

A Tutorial on Authorware

by

David R. DeVaux

Project and Report submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

APPROVED:

Dr. Edward Fox, Chair

Dr. John Carroll

Dr. Mary Beth Rosson

April 25, 1996

Blacksburg, Virginia

Keywords: Authorware, Authoring, Multimedia, CBT

Copyright 1996 David R. DeVaux

A TUTORIAL ON AUTHORWARE

by

David DeVaux

Dr. Edward Fox, Chairman

Computer Science

Abstract

Authorware is an icon-based multimedia authoring tool which allows the rapid development of complex interactive multimedia projects, particularly courseware and kiosk applications, for both the Macintosh and Microsoft Windows operating systems. This project consists of three main elements: interactive courseware, written in Authorware, which teaches the student basic concepts involved in Authorware programming, and demonstrates the function of each of the icons used to program in Authorware; a tutorial through which students are given the opportunity to use Authorware to incorporate various media elements, including written audio, graphics, video, and text, into their own interactive courseware; and various course materials, including a statement of objectives, study questions, and quiz questions. These materials were developed for use in the Virginia Tech Computer Science course CS4984 (*Multimedia, Hypertext, and Information Access*) as part of the unit on *System and Application Construction*.

Acknowledgment

I would like to thank Dr. Edward Fox for his guidance and support, Dr. Carroll and Dr. Rosson for being on my committee, Mary Miller for allowing me to use Interactive Design and Development's facilities and resources, and, last but not least, my wife Jeannette for her patience and understanding.

Table of Contents

ABSTRACT	ii
ACKNOWLEDGMENT	iii
TABLE OF CONTENTS	iv
TABLE OF FIGURES	v
1. INTRODUCTION	1
1.1 AUTHORWARE.....	1
1.2 MOTIVATION FOR PROJECT.....	6
2. RELATED LITERATURE AND WORKS	8
3. COURSEWARE DEVELOPED FOR CS4984	9
3.1 INTERACTIVE AUTHORWARE INTRODUCTION.....	10
3.1.1 Overview.....	11
3.1.2 Icons.....	13
3.2 TUTORIAL	14
3.3 OTHER MATERIALS	21
4. EVALUATION OF COMPLETED STUDENT TUTORIALS	22
5. EVALUATION OF STUDENT ANSWERS TO QUIZ QUESTIONS	24
6. USE OF TUTORIAL AT SNE	27
7. CONCLUSIONS	29
BIBLIOGRAPHY	30
APPENDIX A - TUTORIAL	31
APPENDIX B - UNIT MATERIALS	71
APPENDIX C - QUIZ QUESTIONS	73
APPENDIX D - QUIZ ANSWERS	75
VITA	79

Table of Figures

FIGURE 1 : AUTHORWARE ICON PALETTE	3
FIGURE 2 : EXAMPLE FLOWLINE.....	3
FIGURE 3 : EXAMPLE DESIGN WINDOW.....	4
FIGURE 4 : EXAMPLE PRESENTATION WINDOW	4
FIGURE 5 : OVERVIEW, PAGE 1.....	11
FIGURE 6 : OVERVIEW, PAGE 2.....	12
FIGURE 7 : ICON INTRODUCTION.....	13
FIGURE 8 : INITIAL TUTORIAL CODE (AS GIVEN TO THE STUDENT).....	16
FIGURE 9 : (COMPLETED) TUTORIAL CODE.....	16
FIGURE 10 : AUTHORWARE LIBRARY.....	17
FIGURE 11 : SAMPLE SCREEN FROM COMPLETED TUTORIAL	19
FIGURE 12 : CODE FOR PRECEDING SCREEN.....	19

1. Introduction

1.1 Authorware

Authorware is an icon-based authoring environment developed by Macromedia, Inc. [Macromedia, Inc. 1992b] The version used for this project, 2.0, was released in 1992. A newer version, 3.0, was released in 1995. This version introduces two new icons, the navigation and framework icons, which the Authorware programmer can utilize to create complex program flow structures and subroutine-like constructs. Hypertext capability was also added with this version. The next release, 3.5, will provide the ability to create courseware which can be used over the World Wide Web, if the user has the appropriate plug-in for his/her Web browser. However, Authorware 2.0 was chosen for this project because this is the version currently available to students taking the CS4984 class. The course materials developed for this project are certainly applicable to newer versions of Authorware; they simply do not cover the new features introduced in Authorware 3.0, and those that will be introduced in Authorware 3.5.

Authorware can be used to quickly and easily develop interactive multimedia software incorporating text, high resolution graphics, audio, animations, and video. Authorware is available for both the Macintosh and Microsoft Window operating systems. In fact, Authorware programs created on Macintosh can be easily ported to the Windows environment.

Authorware programs consist of icons, each of which performs some function (such as displaying a graphic, providing program execution branching, or playing a movie, among others), much like each statement in a text-based programming language. Icons are dragged from an icon palette (Figure 1) to a "flowline" (Figure 2) to create a program - when someone runs the program, execution proceeds from the top of the flowline to the bottom (unless special "decision" icons or "interaction" icons affect program execution, much like if-then statements and loop statements affect program execution flow in conventional programming languages.)

Other authoring systems employ iconic programming (e.g., IconAuthor [AimTech Corporation 1995]) or other approaches to simplify the task of multimedia application development. Macromedia also distributes Director, which uses a score/timeline approach coupled with "actors" properly "staged" [Macromedia, Inc. 1994]. On the other hand,

HyperCard [Apple Computer, Inc. 1993] uses an approach to application development which employs “cards” on which “buttons” and “fields” are placed, each of which may have an associated “script”. And, ToolBook [Asymetrix Corporation 1989] applications consist of a group of “books”, each designed for a particular purpose. Authorware was chosen for this project due to the author’s experience with it, because Virginia Tech has a site license to it, and because its hardware requirements are widely met.

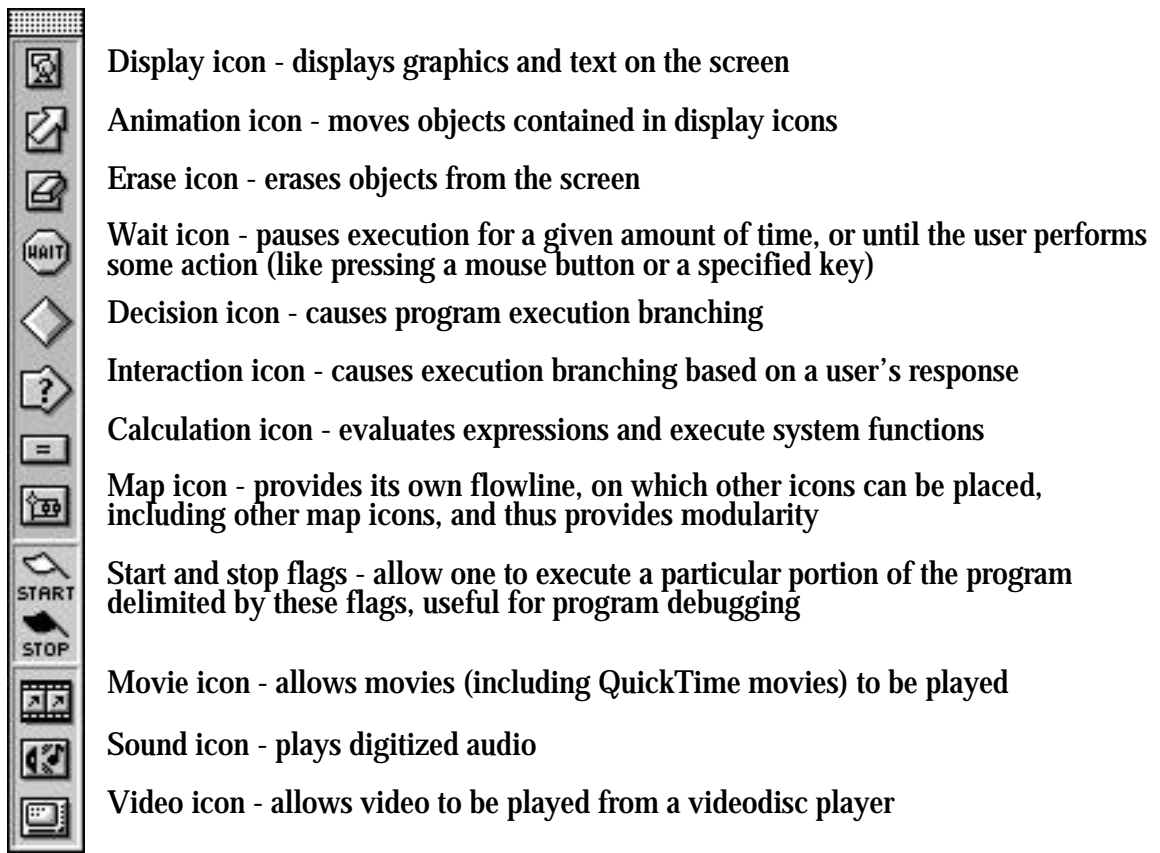


Figure 1 : Authorware Icon Palette

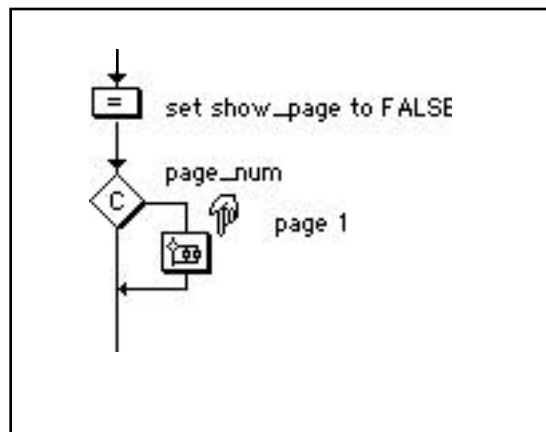


Figure 2 : Example Flowline

There are two main kinds of windows in Authorware - Design (Figure 3) and Presentation (Figure 4). Design windows contain the flowlines on which icons are placed to create a program. The Presentation window is the window in which the program executes.

Below are shown an example Design Window and the Presentation Window as it would appear if the icons in the Design Window were executed.

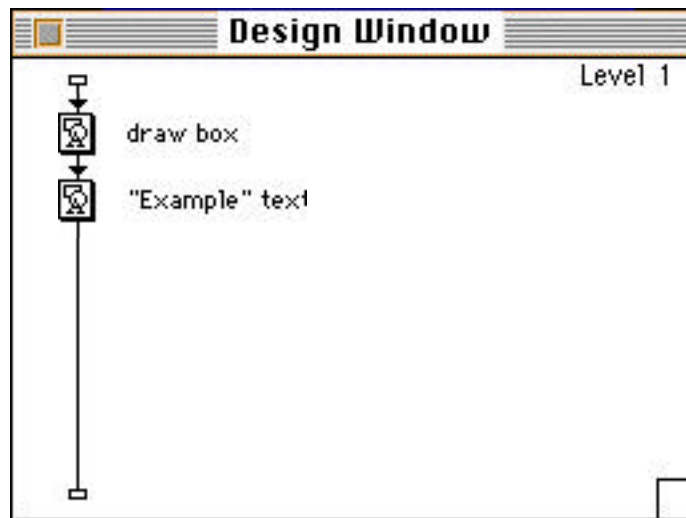


Figure 3 : Example Design Window

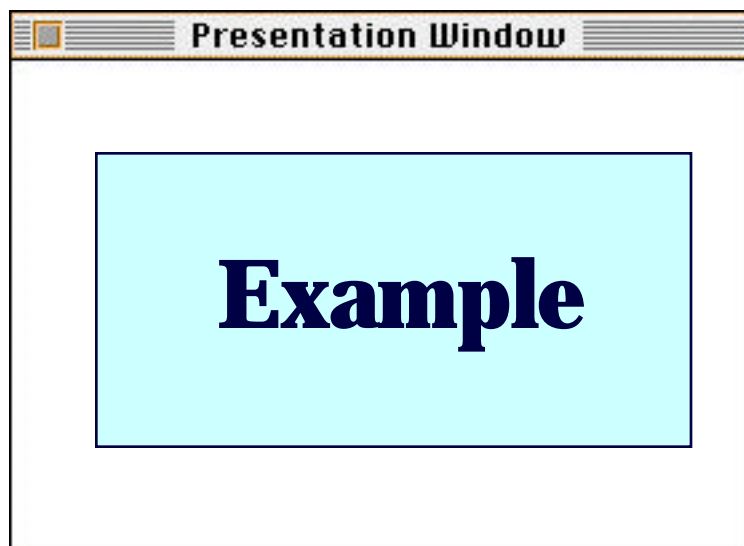


Figure 4 : Example Presentation Window

Authorware allows the programmer to switch back and forth between the Design and Presentation Windows at will. This is an extremely useful feature, as it eliminates the need for lengthy compilation; one can immediately ensure that a new piece of code works properly, and then make any necessary changes.

Daniel Greenberg, in a review of Authorware written for *Digital Video* magazine, refers to the ability of Authorware to “deliver tremendous multimedia power via a simple graphical interface” [Greenberg 1995], and indeed, the ease with which Authorware can be used to create multimedia software makes it an extremely valuable tool. The icon-based programming paradigm employed makes programming intuitive - if one wants to play a movie, one simply drags a movie icon onto the flowline, and sets the appropriate attributes in a dialog box. Thus, non-programmers can create courseware which incorporates a variety of media elements. However, Authorware also does include icons used for program control branching, and allows “script” to be included in calculation icons. So, programmers can easily reproduce familiar constructs in Authorware that they usually employ in languages like Pascal and C. If-then statements, case statements, and loops can all be constructed with Authorware’s icons. And, the “flow-line” form of an Authorware program parallels the form of a flowchart, an object familiar to many.

In addition. Authorware provides the ability to easily create complex user interactions. “Hot-spots” can be created on the screen just by drawing them with the mouse. These hot-spots can be activated by clicking the mouse button. Or, if a touch screen is used, they can be activated by touching them directly on the screen. (This requires no special programming.) Push buttons, text entry areas, draggable objects, and timed responses can all be easily created. Furthermore, Authorware provides means for tracking and evaluating responses to these interactions.

Therefore, Authorware is quickly gaining in popularity as a tool for multimedia development, particularly for computer based training (CBT) and kiosk applications. In fact, *NewMedia* magazine awarded Authorware 3.0 the 1996 HyperAward for Computer-Based Training Software, referring to Authorware as “the market-leading computer-based training tool” [Waring, Hood 1996].

Today’s competitive software market demands rapid application development. Authorware provides high - level constructs for multimedia functions, such as displaying graphics or playing video. Concurrency and timing functions are provided as well in

Authorware, which can be difficult or impossible to code in traditional languages, but are critical in multimedia software. Authorware allows multimedia developers to concentrate more on the design and content of the work to be developed, rather than numerous low-level programming details.

