo programming, which was once considered advanced and difficult, easy for an absolute beginner. These behaviors are very useful and efficient in helping to speed up the multimedia development process. Besides the predefined behaviors, it is possible to create customs behavior by editing the scripts of the behavior with custom Lingo programming.

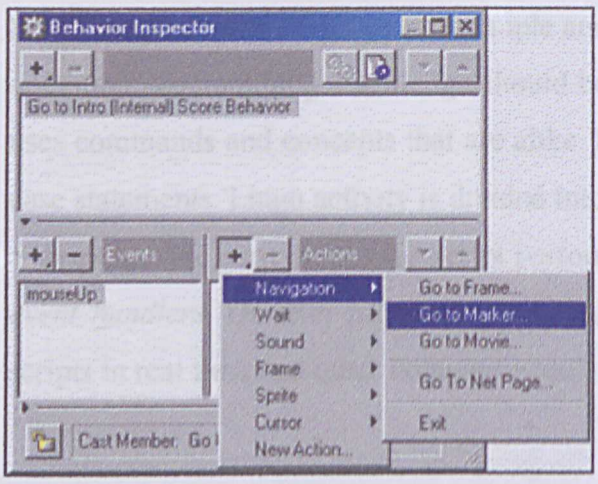


Figure 6.1: The Behavior Inspector window

The name of this behavior is Go to Intro. It will jump to the marker that was set at the beginning of the opening animation when the user clicked the mouse on the button assigned with this behavior.

There are many other combinations that can be selected under events and actions to suit certain required behavior.

Most of the programming used to create interactive and animations in this multimedia presentation are generated by the inspector behavior and are reused many times at suitable occasions through out the presentation. For example the action done when a button is clicked. Any of the behavior scripts generated will appear as Cast members in the cast window. As for this project, all of the behavior scripts are stored in the external Scripts Cast file. These behaviors can be applied on other button graphic that requires the same behavior by just dragging the script from the Script Cast and dropping it on the particular button graphic.

6.3 Lingo Programming

Lingo, the scripting language that were used to create the pre-programmed behaviors via the behavior inspector earlier, is actually a complete programming language that can be used to control virtually every part of Director's interface. Lingo, is a remarkable, and expansive, tool because its uses are virtually limitless. In short it is a very powerful multimedia programming language. The power of Lingo is the ability to give control over the presentation to the viewer. This is what makes the presentation interactive. If the programming concepts are thoroughly mastered, there are many amazing things that could be done to enhance the presentation.

Lingo programming language uses simple and direct English commands. Individuals with computer programming knowledge should benefit from this programming language as it uses commands and concepts that are alike. For instance it uses if....then statements and case statements. Lingo activity is divided into two basic categories, the event that triggers a script and the activity that the script performs. The events that trigger scripts are called *event handlers*. Director provides a scripting window that is capable of compiling the scripts in real time and other debugging tools.