1.2 Motivation for Project

Multimedia is a quickly growing field, as is computer based training (CBT). In fact, the number of multimedia CD-ROMs shipped worldwide reached 53.9 million in 1994, up from 16.5 million in 1993 [Gregarus, Wong 1995]. And, the amount of sales generated from multimedia CD-ROM titles is expected to grow from \$584 million in 1994 to about \$1.5 billion in 1996, and to about \$3.9 billion in 1997, according to Forrester Research [Manasco, Albinak, and Bross 1995]. Thus, it is important that current and future computer science curricula provide students the opportunity to be exposed to the concepts and tools involved in multimedia production. There are many steps involved in multimedia production, including graphic creation and editing; audio creation, recording/digitizing, and editing; and video production, capturing, and editing. However, the computer science student, through his/her education, is uniquely qualified to perform the programming necessary to integrate these elements into multimedia software/courseware.

Multimedia programming can be, and is often, performed in “general-purpose” programming languages like C and C++. Languages like these offer the benefit of extreme flexibility. In addition, programs written in these types of languages are usually efficient - both in terms of speed and memory usage. However, the production of multimedia with these types of languages is often tedious and time-consuming, as the programmer often has to deal with timing issues and somewhat cryptic low-level drive calls.

Presentation programs, like Microsoft’s PowerPoint, or Aldus’ Persuasion, are very high-level, and are relatively easy for the non-programmer to use. Such programs are excellent for building “slide-show” multimedia presentations. However, they usually do not allow for complex user interactions, external file handling, program branching (conditionals and loops), and other features often necessary for developing multimedia applications.

Authoring languages, on the other hand, offer a compromise between these two extremes. They provide high-level constructs for integrating multimedia elements. In addition, they often have features reminiscent of “general-purpose” programming languages, like branching constructs and user variables. Thus, while authoring languages do not offer all the flexibility of “lower-level” general purpose languages, they are much more flexible than presentation packages. And, application development is usually much more rapid with authoring languages than with languages like C and C++. If necessary, languages like C and C++ can be used to implement functions not available in the authoring package.

Authorware is a particularly powerful authoring language. It offers a programming-language like structure - looping constructs and conditionals are provided, modularity can be achieved through the use of “map” icons and separate program modules, and system functions can be called and expressions evaluated within calculation icons. Yet, the author can place media elements within a program simply by dragging the appropriate icons to the correct place on the flowline, setting a few properties, and (if applicable) positioning the object on the screen simply by dragging it. Creating complex user interactions also is easily and rapidly achieved. External functions (XCMD’s and XFCN’s on the Macintosh, and DLL’s and UCD’s in Windows) can be called to perform low-level functions that cannot be written in Authorware. Therefore, Authorware is a particularly suitable topic of study as an environment for rapidly developing complex multimedia applications.

Thus, under the direction of Dr. Edward Fox, I have developed course materials for the CS4984 course to demonstrate the ease with which interactive multimedia can be produced with Authorware, and help the student gain a basic understanding of the icon-based method of Authorware programming (as opposed to the statement-based method to which they are more likely to be accustomed).

2. Related Literature and Works

Since Authorware is a relatively new language, and has not yet received a wide-spread following, very little literature exists regarding its use. Thus, the *User's Guide* [1992] and *Using Variables and Functions* [1992] books provided by Macromedia as part of the Authorware package are still the chief reference for Authorware programmers.

In addition, Macromedia provides a *Tutorial* [1992] book with Authorware that teaches beginning programmers about Authorware by guiding them through the construction of courseware that instructs the user about the basics of 35mm photography. (While this tutorial is a valuable instructional aid, it is too involved to use as part of a one week introductory overview of Authorware. Thus, it was not used for the CS 4984 class.)

Joe Ganci of Universal Systems, Inc. has written a three-volume set of books on Authorware [Ganci 1995] covering Authorware internal system functions and variables, as well as the use of external functions (XCMD's and DLL's) with Authorware.

Perhaps one of the most valuable sources of information regarding Authorware, however, is the Authorware listserv. Here, users of Authorware at all levels of experience share their own discoveries and ideas about Authorware programming. In addition, Peter Arien and Paul Hamilton, both employees of Macromedia, often answer questions and give invaluable advice here as well. This list may be subscribed to by sending e-mail with "SUBSCRIBE AWARE" in the body of the message to listserv@cc1.kuleuven.ac.be. Messages to the list should be sent to aware@cc1.kuleuven.ac.be.

3. Courseware Developed for CS4984

The materials developed consist of the following:

1. Interactive courseware, written in Authorware, which teaches the student basic concepts involved in Authorware programming, and demonstrates the function of each of the icons used to program in Authorware.
2. A tutorial through which students are given the opportunity to use Authorware to incorporate various media elements, including audio, graphics, video, and text, into their own interactive courseware.
3. Various other course materials, including objectives, study questions, and quiz questions.

These materials were developed for CS4984 (*Multimedia, Hypertext, and Information Access*), a senior level class in the Computer Science department at Virginia Tech. This is currently a Special Study course that will become a regular course (CS4624) after 1996. The course catalog description for CS4984 is given below [Fox 1996a]:

Introduces the architectures, concepts, data, hardware, methods, models, software, standards, structures, technologies, and issues involved with: multimedia information and systems; hypertext and hypermedia; networked information; electronic publishing; virtual reality; and information access. Students will learn how to capture, represent, store, compress, manipulate, interact with, and present: text, drawings, still images, animations, audio and video. They will work with video conferencing, authoring systems, and digital libraries.

CS4984 follows the Personalized System of Instruction, a teaching plan developed by Fred S. Keller, J. Gilmour Sherman, and others. The main characteristics of this system are given below [Keller 1968]:

- go-at-your-own-pace: so students can proceed according to their abilities, interests, and personal schedules
- unit-perfection requirement: which means students must demonstrate mastery of a unit before proceeding to other units
- lectures and demonstrations for motivation: instead of for communication of critical information
- stress on the written word for teacher-student communication: which helps develop comprehension and expression skills
- tutoring/proctoring: which allows repeats on exams, enhanced personal-social interaction, and personalized instruction

Following this model, the CS4984 class consists of six distinct units, each intended to cover a particular topic. The Authorware materials were developed for the Application Construction unit, and comprise approximately half of the unit. The main objectives for the Application Construction unit are as follows [Fox 1996b]:

- effectively author hypertext, hypermedia or multimedia works
- critique software and hardware systems in this area, considering functionality, interface, quality of presentation, and other important criteria

The content of each of the materials developed, as well as their instructional value, is discussed below.

3.1 Interactive Authorware Introduction

The first part of the course materials on Authorware consists of an interactive introduction to Authorware, written in Authorware. While this introduction is by no means a complete description of Authorware, it serves to expose students to the basic concepts involved in Authorware programming, as well as to motivate them by showing how Authorware can be used to integrate various media elements (animations, movies, graphics, sounds) into an interactive presentation. In addition, the source for this program can be made available to students, as an example of Authorware programming. The introduction consists of two parts: 1) a brief overview of the components of Authorware (the Design and Presentation windows, and the icon palette) and the icon-based programming paradigm, and 2) a description of each of the Authorware icons.

3.1.1 Overview

The Overview consists of two pages of information. The first (Figure 5) introduces the student to the Authorware icon palette, the flowline, and the icon-based programming paradigm.

Authorware Introduction: Overview Overview Icons Exit 9:33 PM

Icons in Authorware

- Authorware is an "icon-based" language - i.e., a program consists of a set of icons, each of which performs some function, much like each statement in a text-based programming language performs some function.
- Icons are dragged from an icon palette to a "flowline" to create a program - when someone runs the program, execution proceeds from the top of the flowline to the bottom (unless special "decision" icons or "interaction" icons affect program execution, much like if-then statements and loop statements effect program execution flow in conventional programming languages)

The icon palette

Example: Part of a Flowline

Previous page 1 of 2 Next

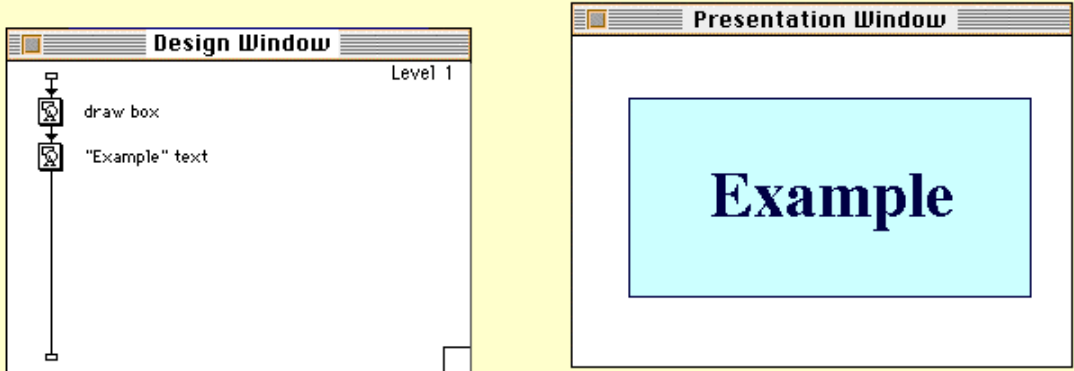
Figure 5 : Overview, page 1

The second page (Figure 6) introduces the Design and Presentation windows.

Authorware Introduction: Overview Overview Icons Exit 2:33 PM

Design and Presentation Windows

- There are two main kinds of windows in Authorware - Design and Presentation. Design windows contain the flowlines on which icons are placed to create a program. The Presentation window is the window in which the program executes.
- Below are shown an example Design Window and the Presentation Window as it would appear if the icons in the Design Window were executed.



Example Design Window

Example Presentation Window (the result of executing the icons in the Design Window)

Previous page 2 of 2 Next

Figure 6 : Overview, page 2

3.1.2 Icons

The “Icons” button on the control bar, shown in Figure 6, takes the student to a screen where he/she can learn the function of each of the Authorware icons. When the student selects an icon from the palette, the icon moves to the top of the screen and its title appears. Then, a sample piece of Authorware code using this icon appears in the Design window. Finally, this code is executed in the Presentation window. The first icon in all the code samples is always a display icon which contains text explaining the function of the icon currently selected. The state of this screen after the animation icon is selected (Figure 7) is shown below as an example.

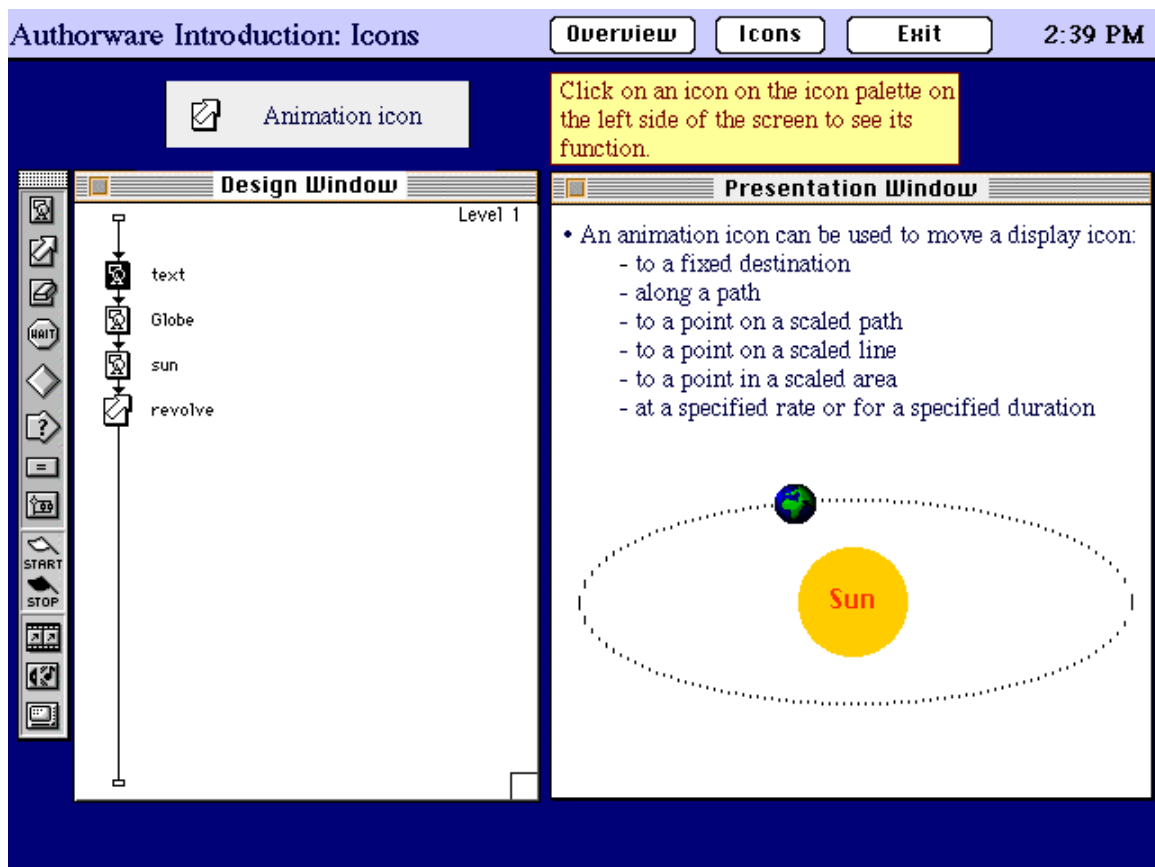


Figure 7 : Icon introduction

3.2 Tutorial

The interactive introduction exposes students to Authorware programming by example. The tutorial, however, allows students to create their own Authorware program. The goal in developing the tutorial was to produce a short course that would not take an inordinate amount of time to complete (not more than an hour or two), while still accomplishing the following instructional objectives:

1. exposing the student to the icon-based programming paradigm
2. demonstrating the potential for creating complex user interactions in Authorware
3. showing how Authorware allows various types of media to be integrated into interactive courseware

Thus, the tutorial was designed with these objectives in mind. The tutorial guides the student in developing a short course which consists of six “pages” of material. The student must construct a “page-turning” interaction with controls for moving forward and back through the pages, and a mechanism for displaying the current page number. A page-turning type of interaction was chosen because:

1. Content is often presented in pages within multimedia software. In general, people are comfortable with turning the page to see more information, as this parallels the way they use a book, a real-world object familiar to most people. Thus, one of the first things a multimedia programmer often learns to do in a new authoring language is to create pages of content that a user can navigate through.
2. A page turning interaction is not a particularly complex type of user interaction. Creating buttons that allow the user to leaf through pages of content is not as complex as creating interactions which respond when the user drags an object to a particular location, or types in a particular string of text into a text entry field. Yet, the creation of a page-turning interaction is sufficient to allow students to experience creating a useful user interaction in Authorware.

A file with a few of the necessary icons already in it is given to the student. The student then must add the necessary icons to complete the program (and set the necessary properties for these icons, as well as creating the content of the calculation icons). The

program as given to the student (Figure 8), as well as the completed program (Figure 9), are shown below. (Note: The content of each of the icons and/or their properties are not shown here - just enough detail to show the basic program structure is given.)

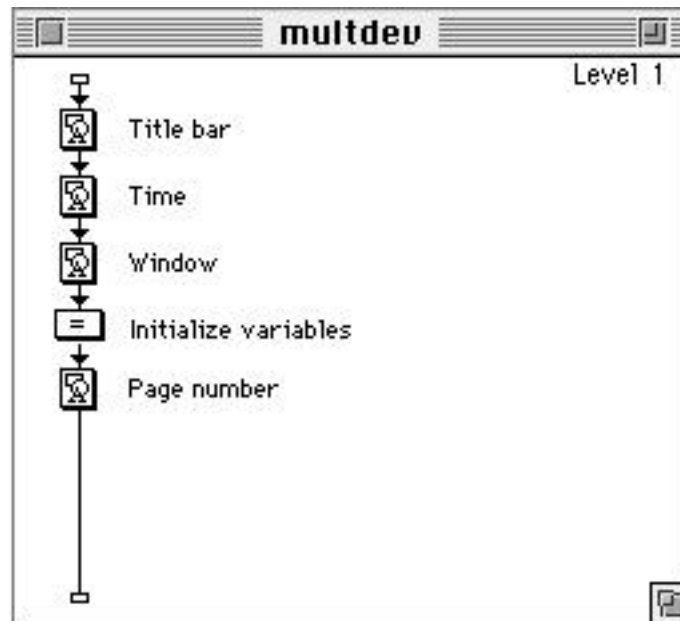


Figure 8 : Initial Tutorial Code (As Given to the Student)

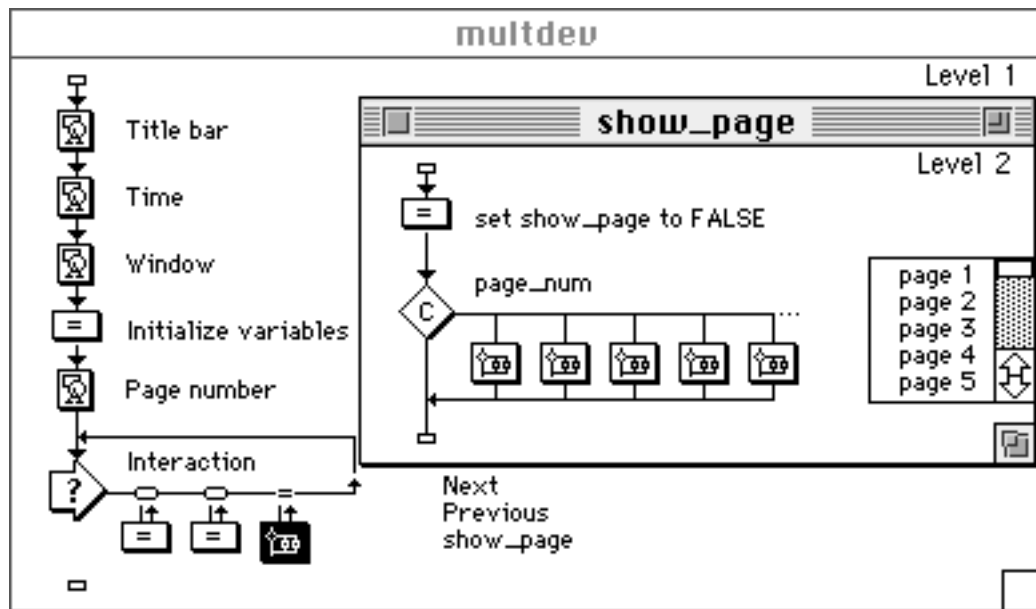


Figure 9 : (Completed) Tutorial Code

In addition, the student must take media elements (graphics, sound, and movie icons) from an Authorware library (Figure 10) and place them on the flowline in order to assemble each page. (It was decided to create the media elements for the students because they might not have the necessary resources to do so, it would be too time intensive, and the goal was to demonstrate Authorware's ability to integrate media, not create it.)

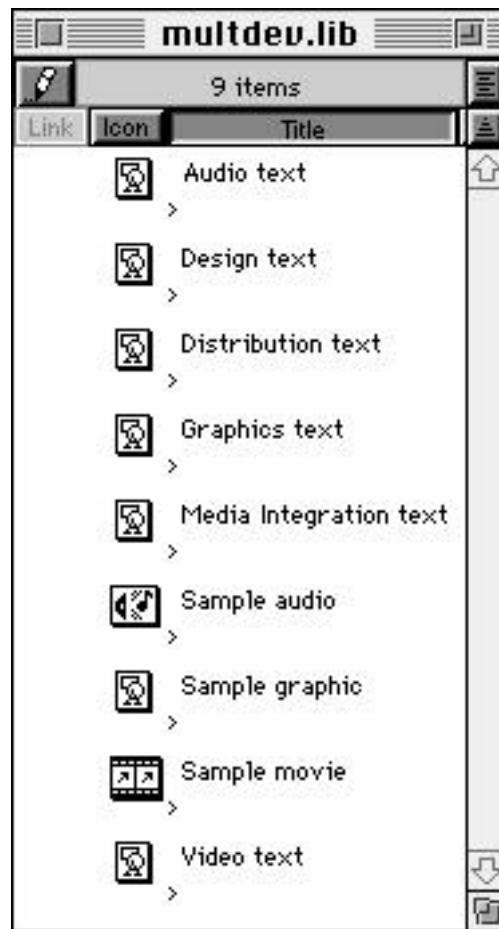


Figure 10 : Authorware Library

The topic for the courseware developed in the tutorial concerns the steps involved in multimedia development. This topic was chosen for two reasons:

1. The steps involved in multimedia development is a topic which is relevant to the class for which the tutorial was designed. Since a major focus of the CS4984 class is multimedia, it is appropriate for the students taking this class to create courseware which concerns multimedia development.
2. The author works for a multimedia design and development company, and thus has had significant experience developing multimedia projects. Therefore, the author was qualified to write the content used for the courseware created via completion of the tutorial.

The text (like the other media elements) is provided for the student, so they can concentrate on learning about Authorware programming, instead of content development. Below is shown a sample screen from the completed Tutorial (Figure 11), as well as the code (Figure 12) which creates the content (i.e., the text and the picture of the computer) on this particular screen. Note that the two icons in the sample code (Figure 11) have titles which are in italics. This indicates that they are actually “links” to items stored in the library.

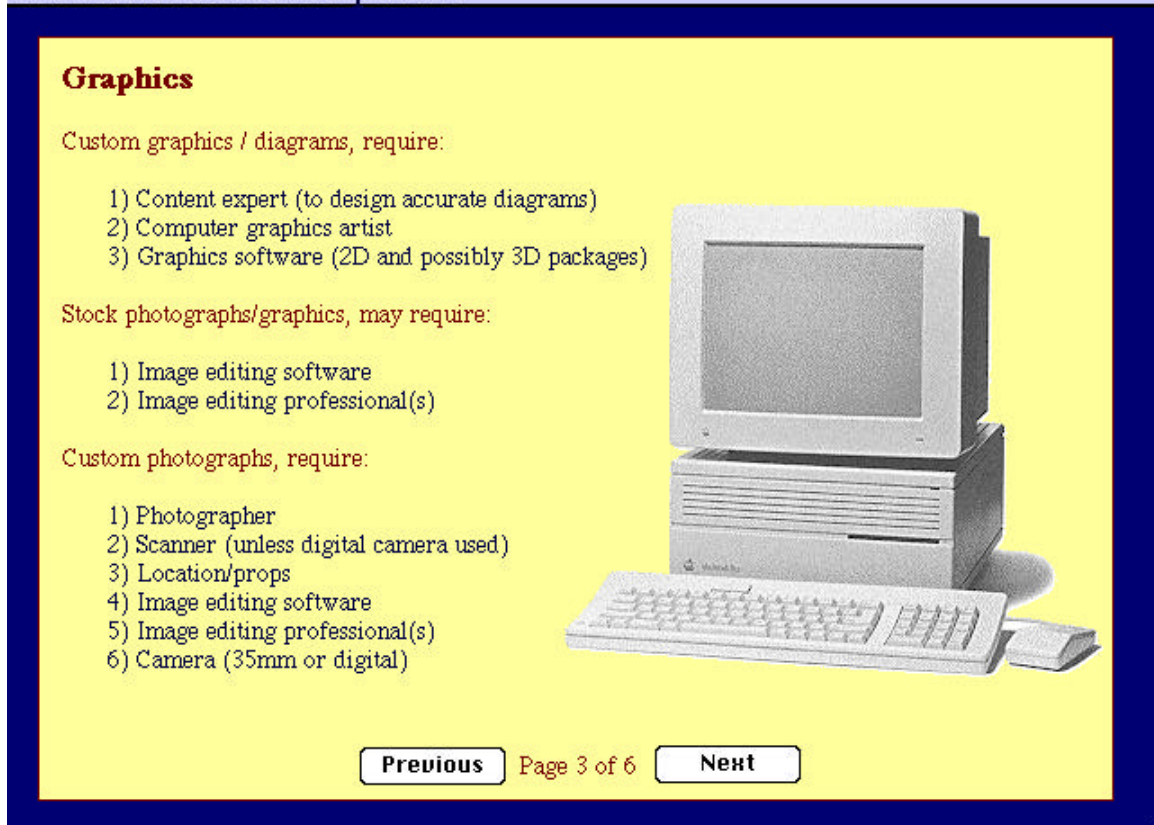


Figure 11 : Sample Screen from Completed Tutorial

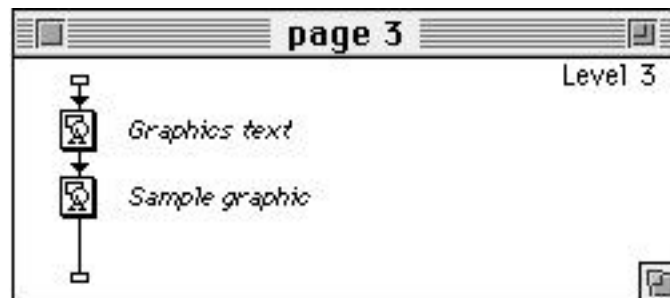


Figure 12 : Code for Preceding Screen

When the student finishes the tutorial, he/she will have done the following:

1. Used all but a few of the Authorware icons
2. Built a simple user interaction
3. Integrated graphics, sound, text, and a movie into their courseware
4. Learned how to obtain items from an Authorware library

Thus, by completing the tutorial, a student will be exposed to various features of Authorware, and many of the concepts involved in Authorware programming. In addition, the student should then be able to compare and contrast icon-based programming with other types of programming they have done in the past.

3.3 Other Materials

It was necessary to develop other materials to create a complete Authorware instructional unit. The materials developed are listed below.

1. A “**unit document**” (Appendix B) consisting of
 - **Issues** - basically, why Authorware is an appropriate topic of study
 - **Objectives** - the information the student should learn as a result of completing this unit
 - **Exercises** - in this case, running the introductory Authorware program and learning the information presented, as well as completing the tutorial
 - **Study Questions** - questions that the student should use as practice questions when studying for the quiz

2. **Quiz questions** (Appendix C) - The quiz questions are intended to test the student’s knowledge of Authorware, as learned through using the introductory courseware and completing the tutorial. Three sets of quiz questions were written; these are given in Appendix C. The answers to these quiz questions are given in Appendix D. Note, however, that only the first set of questions was used for the CS4984 class in 1995.

4. Evaluation of Completed Student Tutorials

There were twenty-nine disks turned in for the tutorial assignment (note that the students were allowed to collaborate, so that in some cases, two students turned in one disk.) The students just turned in the completed Authorware program - not the library or external QuickTime movie that was used. Thus, to test their programs, I copied them to a folder which had these files in it.

The tutorial was intended to *teach* the students basic concepts in Authorware, not assess their knowledge. Thus, it was very straightforward and unambiguous. As a result, the students did not have any major difficulties completing the tutorial.

Four students (or pairs of students) moved some of the background graphics that were provided to them. Two moved the title bar, and two moved the clock displayed in the top right corner of the screen. One of these also moved the box in which the content of the course is displayed. Most likely, these graphics were moved accidentally - Authorware allows the author to move objects about in the Presentation Window at will - thus it would be an easy mistake for someone to accidentally move objects about. The students were not given instructions on moving objects in the Presentation Window because this information was not necessary for completion of the tutorial. Therefore, the students that accidentally moved the graphics did not know how to put them back in their original locations. However, these small difficulties should not have had any significant effect on the knowledge the students gained, since their completed programs were otherwise correct.

One student used a different type of pushbutton for the “Next” button than was specified. His program worked correctly. He probably accidentally clicked the radio button corresponding to this type of pushbutton in the response type window. This error should not have had any significant effect on the knowledge the student gained from completion of the tutorial.

One student had a corrupted Authorware file - when an attempt is made to open his file, an error message is issued - “The file ‘multdev.lib’ can’t be opened because the file is in use elsewhere (-49).” Then, an additional message is issued - “unable to post aliased icon due to error in loading data.” After these messages are issued, the file is opened, but most of the links to the library “multdev.lib” are broken. The library, however, works

correctly with all of the other student programs. Thus, one can reasonably assume that the student's program file was damaged in some manner.

5. Evaluation of Student Answers to Quiz Questions

Dr. Fox received 40 quiz responses from the students via e-mail. Most of the students did well answering the quiz questions; however, some students did have difficulty with one or two of the questions. The quiz questions given are in Appendix C, Set 1; the corresponding answers are in Appendix D, Set 1.

The first question consisted of three multiple choice sub-questions. All but four of the 40 students gave correct answers for all of the sub-questions. Two of these four students gave the same incorrect response for the first sub-question : “Which of the following icons allows text to be shown on the screen?” Both of these students indicated that both “Text icon” and “Display icon” were correct, when, in fact, only “Display icon” is correct. Of the other two students, one incorrectly indicated that a Video icon is used to display QuickTime movies; the correct icon is a Movie icon. The other indicated that “the window in which Authorware program execution takes place” is the Design Window, when in fact, it is the Presentation Window.

The objective of this question was to assess knowledge of the terminology involved in Authorware as learned from the tutorial. Thirty-six of the 40 students correctly answered all three sub-questions, and the remaining four students each missed only one sub-question. Thus, the students seem to have learned the terminology involved in Authorware well.

The next question asked the students to name the Authorware icon which provides capability for modularity and nesting, and explain how the icon accomplishes these functions. The correct icon is a Map icon. All but two of the students named the correct icon; one gave “Interaction icon” as an answer, and the other answered “Grouping icon”. Thirty-three students indicated that each Map icon provides its own flowline, and 34 explained that map icons may be “nested” within one another.