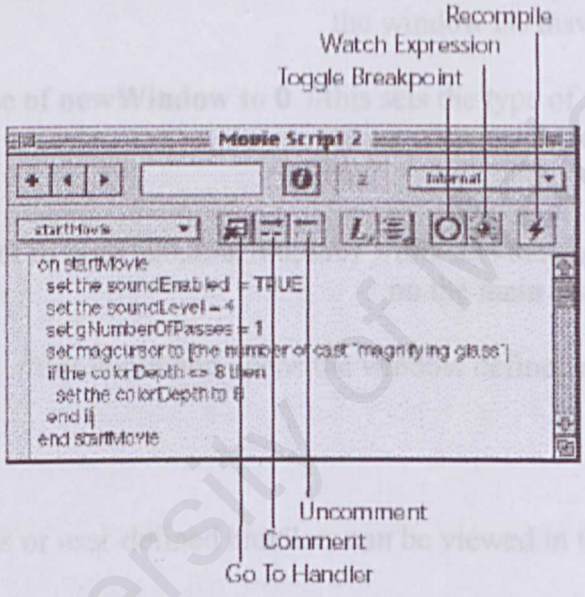


Figure 6.2: Lingo Script Window

In this project, there are a few elements that need to be custom programmed. Lingo programs that are fully programmed independently are called *user-defined handlers*. Below is one example of a user-defined handler used in the presentation to make a Movie In A Window (MIAW) popped up when the correct answer button for the multiple choice tutorial question is being triggered:

```
on mouseUp me      // event handler(when the mouse is clicked)
  global newWindow      //declaring a window variable
  set newWindow to window "Window1"     //declaring the name of the window as
                                        window1
  set the filename of newWindow to "correct.dir"  //set the variable newWindow to
                                                  the filename correct.dir
  set the title of newWindow to "Answer"  //the title to be displayed on the title bar of
                                          the window is Answer
  set the windowType of newWindow to 0  //this sets the type of window to a normal
                                        setting with just a close button on the title
                                        bar
  set newWindow.rect to rect(500,350,700,480)  //this sets the position of the window
                                               on the main window
  open newWindow     //this command opens the window defined above
end    //end of script
```

Other custom programs or user-defined handlers can be viewed in the Appendix section.

6.4 Debugging and Testing Programs

It is absolutely critical to test the project at all stages of development, debugging usually refers to unit testing that is similar to testing the functionality of the program. This is vital because early detection of program flaws help to improve the flow of the project. Debugging is compulsory if there are programming or scipts involved. This is because scripts do not always perform the required function the first time. The script often has an error in its syntax: many times a word is misspelled or a small part of the script is missing. Other times the script might work but does not produce the expected result. Mistakes or bugs always occur when writing Lingo, so a good programming habit is to allow enough time for debugging while developing multimedia presentations.

6.5 Basic Debugging

Debugging involves strategy and analysis, not a standard step-by-step procedure. This is the basic debugging approaches that have been applied to debug the codes.

Identifying the Problem

The first thing to do when debugging is to identify the problem. The techniques in identifying the problem that occurred within a script are simply by asking questions like. Is this button doing the right thing? Is the movie going to the wrong frame?

Locating the problem

Locating the bugs is not an easy task. It has to be done intricately on every line. These are the ways taken in this project to locate the problem:

- thinking backwards through the chain to identify where the unexpected started to happen.
- Use the Message window to trace which frames the movie goes through and the handlers that Lingo runs
- Determining what the Lingo should be doing and considering which part of the statement that are related to the problem.
- Does the problem occur only on certain computers and not others?
- Focusing on specific lines of Lingo by inserting a breakpoint-a point where Lingo pauses in a line. This gives the chance to analyze conditions at that point before Lingo proceeds.

Script error messages

The first debugging test occurs when the Script window is closed. If an error message appeared when attempting to close the Script window it means the script contains incorrect syntax. The message usually includes the line in which the problem was first detected. A question mark appears at the spot in the line where Director first found the problem.

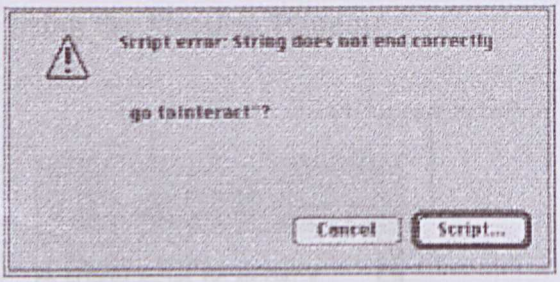


Figure 6.3: Script Error Message Window

The message "String does not end correctly" tells that the problem is related to the string in the line go to interact. When checked against the code actually the problem is that no beginning quotation mark precedes Interact

Syntax errors are probably the most common bug in Lingo. When script within this presentation fails, these are the debugging guidelines used to identify the bugs:

- Terms are checked to ensure correct spelling, spaces are in the correct places, and necessary punctuation is used.
- Check whether quotation marks surround the names of cast members, labels, and strings within the statement.
- All necessary parameters are present. References were made to the Lingo Dictionary to determine any additional parameters that have been left out.