The objective of this question was to discover if the students could correctly identify the Authorware icon which provides modularity and nesting, and describe how this icon provides these capabilities. (This material should have been learned via completion of the tutorial.) The success that most of the students had in answering this question indicates that they did understand that the Map icon provides modularity and nesting capabilities, and

that it does so by providing its own flowline and allowing other icons, including Map icons, to be placed on it.

The final question asked the student to “describe an Authorware program that causes one of three graphics (named graphic 1, graphic 2, graphic 3) to be displayed based on the value of a variable.” Students’ responses to this questions varied significantly more than the other two questions. Twenty-three of the students gave correct responses, the same as or similar to the answer given in Appendix F, which involves using a Decision icon set to “Calculated Path”, with three display icons attached to it. Another student seemed to have the right idea, in that he first described the use of a “Calculated Path” Decision icon. However, he then specified that the calculation icons would have to be attached to this Decision icon and set to “conditional”, with each Display icon attached to one of these Calculation icons. It seems that he confused the Conditional Response type of interaction response with calculation icons.

Ten of the students gave a response modeled after the tutorial; technically, this type of response may be correct, since it does involve the use of a variable to display the appropriate graphic. The only difference is that a user must press “Next” and “Previous” buttons which increment or decrement the value of the variable used in a “Calculated Path” Decision icon to display the right graphic. Two of these students whose response seemed to follow the tutorial, however, described Authorware code which would not function properly. The other eight students described properly functioning programs.

One student also gave a correct response that was similar to the tutorial-modeled responses; however, instead of using “Next” and “Previous” buttons, he described the use of three buttons, one for each graphic, which would cause the value of the variable to be set appropriately.

One student gave an interesting, mostly correct, answer involving the use of an Interaction icon with three Map icons attached, each of which contains a Display icon for one of the three graphics. He specified that the response type for each Map icon would be set to “Conditional”, with each condition set to test for equality of the variable to the number for the given graphic. The only problem with this answer is that the responses would need to be set to “Automatch”; however, given the level of exposure these students had to Authorware, this answer is a good one.

The four remaining students gave incorrect responses. In particular, one student seemed to mistakenly think that Display icons could be attached to Calculation icons, and attempted to put together a program which involved three Calculation icons, each of which tested for a particular value of the variable, with a Display icon attached to each one. Another possible explanation is that he meant to use Decision icons set to “Calculated Path” instead of Calculation icons, in which case his program would be more correct. The other three students gave answers that were unclear and did not describe anything resembling a correct program.

The objective of this question was to assess the student's understanding of how Authorware icons can be put together to create a working program. Most of the students gave a correctly working program as a response. However, only 23 out of 40 students gave an answer that matched the expected response. This indicates that the question is probably unclear; in the future, it should be explained that the correct graphic should be displayed based on the value of a variable without requiring intervention from the user. Another conclusion is that when students with some background in computing (i.e., who have satisfied the course prerequisites) are given an open programming assignment, especially in a new language, there is a not insignificant probability that they will provide quiz responses adopting approaches different from that expected by the designer of the courseware utilized.

6. Use of Tutorial at SNE

In addition to its use in the CS4984 class, the Authorware tutorial also was used for a multimedia workshop held for the 1995 Society for Nutrition Education (SNE) conference in Washington D.C., and conducted by Interactive Design and Development (IDD), a multimedia company in Blacksburg, VA. This workshop was intended to introduce the participants to various aspects of multimedia software production, including video and audio capture and compression, graphic creation and editing, and, finally, the use of a multimedia authoring package to bring the elements together in order to create interactive courseware. Since the author is employed by IDD, and had recently finished writing the Authorware tutorial, IDD decided to use it to introduce authoring software at the workshop.

The Authorware portion of the workshop was held in a computer lab setting, with ten Macintosh computers located around the periphery of the room. Twenty people participated in the Authorware tutorial session, two to each computer. One hour was set aside for this session. An instructor (the author) led the workshop participants through the tutorial by completing it at the front of the room on a computer with an LCD projection panel attached. In this way, the participants could “follow along” with the instructor as they completed the tutorial. Five other IDD employees walked about the room assisting individuals with any questions or problems they had.

All the workshop participants were eventually able to successfully complete the tutorial. The instructor took one hour to complete the tutorial, explaining each step as he proceeded through it. Some (four or five) participants worked ahead of the instructor, and were able to complete the tutorial in 45 to 50 minutes. Most of the rest of the participants finished at the same time as or immediately after the instructor did. Two participants, however, took about 70 minutes to complete the tutorial, since they lagged behind the instructor due to errors which caused them delay. IDD employees were able to assist these people in correcting their errors and completing the tutorial.

The degree of individual assistance the participants required from IDD employees varied greatly, mostly due to the varying degree of experience with computers. Some of the less experienced people at the workshop asked many questions, both from the instructor and the assistants, while others needed little or no help, and completed the entire tutorial by

following the tutorial document. It soon became evident that, unlike with the senior-level Computer Science students, some degree of assistance is necessary when administering the tutorial to people with little or no computer experience. However, the tutorial did achieve the desired goal in this particular setting, which was to simply expose the workshop participants to the use of authoring software to create interactive courseware. The intent was not to teach them to become multimedia programmers, but just to give them an idea of what is possible.

While no formal analysis or review of this workshop was performed, the participants did seem to enjoy the tutorial, and derive benefit from it. Several (about five) participants showed an interest in obtaining Authorware to use in their work as educators. Others mentioned projects that they were interested in developing, and asked IDD employees questions regarding the use of multimedia in their projects. Most of the workshop participants had never experimented with multimedia before the workshop; the Authorware tutorial, along with the other activities in the workshop, gave them ideas about how they could employ multimedia technology to teach people about nutrition.

7. Conclusions

The Authorware tutorial has proved to be an effective method for introducing people with varying computer and programming skills to Authorware. The CS4984 students showed, via their success completing the tutorial and answering the quiz questions, that they had grasped the basic concepts involved in Authorware programming. Most of the users at the SNE workshop had never used an authoring package; the instructor, by leading them through the Authorware tutorial, showed them how they could use an authoring package to combine various media elements into interactive multimedia courseware.

When used to teach people with little or no programming and/or general computer experience (such as the SNE workshop group), it is advisable to have an instructor(s) present to answer questions. In such a situation, it is also useful to have someone “lead” the class by going through the tutorial at the front of the class, so that the students may follow along. However students with some programming experience (like the CS4984 students) seem to need no assistance in completing the tutorial, as was the intent when developing it.

Bibliography

AimTech Corporation. *IconAuthor 6.0 Home Page*. 1995:
[<http://www.aimtech.com/Iia.htm>].

Apple Computer, Inc. *HyperCard Reference Manual*. Cupertino: Apple Computer, Inc., 1993.

Asymetrix Corporation. *Using Toolbook - A Guide to Building and Working with Books*. Bellevue: Asymetrix Corporation, 1989.

Fox, Edward. *CS4624 Catalog Info*. 1996:
[http://ei.cs.vt.edu/~mm/s96/sspace/Catalog_Info_124.html].

Fox, Edward. *CS4624 Objectives - AC*. 1996:
[http://ei.cs.vt.edu/~mm/s96/sspace/Objectives_676.html].

Ganci, Joseph. *Authorware System Variables and Functions*. 3 vols. Woodbridge: Successful Multimedia, 1995.

Greenberg, Daniel. "Authorware 3.0." *Digital Video* vol 3, no 12, page 18, 20. Dec. 1995.

Gregarus, Fred and Sandy Wong. *Multimedia Status*. Palo Alto: Fenwick and West, 1995: [<http://www.batnet.com/oikoumene/mmediastatus.html>].

Keller, Fred. "Goodbye, teacher ..." *Journal of Applied Behavioral Analysis* 1(1), pages 79 - 89. Spring 1968.

Manasco, Britton, Paige Albinak, and David Bross. "CD-ROM Developers Said to Lack Creativity; Sales Not Expected to Take Off Until 1997." *Multimedia Week* page 8. 22 May 1996.

Spiers, Joseph. "Were Holiday Sales All That Bad?" *Fortune* vol 133, no 3, [<http://pathfinder.com/@@wJK18SLIXgAAQC2k/fortune/magazine/1996/960205/holiday.html>]. Feb. 1996.

Macromedia, Inc., *Tutorial*. San Francisco: Macromedia, Inc., 1992.

Macromedia, Inc., *User's Guide*. San Francisco: Macromedia, Inc., 1992.

Macromedia, Inc., *Using Director*. San Francisco: Macromedia, Inc., 1994.

Macromedia, Inc., *Using Variables and Functions*. San Francisco: Macromedia, Inc., 1992.

Waring, Becky and Phil Hood. "The NewMedia 1996 HyperAwards." *NewMedia* vol 6, no 1, pages 54 - 56, 58. Jan. 1996.

Appendix A - Tutorial

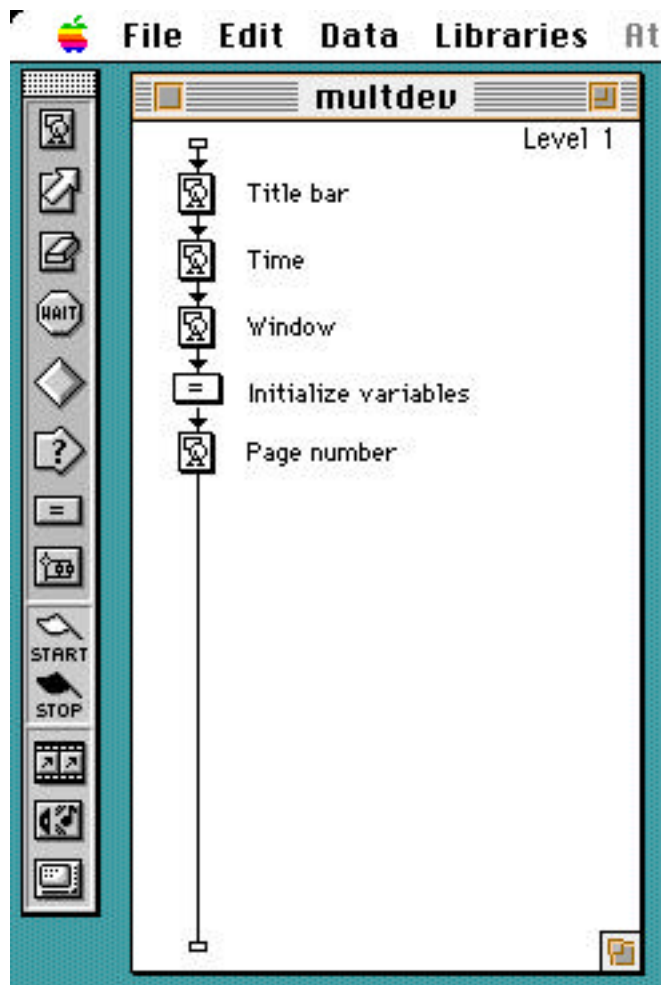
1.0 Getting Started

1.1 Starting Authorware

(1) Please launch Authorware Professional from the At Ease menu.

(2) Go to the File menu and Open file multdev in the Authorware Demo folder, which is inside the CS4984 folder, that is on the Assignments disk.

You will see a Design Window entitled “multdev”, as well as the Authorware icon palette.



The Design Window is where you “program” in Authorware. Authorware applications are created by selecting icons from the palette (shown to the left in the illustration) and placing them on the flowline (the line that goes from the top to the bottom of the Design Window). Different icons perform different functions. In general, icons are executed along the flowline from top to bottom, much like the statements in a traditional programming language are executed from the top to the bottom of the file in which they are located. (However, Authorware, like a programming language, contains branching constructs which affect the program flow.) Note that there are icons already present on the flowline:



The first icon, entitled “Titlebar” displays a grey bar with the title “Multimedia Development” at the top of the screen. This type of icon is referred to as a Display Icon.



The second icon, entitled “Time”, displays the time at the upper left corner of the screen.



The third icon, entitled “Window”, displays a window within which the content of this presentation will be displayed.



The fourth icon, entitled “Initialize variables”, is a Calculation Icon. This icon contains lines of code which initialize the various variables used in this program.



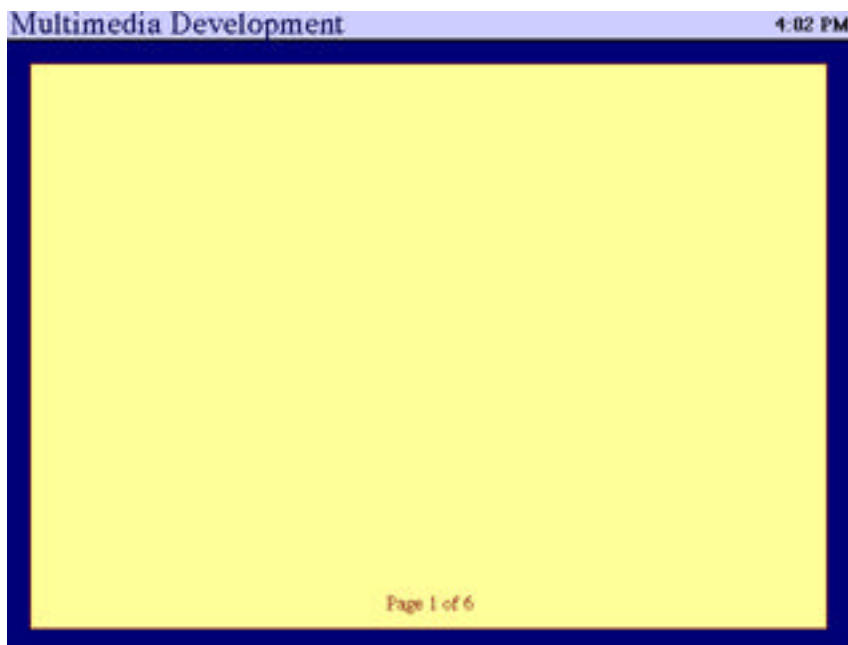
The fifth icon, a Display Icon entitled “Page number”, displays the current page number, as well as the total number of pages.

1.2 Running the Program

Let’s see what our program does thus far. Authorware lets you run your program at any time to see if it works the way you have intended.

1) To run the program, select “Run” from the “Try It” menu (or press Apple R).

You should see a screen that looks like this:



This is the Presentation Window. It is the window in which your Authorware program will run. In addition, you can make modifications to the way your program will appear to the user here (draw and import graphics contained in Display Icons, move and resize buttons and graphics, etc.).

2) To return to the Design Window, press “Command J”.

Now, you will see the Design Window again. Authorware lets you switch back and forth between the Design and Presentation Windows at will. This is an extremely useful feature, as it eliminates the need for lengthy compilation; you can immediately see if your program is working properly, and then go and make any necessary changes.

In addition, through the use of the Presentation Window, Authorware lets you make certain changes (draw graphics, move graphics, movies, and buttons, etc.) much more easily than in a traditional programming language.


2.0 Creating a Page Turning Interaction

In this lesson, we'll create pushbuttons that allow the user to turn pages in the course we're going to develop.

Note: You should be sure to periodically save your work - to do so at any time, simply select “Save” from the “File” menu (when you see the Design Window - since we have a full screen Presentation Window, the menus aren't available when the Presentation Window is visible). Use your diskette for the “save” of each new version.

2.1 The Interaction Icon

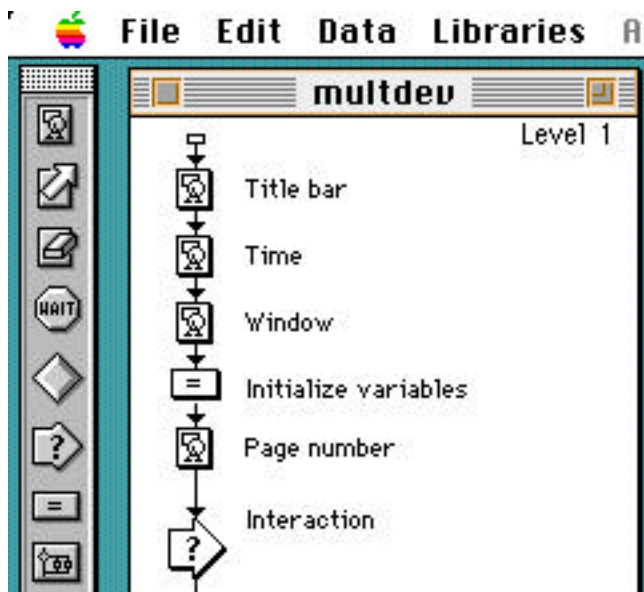
1) Drag an Interaction Icon from the icon palette onto the flowline (below the other icons on the flowline).

An Interaction Icon looks like this . Interaction Icons are used to create user interactions (allowing the user to press buttons, enter text, etc.) Notice that to the immediate right of the icon is the highlighted word “Untitled”. This is the icon title.

Note that if you ever find that a window is not big enough, you may resize it. The control for resizing a window may be found in the lower right-hand corner of the window. Simply click on this control and, while holding the mouse button down, drag the mouse. The window size will change as you drag. Release the mouse when the window is the size you want.

2) Type “Interaction”

Note: If the icon is not highlighted, you will have to select it first before giving it a title. To do so, just click either on the icon or its title. Then you can type “Interaction” to title it. When you are finished, you should see something similar to that shown below:




We’ve created the basic framework for all Authorware interactions. However, now that we’ve done that, we actually have to set up things for the user to respond to (pushbuttons, text entry areas, etc.), and allow action to be taken based on the user’s input. To do this, we attach items called “responses” to an Interaction Icon.

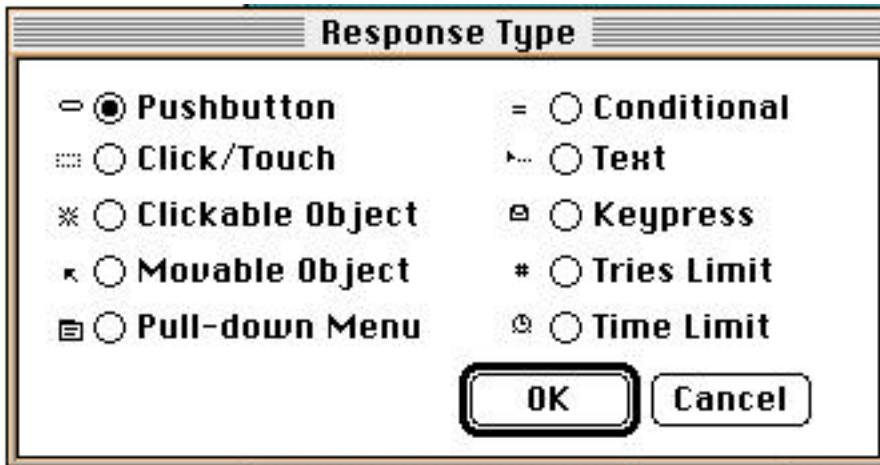
2.2 Creating a “Next” Pushbutton Response

Now, we’ll create a pushbutton that the user can press to turn back one page.

1) Drag a Calculation Icon from the Authorware icon palette to the immediate right of the Interaction Icon.

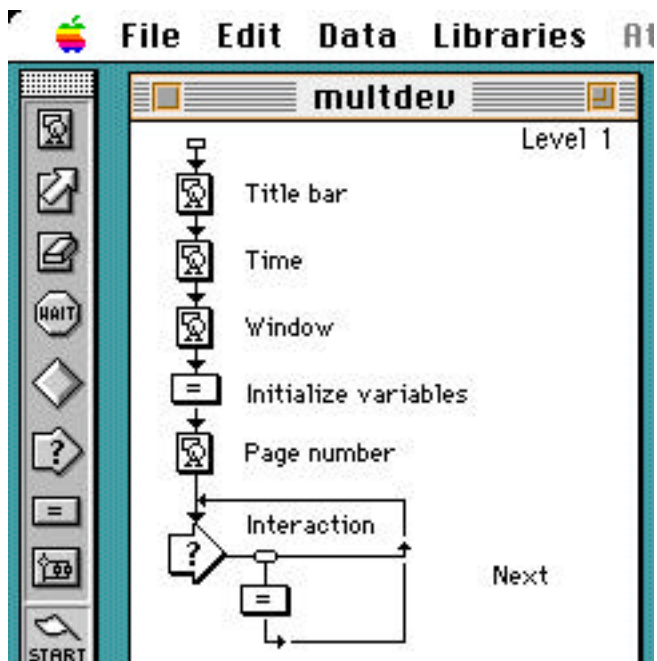
A Calculation Icon looks like this . We’ll actually specify the calculations this icon will perform later.

A window like that shown below will be displayed. This window lets you choose what “response” type you want the user to interact with. We want to create a pushbutton, so select this option, then click the “OK” button. (Actually, this option should be selected by default, in which case just clicking the “OK” button will suffice.)



2) Entitle this icon “Next”

Note: If the icon is not highlighted, you will have to select it first before giving it a title. To do so, just click either on the icon or its title. Then you can type “Next” to title it. Note that the titles for icons attached to Interaction Icons are located to the right of the interaction (Interaction Icon + attached icons). When you are finished, you should see something similar to that shown below:



Notice the arrows going from and to the Interaction Icon, and from the Calculation Icon attached to the Interaction Icon. These arrows show the “flow” as affected by the Interaction Icon. When an Interaction Icon is encountered, program flow pauses until some user response occurs (mouse is moved, text is typed, mouse button is clicked, etc.). Then, program flow moves along the flowline attached to the Interaction Icon from left to right until the user’s action “matches” the response found.


When our Interaction Icon is executed, flow will pause until the user performs an action. Then, flow will move from the Interaction Icon to the right along its flowline. When the Next pushbutton response is encountered, a test is performed to see if the user pressed the Next button. If so, flow will travel to the Calculation Icon attached to the Next response. Then, execution will bypass the rest of the Interaction Icon flowline and return to the Interaction Icon.

If the user did not press Next, flow would proceed to the end of the Interaction Icon’s flowline and return to the Interaction Icon.

2.3 Changing Response Options

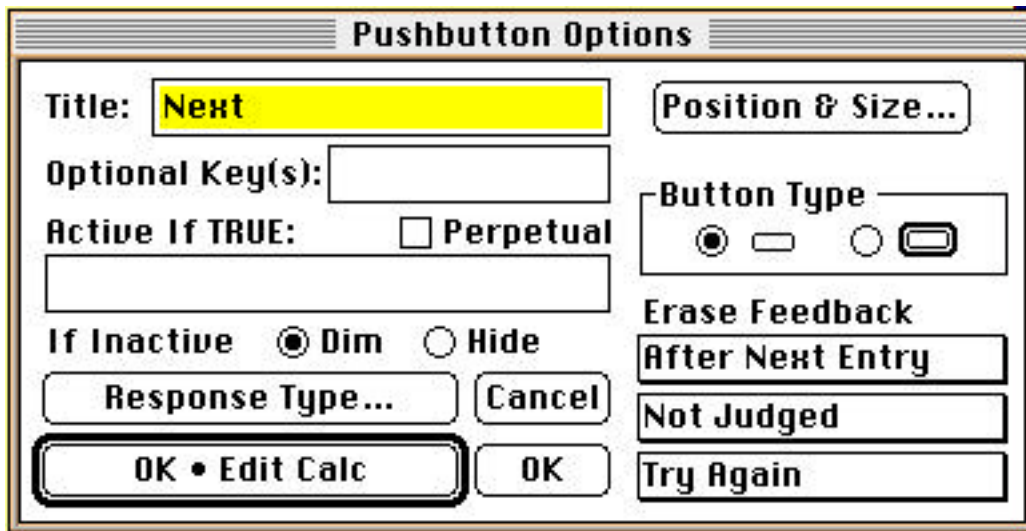
We’ve created a button. But, we haven’t placed it properly. And, if the button is pressed, we will want program flow to go back to the Interaction Icon’s flowline, instead of bypassing it and returning directly to the Interaction Icon (we’ll see why later). We will now see how to change these properties.

1) Open (double click on) the Next pushbutton response icon

The Next pushbutton response icon is the small one located on the Interaction Icon's flowline that looks like this . When you have opened the icon, you should see the Presentation Window open. It should look like it did before, except now there is a button labeled "Next" that looks like this:



as well as a window that looks like this:



This window allows you to set various properties for this (button) response. And, the button can be resized and moved about.

2) Change the execution flow from "Try Again" to "Continue"

In the bottom right hand corner of the "Pushbutton Options" window is a pull down menu. The currently selected option is "Try Again." This means that after the icon attached to this response is executed, flow will bypass the Interaction Icon's flowline, as discussed earlier. Change the currently selected option to "Continue". (Just click on "Try Again" and hold down the mouse button. A menu will appear. While holding down the mouse button, select "Continue" from the menu, and then let go. You will see that the currently selected option has changed to "Continue".) Choosing "Continue" will cause program execution to continue back to the Interaction Icon's flowline and continue along it to the right after the icon attached to this response is executed.

3) In the text box labeled "Active if TRUE" type the following:

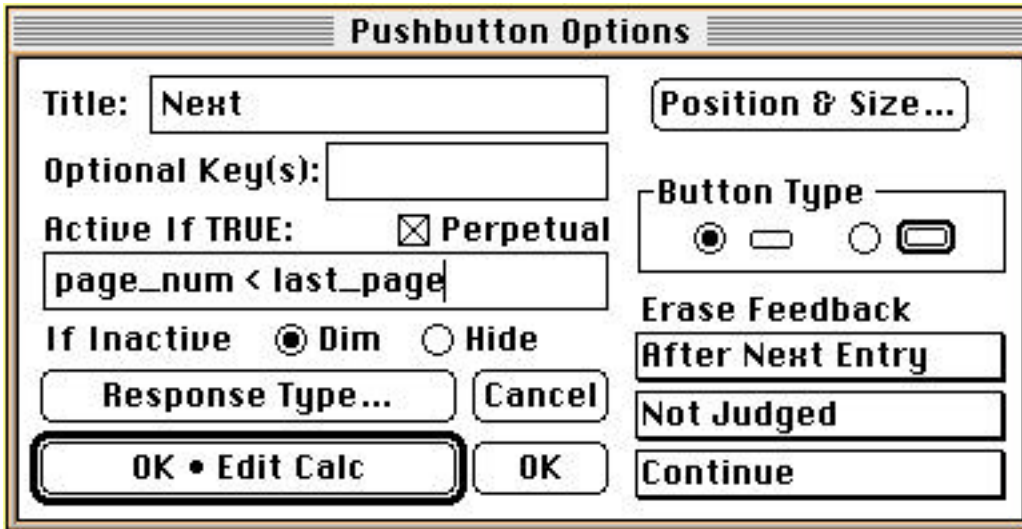
page_num < last_page

This will cause the Next button to be active only if the variable "page_num", which stores the number of the current page, is less than the value of "last_page", which stores the number of the last available page. In other words, we don't want the user to be able to press the "Next" button if there is no next page!

4) Turn “Perpetual” on.

Click in the box located next to the word “Perpetual”. You should see an “X” appear in the box. If a response is Perpetual, it is always active (provided the conditional in the “Active if TRUE” box holds true). Even if the user is engaging in some interaction on one of the “pages” of our course, or if a movie or sound is playing, we want the user to be able to interrupt whatever is going on and switch pages.

At this point, the “Pushbutton Options” window should now look like this:



5) Move the “Next” pushbutton to the immediate right of the words “Page 1 of 6” located at the bottom of the “Presentation Window”.

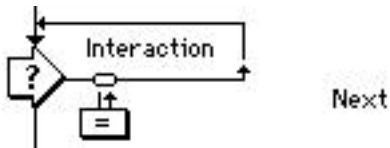
Place the mouse pointer in the middle of the “Next” button, and drag the button just to the right of the words “Page 1 of 6” at the bottom of the “Presentation Window”. This demonstrates the ease with which Authorware lets you position objects.

Now, the screen should look something like this:



6) Click on the “OK” button in the “Pushbutton Options” window

You will be returned back to the “Design” window. Note that the interaction you’ve created now looks a bit different than before:



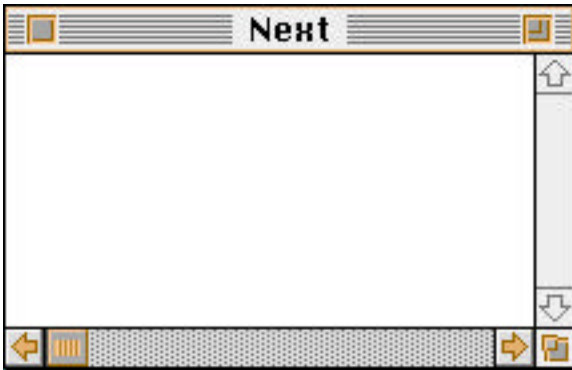
The arrow that leaves the calculation icon now points back to the Interaction Icon’s flowline. This happened as a result of switching from “Try Again” to “Continue”.

2.4 Editing a Calculation Icon

Now, we will edit the contents of the Calculation Icon attached to the “Next” pushbutton response, so that it performs the functions we want it to. Calculation Icons can contain statements that resemble traditional programming statements. Functions may be called within them, and calculations performed.