6.6 Advanced Debugging

There were times when these basic debugging techniques were ineffective and more advanced techniques have to be applied. Director provides a Lingo debugging environment that helps to identify the problems in a more accurate method. These are the advanced debugging environments used:

Using the Message Window

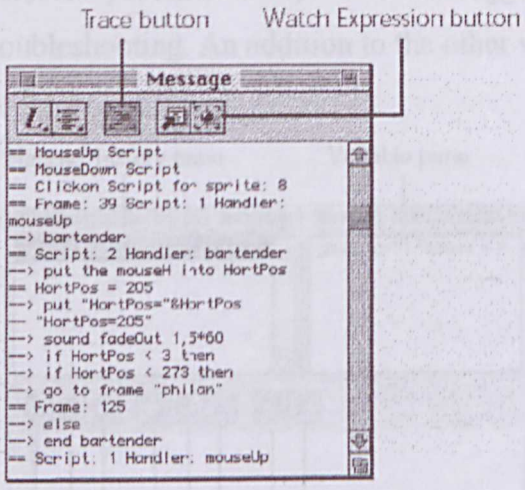


Figure 6.4: Lingo's Message Window

When the trace button is turned on, the Message window displays a record of what the movie does as it plays. This has help to track and observe every line of Lingo's scripting flow, which significantly identifies the real problem. The watch expression button assists in tracing the changes in variables. When a variable is selected its value will be displayed and updated as the movie plays.

Using the Watcher Window

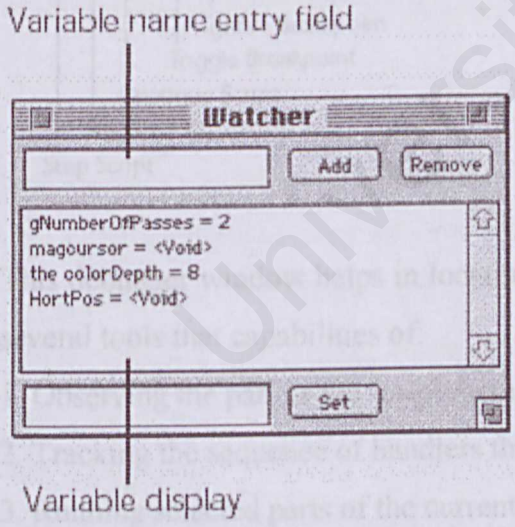


Figure 6.5: Lingo's Watcher Window

This window checks the value of variables and expressions while the movie plays by displaying the values in the Watcher window. Values in the Watcher window will be updated while stepping through the script lines.

Using the Debugger Window

Director provides many useful debugging tools to assist in fast and efficient bugs troubleshooting. An addition to the other windows there is the debugger window.

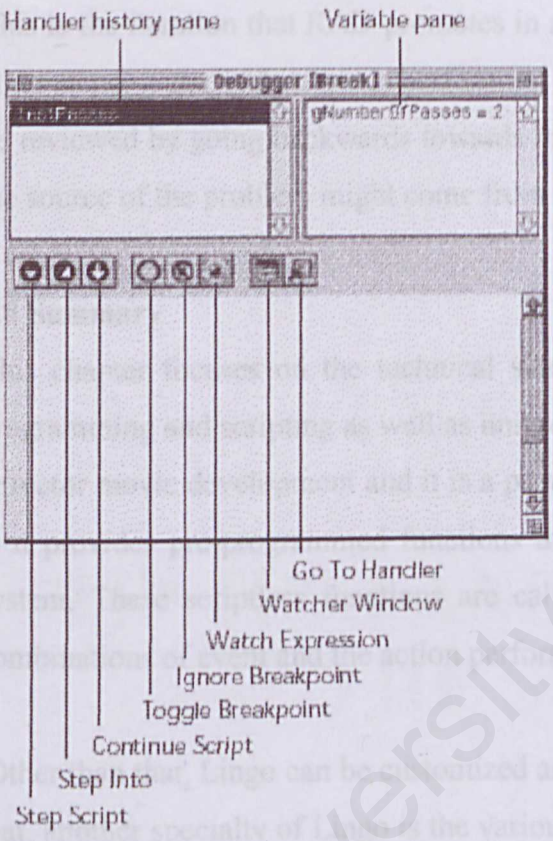


Figure 5.6: Lingo's Debugger Window

This debugger window helps in locating and correcting bugs in Lingo scripts. It includes several tools that capabilities of:

1. Observing the part of the script that includes the current line of Lingo
2. Tracking the sequence of handlers that were called before getting to the current handler
3. Running selected parts of the current handler
4. Displaying the value of any local variable, or properties related to the Lingo that were investigated
5. Opening related windows such as the Watcher window and the Script window

6.7 Integration Testing

When all individual components are working correctly without bugs and flaws, they are combined with the multimedia elements and turned into a working system. The movie will be played and observed. If any problem occurs, troubleshooting it will involve reversing to the scripts and the positioning of the elements within the movie timeframe. This is the iteration that RAD promotes in a system development. During testing or any other stage, if the system is not performing the correct functions, the system will have to be reviewed by going backwards towards the design phase and program writing phase as the source of the problem might come from these phases.

6.8 Summary

This chapter focuses on the technical side of the system implementation. It involves programming and scripting as well as unit testing. Lingo is the scripting language used in Director movie development and it is a powerful animation tool. The advantage of Lingo is it provides pre-programmed functions that are reusable many times through out the system. These scriptless functions are called behaviors and it is created by selecting combinations of event and the action performed if the event was triggered.

Other than that, Lingo can be customized and it is called user-defined behaviors. Besides that, another specialty of Lingo is the various debugging tools provided to better assist in script debugging and troubleshooting in a very fast and effective way. With the helpful debugging tools provided by Director, all bugs in the programs were successfully demolished and functioning, bugs-free programs are produced. When all programs are bugs-free it will be integrated with the multimedia element in the movie for integration test. Integration test ensures that the whole system can work smoothly without any unexpected problem occurring. If any problem does appear the programs or system design have to be reviewed.

CHAPTER 7

TESTING AND DELIVERING THE SYSTEM

Chapter 7 Testing and Delivering the System

7.1 Overview

Testing the system is very different from unit or program testing. Program testing gives the developer complete control over the testing process. However, when testing a system it is important to work with the customer. In this project case, it refers to the supervisor and moderator. In system testing, the objective is to ensure that the system does what the customer wants it to do in other words it fulfills the system requirements, functional requirements and non-functional requirements that were produced during the earlier stages in collaboration with the supervisor and moderator.

7.2 System Testing Approaches

As for this project there are four steps of approaches for system testing:

1. function testing
2. performance testing
3. acceptance testing
4. installation testing

Function Testing

System testing begins with function testing. In the previous chapter, the focus of the tests is on components and their interactions. In function testing, the focus is on system functionality and ignores the system structures and programs. Function testing compares the system's actual performance with its functional requirements. It tests how far has the system fulfils the functional requirements.

Performance Testing

System performance is measured against the performance objectives set by the customer as expressed in the non-functional requirements. So, the types of tests are determined by the kinds of non-functional requirements specified earlier in the system analysis and requirements stage.

Acceptance Testing

When function and performance testing are complete, we are convinced that the system meets all requirements specified during the initial stages of software development. The next step is to ask the users. In this case, the first users of this system are the project's supervisor and moderator.

Delivery Testing

This is done at the very end of all the testings. The main method for this type of testing is to play it on the delivery platform tat is the least capable machine the project is designed for. This machine is called the **lowest common denominator.** This is where the system's requirements such as the minimum RAM requirements and processor speed needed for the project are established. This is done by producing a copy of the software on a CD-ROM and testing it on different computers using different CD-ROM drives and speeds.

7.3 Test Results

Function testing: if referred to the functional requirements of the system there are six main requirements to be accomplished. The conclusion of the function test is the system has successfully fulfilled these six important requirements that were set by the user in the system analysis stage.

Performance testing: this test is conducted with reference to the non-functional requirements of the system. The system is user-friendly where it provides many helpful navigational buttons for the users convenience. Other than that, all information are presented with high clarity and in a very informative manner. The information is reliable, adequate and consistent. As for the interactivity, the system is fully interactive as for all the navigations and pace of learning depend on the users. Users are given many options in every part of the presentation as to where they would like to access or skip. Tutorials are provided with an interactive touch and it presents users with a highly responsive tutor. This way the interest can be kept alive for the rest of the presentation. Just a slight glitch

in the efficiency of the system, the playback performance is delayed as the system holds a huge content of data and mainly caused by the large wave files that need to be loaded into the system.

Acceptance testing: The acceptance testing was done on the 9th of September 2002, where the system was presented to the supervisor and moderator of the project. As overall the interface design were accepted as well as the functions of the system. The user requirements were met but certain issues were raised. The system should give tool tip guides as for every function of the buttons. But due to time constraint a Help function is not developed.

Delivery testing: At delivery testing stage, the lowest common denominator was identified and from there the hardware requirement specifications were developed.