1) Open (double-click on) the Calculation Icon attached to the “Next” response

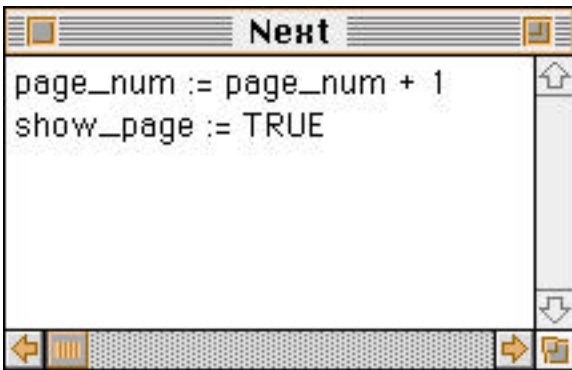
You will see a window like that shown below:



2) Enter the text below in the Calculation Icon window:

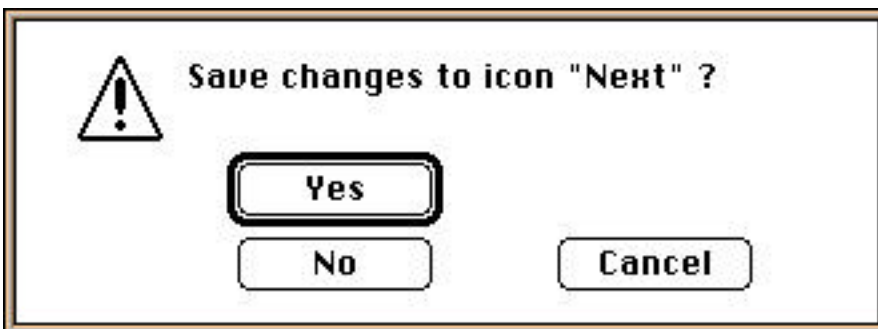
```
page_num := page_num + 1  
show_page := TRUE
```

The first line causes the number of the current page to be incremented by one. The next line sets the Boolean variable “show_page” to TRUE. We’ll use this Boolean variable later to activate the code that will actually show the current page. When you’re finished, the Calculation icon window should look like this:



3) Close the “Next” window

(Click in the small square in the left hand side of the title bar at the top of the window). You will then see a window like that below:



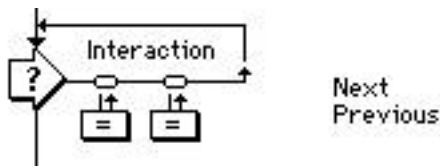
4) Click on the “Yes” button

Now, you are back at the Design Window, with no other windows open. You have finished creating the “Next” button, as well as defining its functionality.

2.5 Adding a “Previous” Pushbutton Response

At this point, you should know how to create a pushbutton response, position the button, change the program flow that takes place after the icon attached to the response icon is executed, and how to edit a calculation icon. You have been given step by step instructions needed to create a “Next” button. Now, you will create a “Previous” button, along with its associated functionality. However, the instructions given will be somewhat less detailed. If you become confused, reference the instructions given for creating the “Next” button.

1) Drag a Calculation Icon from the Authorware icon palette to the immediate right of the “Next” pushbutton response and title it “Previous”



Notice that the response type for “Previous” is pushbutton, and that execution returns to the interaction flowline after the calculation icon is executed. This is because when you add a new icon to an interaction flowline, Authorware, by default, sets certain response options to be the same as the last icon added to the flowline. (Response type and execution flow type are two of these options.)

2) Open the “Previous” pushbutton response icon. Enter the following in the “Active if TRUE” text box:

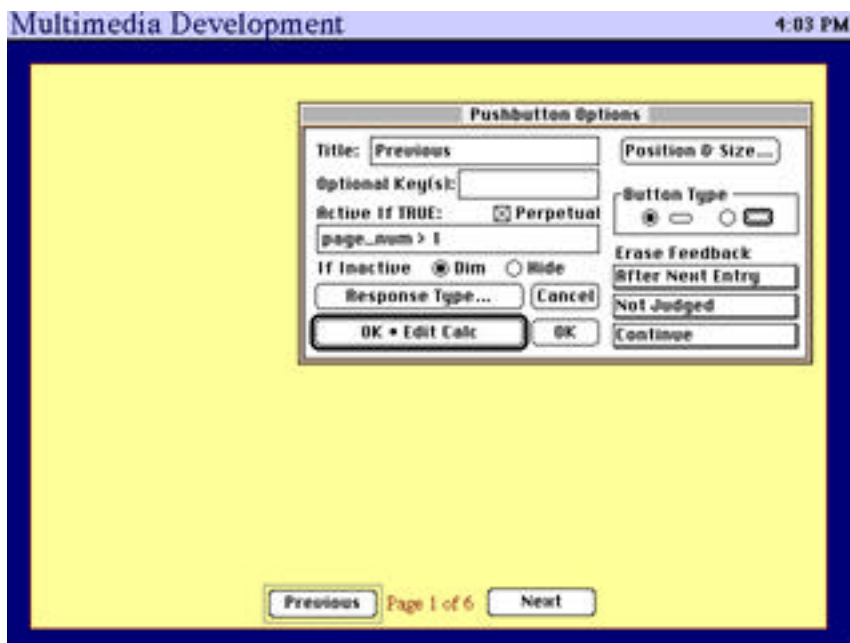
page_num > 1

We do not want the “Previous” button to be active if we are on the first page.

The other options are already set as we want them (“Perpetual” on, “Continue” execution flow), as these were the settings for the “Next” button (as previously explained).

3) Move the “Previous” pushbutton to the immediate left of the words “Page 1 of 6” located at the bottom of the Presentation Window.

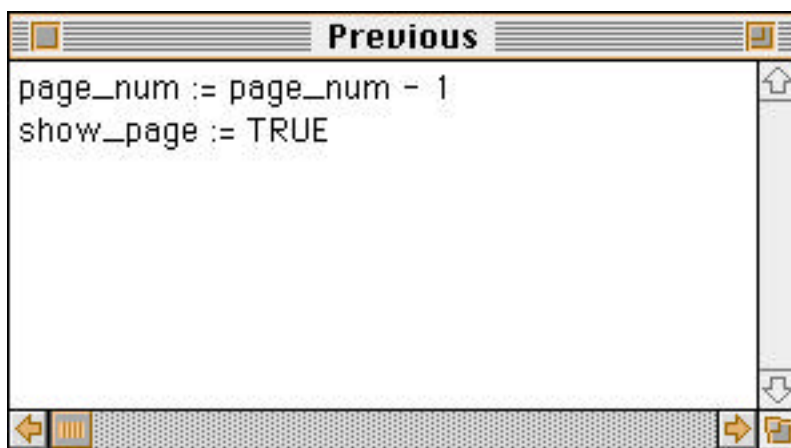
When you are finished, the Presentation Window will look similar to the one below:



4) Click the “OK” button in the “Pushbutton Options” window.

You will be returned to the “Design” window.

5) Open the Calculation Icon attached to the “Previous” pushbutton response icon, and edit its window so that it looks like that shown below:



The code here is similar to that for the “Next” button, except that here we decrement the page number.

6) Close the “Previous” Calculation Icon window (click on the “Yes” button when asked if you want to save changes)

You should be back in the Design Window. You’ve finished adding the Previous button, as well as its associated functionality.

2.5 Testing Your Program

Now, give your program a test run. Select “Run” from the “Try It” menu (or press “Command R”). You will see the Presentation Window. Try clicking on the “Next” and “Previous” buttons. The page number changes, but nothing else happens. That’s because we haven’t created any pages yet! **Do NOT double click anywhere within the Presentation Window. (Double-clicking will put you into “authoring mode”, which allows you to make changes to the graphics, text, etc. you see on the screen. We are not covering this feature of Authorware in this tutorial.) IF you do accidentally double click, press Command Z (for Undo) and close any window that is opened.**

In the next section, we’ll actually create the pages that make up our course. When you are finished testing, press “Command J” to return to the Design Window.


3.0 Showing the Current Page

Now, we’ll actually create the code that causes the current page to be shown.

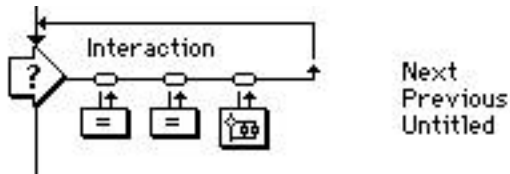
3.1 Adding the “show_page” Conditional Response

When either the “Next” or “Previous” button is clicked, the very last action taken is setting the Boolean variable “show_page” to TRUE. It was mentioned earlier that this would “trigger” some code that would show the appropriate page. Through the use of a Conditional Response, we can cause code to be executed based on the value of a Boolean variable or expression, rather than by a direct user action.

1) Drag a “Map” icon from the Authorware icon palette to the immediate right of the “Previous” pushbutton response icon.

A Map Icon looks like this . (The function of a “Map” icon will be explained later.) Notice that yet again, we have a pushbutton response, with “Continue” execution flow. However, we will want to change the response type, as well as set other options. Also, don’t bother to title this icon as yet. (You’ll see why soon.)

Your interaction now should look like this:



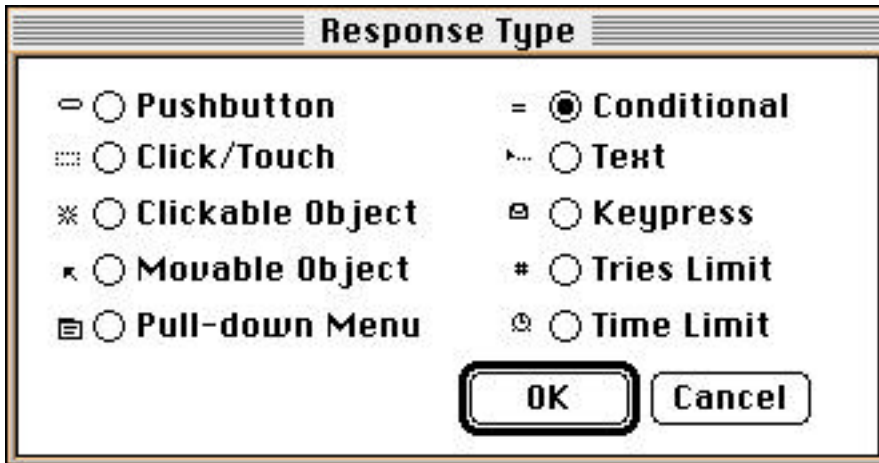
2) Open (double click) on the new “Untitled” pushbutton response

Your screen will look something like this:



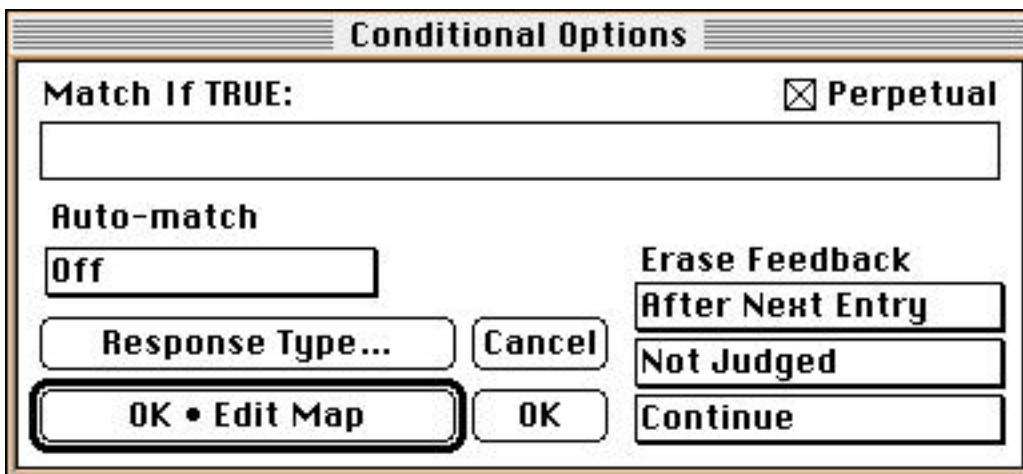
3) In the “Pushbutton Options” window, click the “Response Type” button.

You will see a window titled “Response Type” appear. Select “Conditional”. When you have done this, the window should look like this:



4) Click on the “OK” button.

Where the “Pushbutton Options” window was, you will now see a window like that shown below:



We have changed the type of the response from “Pushbutton” to “Conditional”. Thus, the set of options we have to choose from is somewhat different.

5) In the box under the words “Match if TRUE”, enter the following:

show_page

We want this “Conditional” response to be activated (and thus the icon attached to it executed) if the Boolean variable “show_page” has value TRUE. (This is why the last line of the “Next” and “Previous” code sets this variable to TRUE.)

6) Turn “Perpetual” off

There is no good reason to make this response “Perpetual”, and the polling required to constantly check for matching of “Perpetual” responses can slow things down quite a bit. Thus, it’s a good idea to only make responses “Perpetual” when necessary.

7) Change the selected menu item in the pull down menu under the title “Auto-match” from “Off” to “When True”.

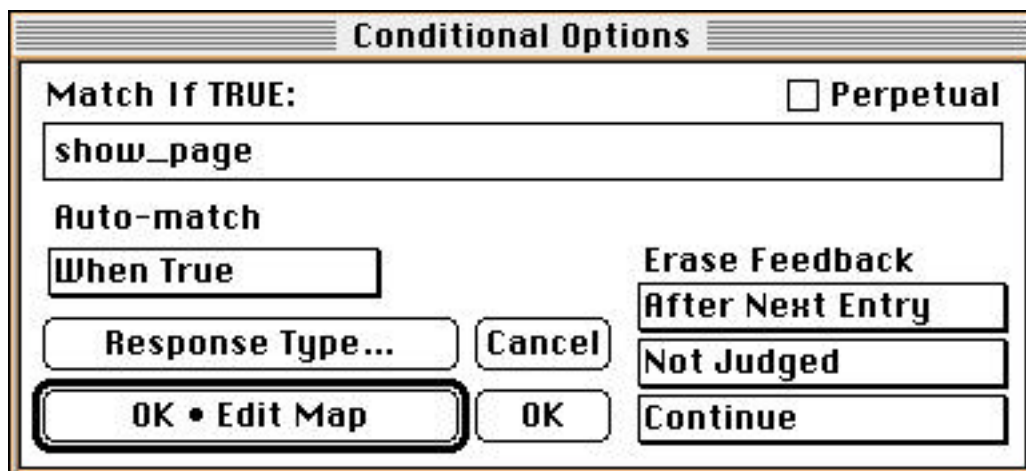
Normally, if a conditional response is located on an interaction flowline, it can only be matched if the mouse is moved, the mouse button is pressed, or text is entered, and at some point, program flow moves along the interaction flowline and happens to “hit” the conditional response icon. This is because if the value of the Boolean expression or the conditional is true, we don’t necessarily want to repeatedly match it. (This is what happens if “Auto-match” is off.)

However, we would like the first page to automatically be displayed when we first run the program. In the “Initialize variables” calculation icon, “show_page” has been initialized to TRUE, and page_num has been initialized to TRUE. (You can check this out for yourself by opening that icon, if you like.) But, we don’t want the user to have to press the “Next” or “Previous” button to show the first page.