Hardware Requirement Specifications

Processor with the speed of 366Mhz and above
Windows 95, 98, NT or later
CD-ROM Drive (40X or greater)
Soundblaster or compatible sound card
16MB RAM or greater (32MB recommended)
External or built-in speakers
8bit or greater Colour Display (640X480 resolution or greater)

7.4 Test Analysis

From the results of the test, an analysis is conducted in order to identify the system strengths and limitations as well as analyzing the extent of the project objectives that are successfully achieved. The system strengths are as follow:

- The system strength lies in the amount of information presented in this software. The information is conveyed through varieties of ways such as text explanations, narrations and line-by-line program explanations.

- The system promotes a highly effective learning concept to help increase the retention and comprehension rate of the users. The line-by-line execution simulation enhances understanding of program execution sequence, which is known to be the frequent problem faced by most students during the programming learning process.

- This system provides good navigation tools with many easy access buttons placed in every part of the presentation. Due to the enormous quantity of information to be conveyed there are many sections and sub-sections to be accessed. This condition can cause users to lose their direction within the presentation. As to avoid this many navigation buttons are placed in every part of the presentation. Besides serving for this purpose, the navigation and easy access buttons play the role in promoting the pleasure of self-learning pace to the users.

On the other hand, the testing process has also discovered several of the system's limitations. They are:

- Due to the huge quantity of information, the loading process of the system has slightly deteriorated. This is cause by the loading of wave files, which represents the narration element. For every page or slide that contains a long duration of explanation the appearance of the particular page or slide is delayed.

- Despite many easy-made navigation buttons provided, the Help function is not provided to guide users in identifying the function of each button. Users are forced to be independent and self-interpret in this matter. Somehow, the buttons use very symbolic graphics and most of the buttons have labels on them, which would help users to easily identify these buttons eventhough the tooltip Help function is not provided. An alternative way is the reference of the User Manual, which describes briefly on the function of the main buttons of this presentation.

- Another drawback is the incapability of the software to initialize the presentation as soon as the CD-ROM is inserted into the users CD-ROM drive. This is due to the non-existence of the autorun method within this software. Users would have to locate the Director's Projector file in the CD-ROM drive directory and double-click on the Projector icon to access to the presentation.

The test analysis has also provide means to determine the extent of objectives that this software has achieved at the delivery stage. These objectives are very important as they determine the final criteria the software will display to the users. And these objectives are vital to achieve as they represent the user requirements which means the more objectives accomplished the more accurate and close the software functions will be to the users expectations. Below are the conclusions of objectives that are successfully achieved:

1. This software is considered a multimedia learning package as it has successfully incorporated most of the important multimedia elements such as text, voice/narration, graphics and animation.
2. It provides an introduction on C programming language and an About window as to give a synopsis on the multimedia learning CD itself.

3. The software has included all the basic programming topic which covers C Fundamentals, Conditionals, Looping and Iteration, Arrays and Strings and Functions.
4. For every sub-topic contained in the main topic mentioned above additional explanations are provided to assist in in-depth knowledge acquiring of a particular sub-topic.
5. In order to promote an effective learning environment, line-by-line program execution simulation with individual line explanation and step-by-step output are implemented. Theses features is implemented in this software to assist beginners as well as C programmers to better comprehend the program reading process which has been claimed to be the main obstacle in understanding the C programming language. This method can improve the comprehension and retention rates of the users.
6. Another feature for an effective learning experience is the interactive tutorials. Interactive tutorials in this software response directly to the user's input and unlike text books where the answers are separated from the questions and this is inconvenience to the users. Besides the multiple choice questions, every topic is provided with a programming question and answers to help sharpen the programming skills of students as well as providing a platform to test their overall understanding.
7. The software successfully presents a user-friendly and intuitive graphical user interface for ease-of-use and better understanding of the presentation's flow. The interfaces provide many navigation buttons to allow fast and efficient access to the required sections of the presentation. More over, they are consistent and symbolic.
8. This multimedia presentation is highly interactive and it promotes self-paced learning concept to give users the flexibility and freedom of acquiring knowledge. The navigation buttons are capable to take users to any part of the presentation from anywhere within the learning package.

7.5 Delivery Process

After all development and testing are completed, it is time to prepare the project for delivery. The important part of this process is to determine that all necessary files and arrangement of files are present in the packaged project. A completed Director presentation must be converted into a file format that can be delivered to other computers for playback. The method used is to convert the movie into a projector, which is a self-contained application that can be activated by double-clicking. A projector contains all of the data that is necessary to play a movie back on a machine.

7.6 Summary

This is the final stage of the software development life-cycle. Testing the system is a non-tehnical experiment or evaluation done on the final system. It involves collaboration with users to ensure their priorities or known as their requirements are fulfilled before the system is delivered to them. Within this stage the system is required to complete four type of testing; function testing, performance testing, acceptance testing and delivery testing. Function testing is done against the functional requirements and performance testing is done against the system's non-functional requirements. Acceptance testing includes review from the users on the final system interface designs and functions. Lastly, is the delivery testing to determine the system's requirements specifications.

Result from the tests are analyzed and from there, the system's strengths and limitations will be significant. Eventually, these facts will help to determine the extent of objectives that the system was able to achieve. Overall, most of the system objectives set earlier during the system requirements and analysis stage are successfully established except for a few minor system limitations that have the possibilities of undergoing enhancement and improvement in the future.

References

1. Horton, Ivor 1997, *Beginning C*, 2nd Edition, Wrox Press Ltd., United Kingdom.

2. Schildt, Herbert 1997, *Teach Yourself C*, 3rd Edition, McGraw-Hill, California.

3. Schildt, Herbert 1995, *C: The Complete Reference*, 3rd Edition, McGraw-Hill, California.

4. Neuschotz, Nilson 2000, *Introduction to Director and Lingo: Multimedia and Internet Applications*, Prentice Hall, New Jersey.

5. Hooper, Simon 1999, *Authorware: An Introduction to Multimedia*, 2nd Edition, Prentice Hall, New Jersey.

6. Neo Mai, Ken Neo Tse Kian 1999, *The Multimedia Sourcebook: Multimedia Authoring and Web Publishing*, Volume 2, Meway Computec Sdn. Bhd., Malaysia.

7. Fluckinger, Francois 1995, *Understanding Networked Multimedia Applications and Technology*, Prentice Hall, England.

8. Pleeger, S.L 2001, *Software Engineering: Theory and Practice*, 2nd Edition, Prentice Hall.

9. A Learnered-Centered Approach to Multimedia Explanations, Vol. 2, no. 2, October, 2000, *http://imej.wfu.edu/articles/2000/2/02index.asp*

10. Interaction for Computer-Aided Learning, Vol. 2, no.1, April, 2000, *http://imej.wfu.edu/articles/2000/1/03/index.asp*

11. Marshall A.D, Programming in C:UNIX System Calls and Subroutines Using C, *http://www.cs.cf.ac.uk/Dave/C/node4.html*

12. Applying Multimedia to Education, *http://www.learningshop.co.uk/introduction.asp*

13. Brad A. Myers., *A Brief History of Human Computer Interaction Technology. ACM interactions.* Vol. 5, no. 2, March, 1998. pp. 44-54.

14. www.macromedia.com/support

15. www.graphicssoft.about.com/library/course/bllps5out.htm

Appendices

Appendix A : User Manual

How to Play the Movie from the CD-ROM

1. Insert the CD-ROM in the CD drive.
2. Open Window Explorer or click on the My Computer icon and locate the CD-ROM drive
3. Double click on the icon labeled Projector.

How to Use the Multimedia Learning Package: Learn to Program in C

Main button guides:

Sound Toggle Button
(turn sound/narration on and off)

About Button
(go to About page)

Exit Button
(to quit the program)

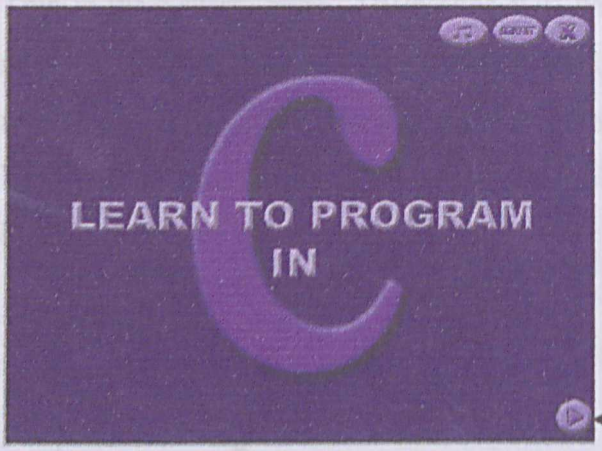
Home Button
(go to opening animation)