Thus we turn the “Auto-match” option to “When True”, so that if the value of “show_page” becomes true, the interaction flowline is traversed (just as if the mouse button was pressed or the mouse is moved). In this way, the first time the interaction is encountered, “show_page” is true, the interaction flowline is traversed, the “Conditional Response” icon is matched, and the attached code executed.

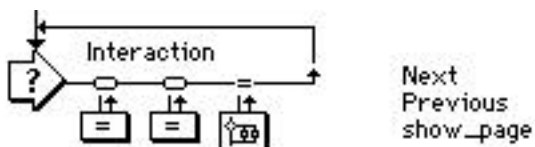
(Note that if we didn’t want the first page to be automatically shown, it would have been sufficient to have Auto-match off, since if the “Next” or “Previous” button is pressed, flow returns to the flowline after the icons attached to these responses are executed, and then the conditional would be matched, since “show_page” is true.)

When you are finished making all these changes, the “Conditional Options” window should look like this:



8) Click on the “OK” button in the “Conditional Options” window

You will be returned to the “Design” window. Your interaction should now look like that shown below:

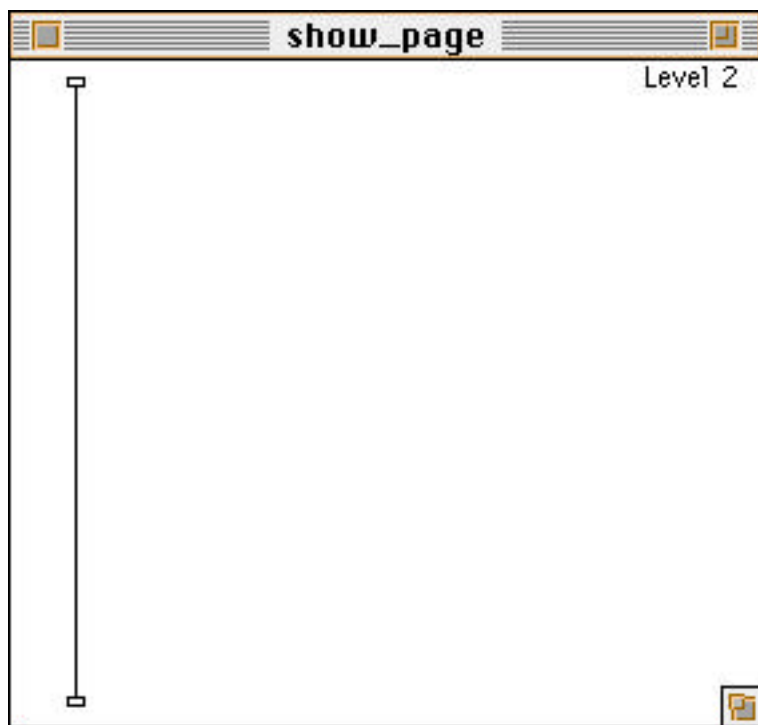


Note that everything looks the same, except that the “Conditional Response” icon now has a name! This is because the name of a “Conditional Response” icon is always the condition that is matched.

You’ve finished creating the “show_page” Conditional Response and setting the appropriate options for it. Next, you’ll have to specify the code that is executed when this response is “matched”.

3.2 The Map Icon

1) Open (double-click on) the “Map” icon attached to the “Conditional Response” icon. You should see a window open that looks like that shown below:



The “Map” icon is very important. A “Map” icon contains its own flowline. When a “Map” icon is encountered during execution, the icons on the “Map” icon’s flowline are executed in exactly the same manner as the icons on the “main” Authorware flowline. Anything that can be placed on the “main” flowline can be placed on a “Map” icon’s flowline. This includes other “Map” icons, which means that Authorware provides the

capacity for unlimited nesting. Notice that, in the upper right corner of this level are the words “Level 2”. This means that this “Map” icon is located at the second level of nesting (i.e., there is one level above it).

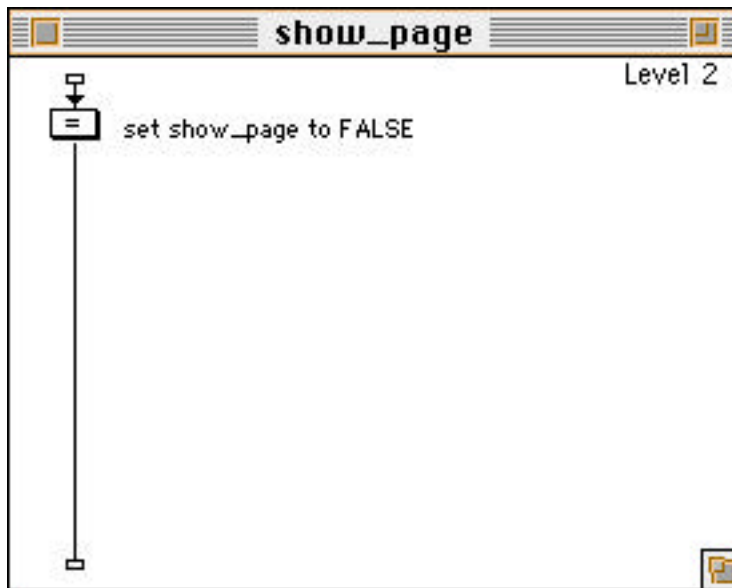
3.3 Resetting the “show_page” Variable

After the code contained in the “Map” icon is executed, we don't want it to be executed again unless the “Next” or “Previous” buttons are pressed. However, this is exactly what will happen with our program as it stands. After the code in the “Map” icon executes, execution will return to the Interaction Icon. But, “show_page” is still TRUE. And, Auto-match is on, so execution will immediately proceed back along the interaction flowline, the “show_page” conditional will be matched again, and so on. Thus, we have just created an endless loop.

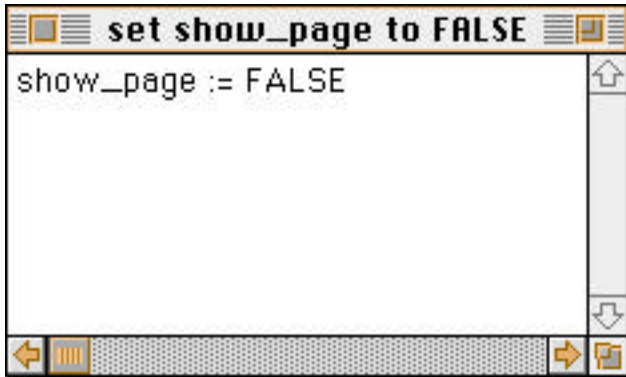
We will prevent this from happening by setting the “show_page” variable to FALSE as soon as the “show_page” conditional is matched.

(1) Drag a “Calculation” icon from the Authorware icon palette to the “Map” icon’s window (entitled “show_page”) and title it “set show_page to FALSE”

The “show_page” window should now look like this:



(2) Open the window for this Calculation Icon, and edit it to look like the one shown below:




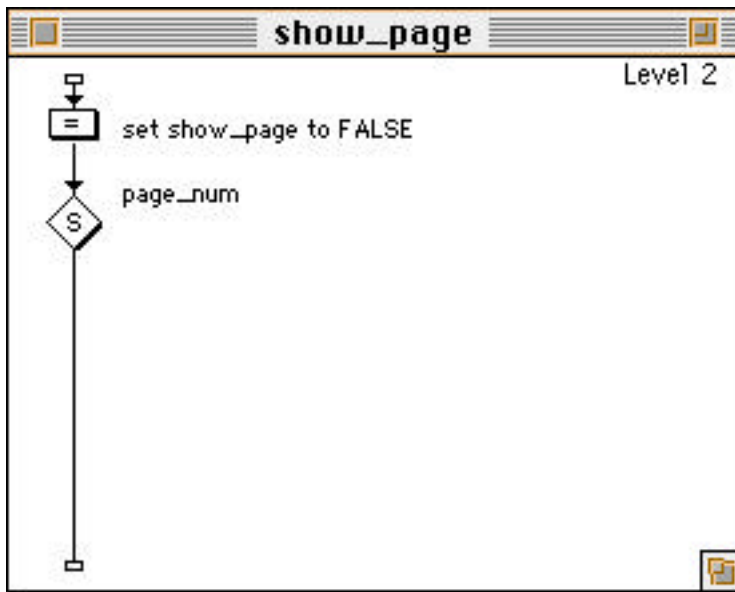
(3) Close the Calculation Icon window (click on the “Yes” button when asked if you want to save changes).

3.4 The Calculated Path Decision Icon

We need to be able to show the current page (whose number is determined by the variable “page_num”). Authorware provides the capacity for program branching through “Decision” icons. The “Calculated Path Decision” icon is a particular type of Decision Icon, which allows program branching to be based on the value of a numeric variable (somewhat like a “case” statement in Pascal, or a “switch” statement in C). We’ll see how to use this to display the current page of our course.

(1) Drag a “Decision” icon from the Authorware icon palette to the “show_page” Map Icon’s flowline directly below the Calculation icon, and title it “page_num”

A Decision Icon looks like this . The “show_page” window should now look like that shown below:



(2) Open the Decision Icon

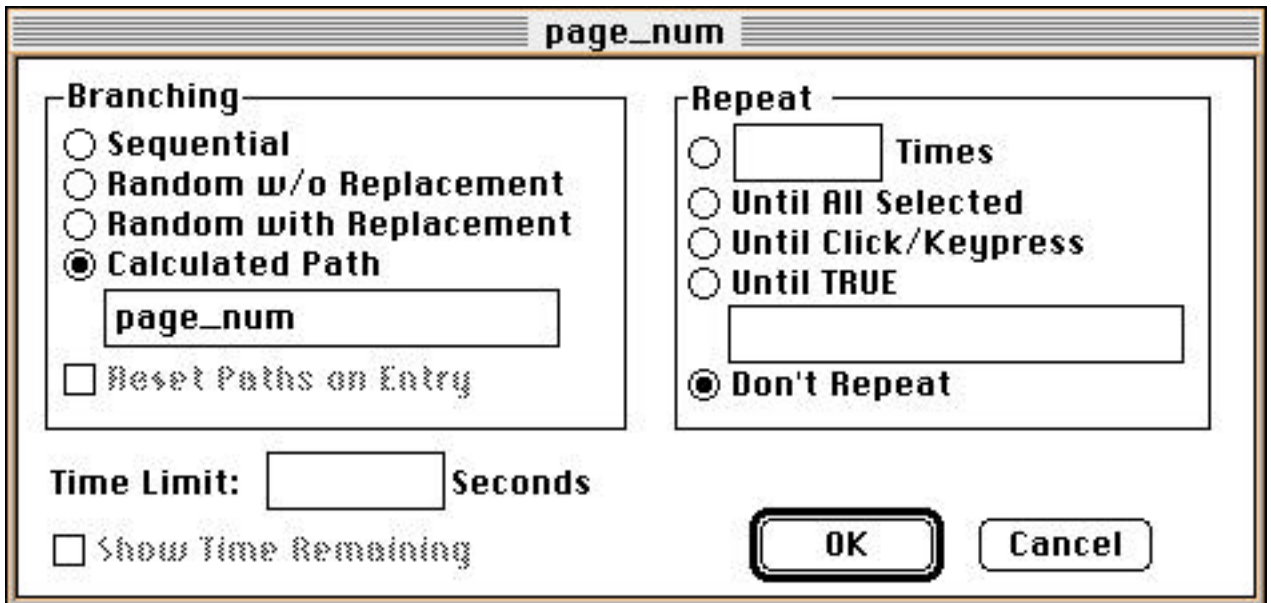
You will see the window shown below:

(3) Change the type of branching to “Calculated Path”

(4) In the text box below the words “Calculated Path”, enter the following:

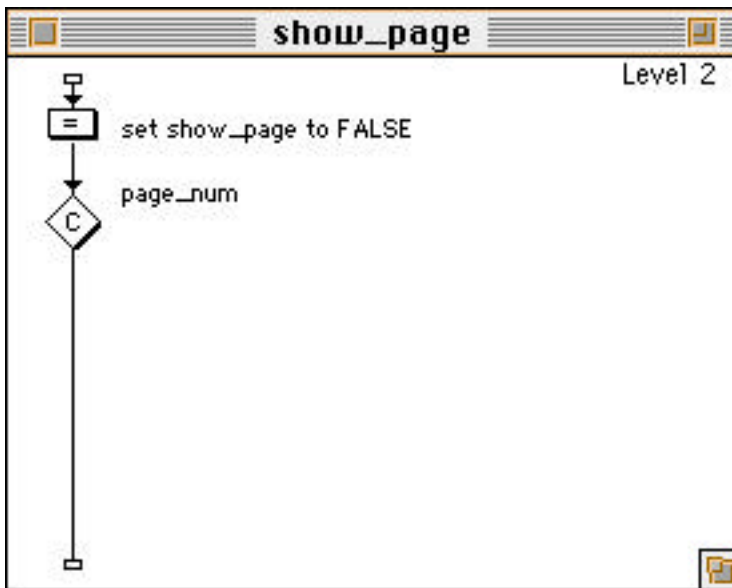
page_num

The window should now look like this:



(5) Click the “OK” button

The Decision Icon window will close, and you will see again the “show_page” flowline. It should look like that shown below:



Notice that the letter shown in the Decision Icon has changed from a “S” to a “C”. This has happened because you’ve changed the type of the Decision Icon from “Sequential” to “Calculated Path”.

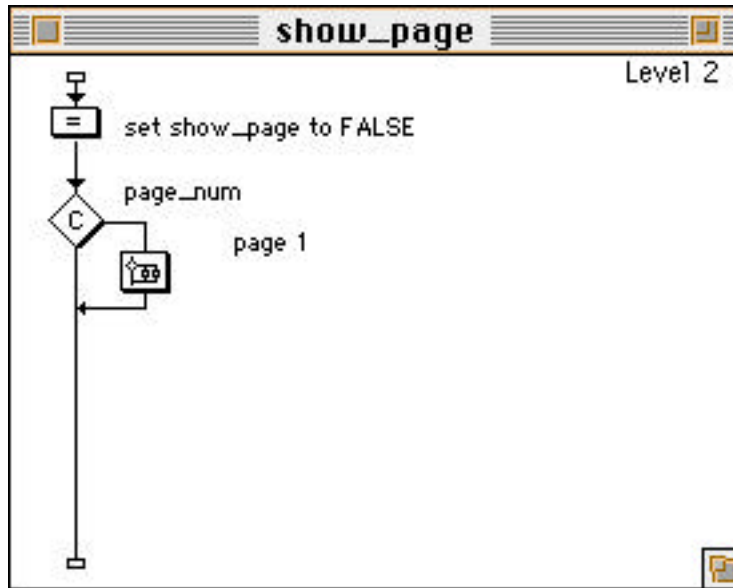
What we’ve done is created an icon that will cause branching to “attached” icons based on the variable “page_num” (and thus will cause the appropriate page to be

displayed). Next, we'll create the "branches" (one per page) which this Calculated Path Decision Icon will choose among.