Back / Previous Button

Next/Forward Button

Main Button
(go to main option screen)

Several Example Pages from the Multimedia Presentation

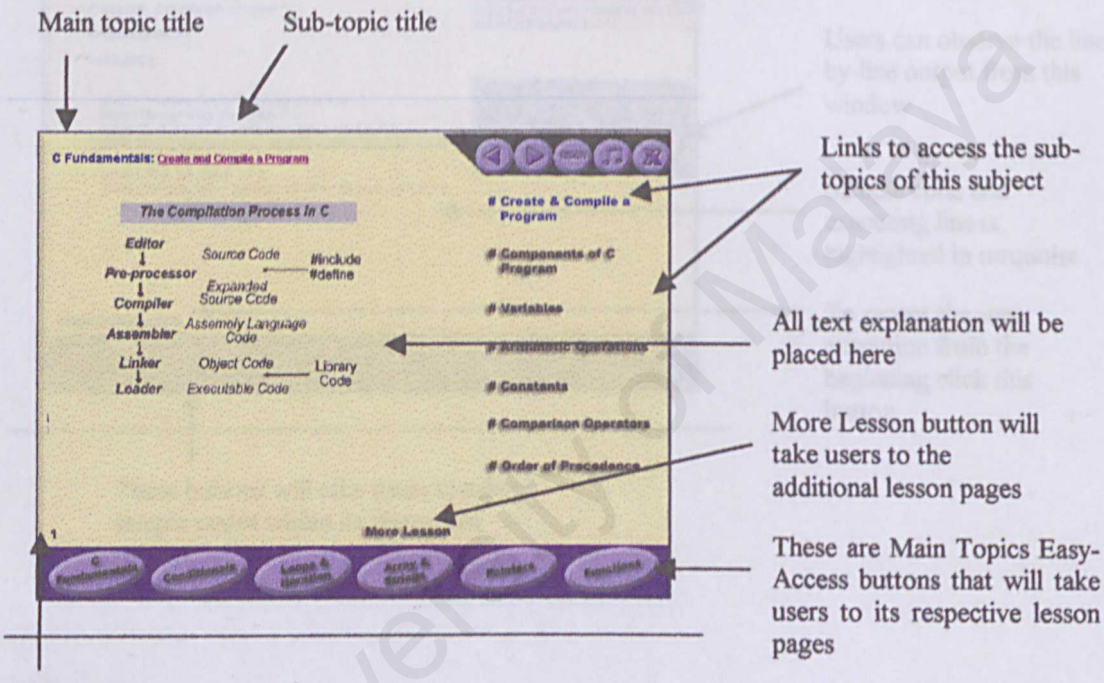


Click this Next/Previous button to enter the multimedia learning package

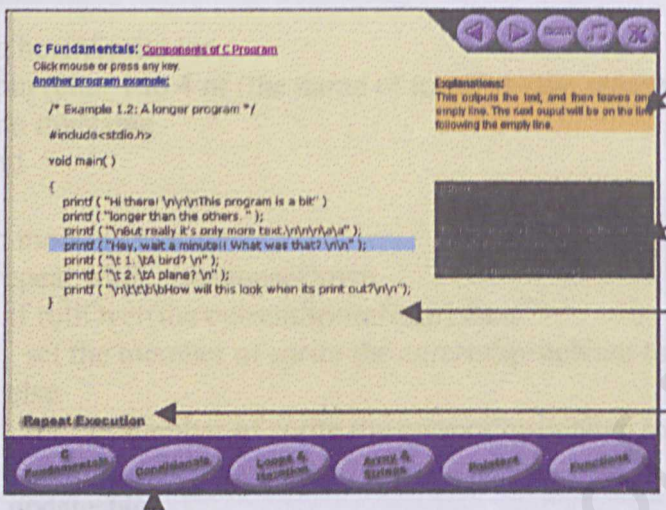
This is the Main Option window, it provides options to access the Introduction Module, Lessons Module, Sample Code Module and the Tutorial Module.

Click on any of this four links to access their respective modules

This is the Lesson page. All lessons will be explained in this window. One sub-topic might have more than one explanation page. Users can browse through the pages forward and backward using the Back and Next buttons. There will be page labels at the lower-left corner of the explanation section. At the last page of a sub-topic lesson the user will be able to see the More Lesson button.



This is the Sample Code page, where the line-by-line program execution simulation is placed. The line being executed will be highlighted in turquoise. There will be line-by-line code explanation and output on the right side of the screen.



Users can view the code explanation from this window

Users can observe the line-by-line output from this window

Sample code and executing line is highlighted in turquoise

To repeat the step execution from the beginning click this button

These buttons will take users to other sample codes within its respective topics

Appendix B: Sample Codes in Lingo Scripting

B1: Script for Rollover Button

```
global gWhichCast

property myName

on beginSprite me
  put (char 1 to 4 of (the name of member (the member of sprite the currentSpriteNum)))
into myName
end

on mouseDown me
  repeat while the mouseDown
    if rollOver(the currentSpriteNum) then
      set the member of sprite the currentSpriteNum to myName&"3"
    else
      set the member of sprite the currentSpriteNum to myName&"1"
    end if
    updatestage
  end repeat
end

on mouseWithin me
    set the member of sprite the currentSpriteNum to member
((myName&"2"),gWhichCast)
end

on mouseUp me
  set the member of sprite the currentSpriteNum to myName&"1"
  puppetSound 3, "button"
  updatestage
  if rollOver(the currentSpriteNum) = False then
    abort
    dontPassEvent
  end if
end

on mouseLeave me
  set the member of sprite the currentSpriteNum to myName&"1"

end
```

B2: Script to Create the Sound Toggle Button

```
property spriteNum, mySprite
property pPlayMem -- Play button member
property pPlayDownMem -- optional down state
property pStopMem -- Stop button member
property pStopDownMem -- optional down state
property pMusicMem -- sound or music to play
property pSoundChannel -- sound channel
property pPlayState -- toggle flag

on beginSprite me
  set mySprite = sprite(spriteNum)
  set pPlayState = soundBusy(pSoundChannel)
  mSwapMyMem
end

on mouseDown me
  if pPlayState then
    set the member of sprite mySprite = pPlayDownMem
  else
    set the member of sprite mySprite = pStopDownMem
  end if
end

on mouseUp me
  set pPlayState = NOT pPlayState
  if pPlayState = 1 then
    -- start playing
    puppetSound(pSoundChannel,pMusicMem)
  else
    -- stop playing
    puppetSound(pSoundChannel,pPlayState)
  end if
  mSwapMyMem
end

on mouseLeave me
  mSwapMyMem
end

on mouseUpOutSide me
  mSwapMyMem
end
```

```
on prepareFrame me

 -- swap members only if the sound has changed states
 -- ie: The music stopped
 if pPlayState <> soundBusy(pSoundChannel) then
  set pPlayState = soundBusy(pSoundChannel)
  mSwapMyMem
 end if
end

on mSwapMyMem me
 if pPlayState then
  -- music playing. Set member to stop member
  set the member of sprite mySprite = pStopMem
 else
  -- music stopped. Set member to play member
  set the member of mySprite = pPlayMem
 end if
end

on getPropertyDescriptionList me
 if the currentSpriteNum = 0 then exit
 set myMem = the member of sprite (the currentSpriteNum)
 set theNum = (the memberNum of myMem) + 1
 set theLib = the castLibNum of member myMem
 set defPlayDown = member theNum of castLib theLib
 set theNum = theNum + 1
 set defStopMem = member theNum of castLib theLib
 set theNum = theNum + 1
 set defStopDown = member theNum of castLib theLib
 set pList = [:]

 addProp(pList,#pPlayMem,[\
 #comment:"Play button:",\
 #format:#bitmap,\
 #default:myMem])

 addProp(pList,#pPlayDownMem,[\
 #comment:"Play button down:",\
 #format:#bitmap,\
 #default:defPlayDown])

 addProp(pList,#pStopMem,[\
 #comment:"Stop button:",\
 #format:#bitmap,\
```

```
  #default:member(defStopMem)])


  addProp(pList,#pStopDownMem,[\
  #comment:"Stop button down:",\
  #format:#bitmap,\
  #default:member(defStopDown)])

  addProp(pList,#pMusicMem,[\
  #comment:"Sound to play:",\
  #format:#sound,\
  #default:1])

  addProp(pList,#pSoundChannel,[\
  #comment:"Sound Channel:",\
  #format:#integer,\
  #default:1,\
  #range:[#min:1,#max:8]])

  return pList
end
```