3.5 Attaching Icons to a Decision Icon

(1) Drag a Map Icon from the icon palette to the immediate right of the Decision Icon. Title this Map Icon "page 1".

The "show_page" flowline should now look like this:



Note that, like an Interaction Icon, a Decision Icon has its own flowline.

(2) Open the "page 1" Map Icon.

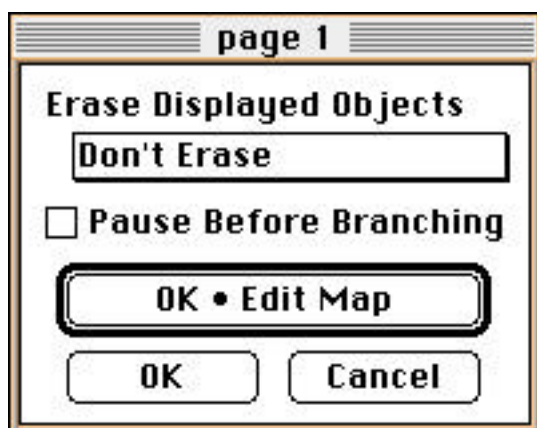
You will see a window that looks like this:



(3) Change the currently selected option in the pull-down menu beneath the title “Erase Displayed Objects” from “Before Next Selection” to “Don’t Erase”

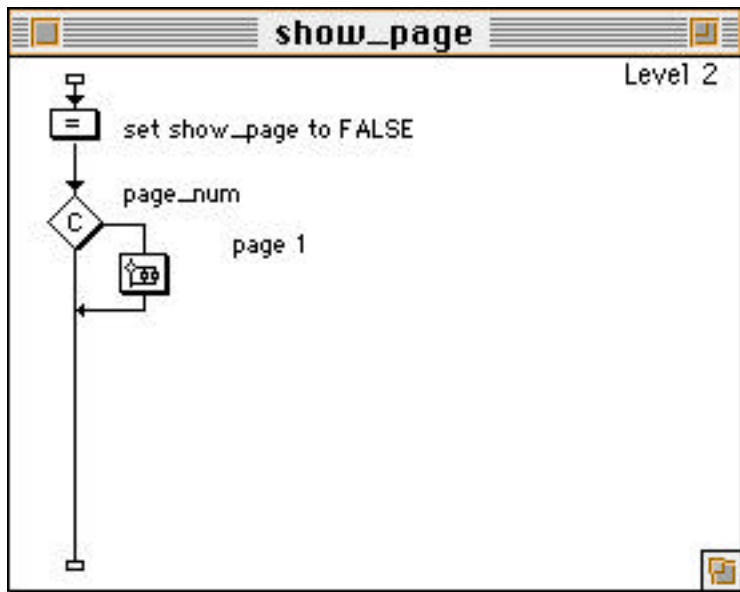
Authorware provides the ability to automatically erase anything displayed in an icon attached to a Decision Icon, either immediately upon leaving that icon (“Before Next Selection”) or upon exiting the branching structure (“Upon Exit”). However, we don’t want to do either of these, so we select “Don’t Erase” to prevent anything we display from being erased as soon as it’s shown! (In case you are wondering how, when a new page of our course is displayed, the last displayed page is erased, open the “show_page” conditional response on the interaction flowline. You then will see that under the title “Erase Feedback”, the currently selected option is “After Next Entry”. This means that the contents of the Map Icon attached to the “show_page” conditional response will be erased the next time a response is matched. Thus, when the Next or Previous buttons are pressed - and thus the appropriate response matched - the contents of the “show_page” Map Icon are automatically erased.)

Now, the “page 1” options window should look like this:



(4) Click “OK”

You should be returned to the “show_page” Map Icon window.



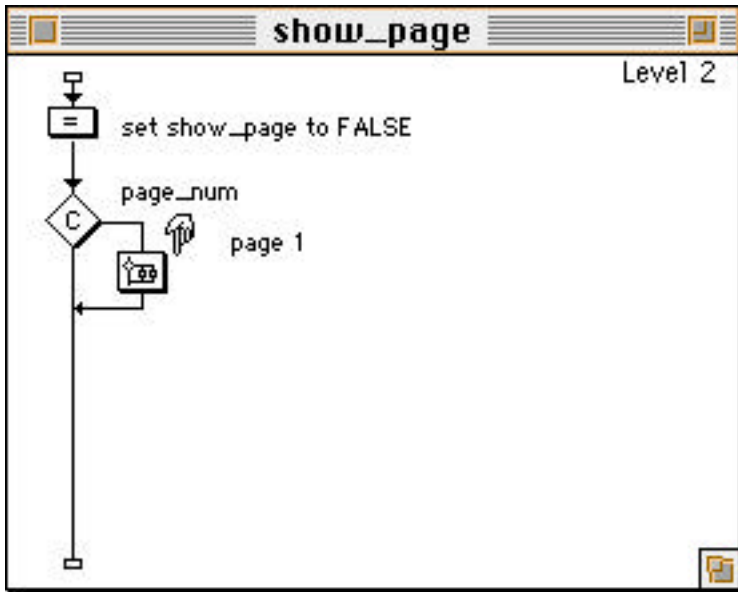
3.6 Adding More Pages

Now that you've attached a Map Icon to the Calculated Path Decision Icon's flowline for the first page of your course, you must add Map Icons for the rest of the pages (2 through 6) as well.

To save time, we'll simply copy the "page 1" Map Icon, and then attach multiple copies of it (by pasting them) to the "page_num" Calculated Path Decision Icon.

- (1) Select the "page 1" Map Icon (single click on it, so it is highlighted)**
- (2) Press "Command C" (this copies the icon to the Macintosh Clipboard)**
- (3) Click (once) to the immediate right of the "page 1" Map Icon**

The "show_page" Map Icon's window should now look like this:

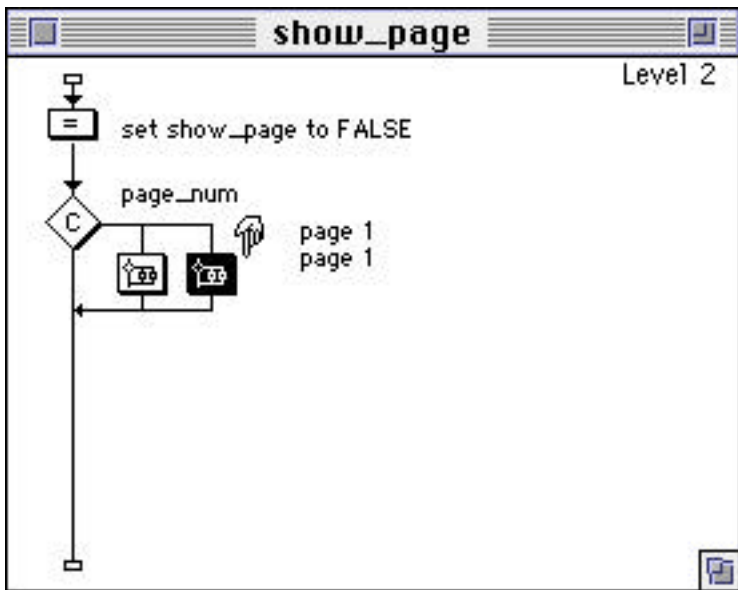


Notice the small picture of a hand to the right of the “page 1” Map Icon. This is called the “paste hand”. It points to the position where the contents of the Clipboard will be pasted, if that action is performed. (To move the paste hand elsewhere, you just have to click where you want it.)

(4) Press “Command V”

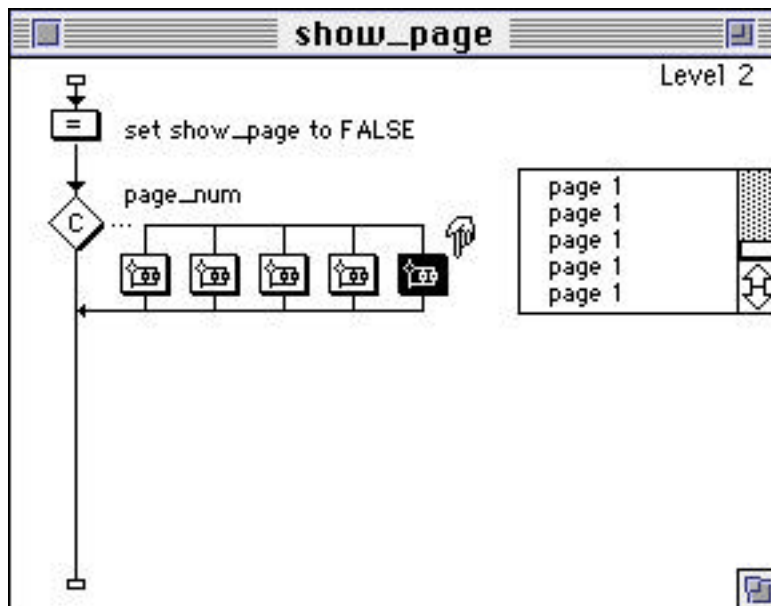
This “pastes” a copy of the “page 1” Map Icon right beside the original one.

The “show_page” window should now look like this:



(5) Press “Command V” 4 more times.

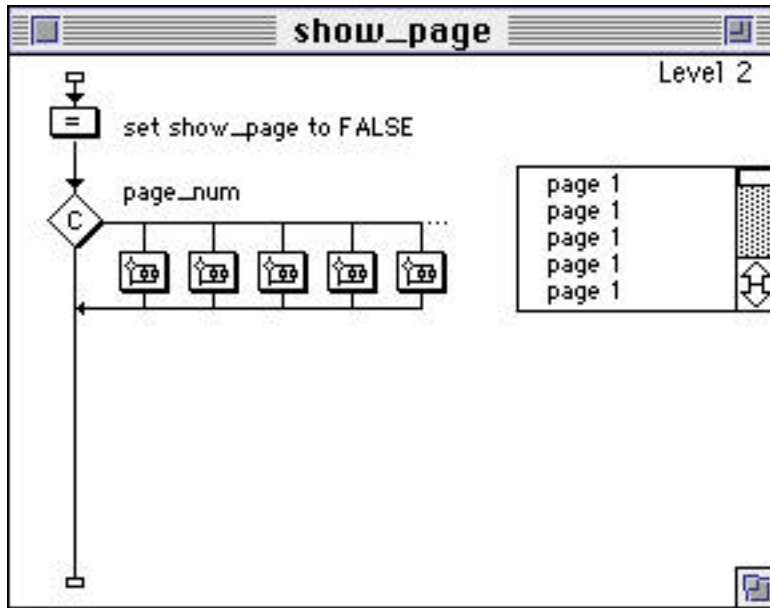
The “show_page” window now looks like this:



You’ve added 4 more copies of the “page 1” Map icon. Note that the list of icon names to the right of the Calculation Icon’s flowline has become a scrolling list. This happens because there is only really space for five icon names beside the Calculated Path Decision Icon’s flowline, so the list was made scrolling so that more icons could be added. Note, also, the ellipses (“...”) beside the Calculated Path Decision Icon. These indicate that there are more attached icons to the left. The five icons whose names are shown in the scrolling list window are the ones that are shown at any given time.

(6) Scroll to the top of the icon name list.

To scroll to the top of the list, click and hold the up arrow in the bottom left corner of the scrolling list window. (You could also drag the scroll bar, the little rectangle on the side of the scrolling window, to the top of the window.) When you’ve scrolled to the top of the list, the scrolling list window should look like that shown below:



Note that the ellipses (“...”) are now to the right of the Calculated Path Decision Icon’s flowline. This means that there are more attached icons to the right.

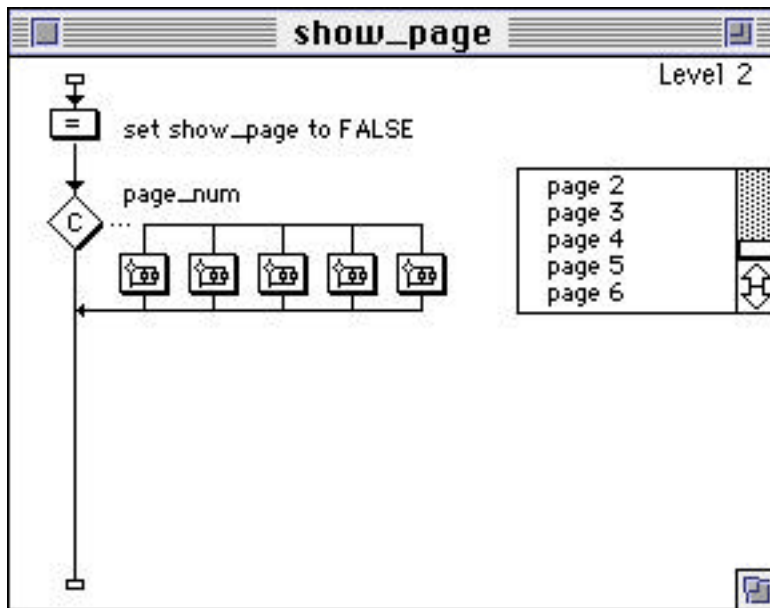
(7) Change the name of the second icon attached to the Calculation Icon from “page 1” to “page 2”

Click on the second name shown in the scrolling list. Note that both the icon and title highlight. Type *page 2*.

(8) Similarly, change the names of the rest of the icons to page 3, page 4, page 5, and page 6.

Note: To proceed to the next icon title in the list, you can just press “Enter” (after changing the current icon’s name).

When you’re done, the Calculated Path Decision Icon’s scrolling list of icon titles should look like that shown below:



Next, we'll add the flowline icons that will cause the first page of our course to be displayed. (We've only added empty Map icons for each page - we need to actually display something inside each of them.)

4.0 Obtaining Content From Libraries

Authorware allows you to store some types of icons in "libraries". Libraries are often used to store graphics and audio icons. When you want to use an icon from the library in your program, you "drag" the icon from the library window to your flowline. A "link" is created on the flowline to the original icon in the library. You can make multiple links to the same icon in a library. There are some important advantages to storing icons in libraries:

- If you need to modify a sound or graphic, you don't have to go hunting through your code to find it.
- If you use the same graphic or sound in multiple places in your program, you can just place it in the library and create multiple links. Then, if you need to modify the graphic or sound, you just need to modify the icon in the library. If, instead, you had not used a library, but just placed a new copy of the icon everywhere you wanted to use it, you would have to hunt through your code and modify each and every copy of the icon.
- A link to an icon is only about 4K, much less, in most cases, than the original icon. Thus, if you are going to use a graphic or sound more than once, placing the icon in a library and making links to it, rather than making new copies of the icon, can greatly reduce the size of your program.

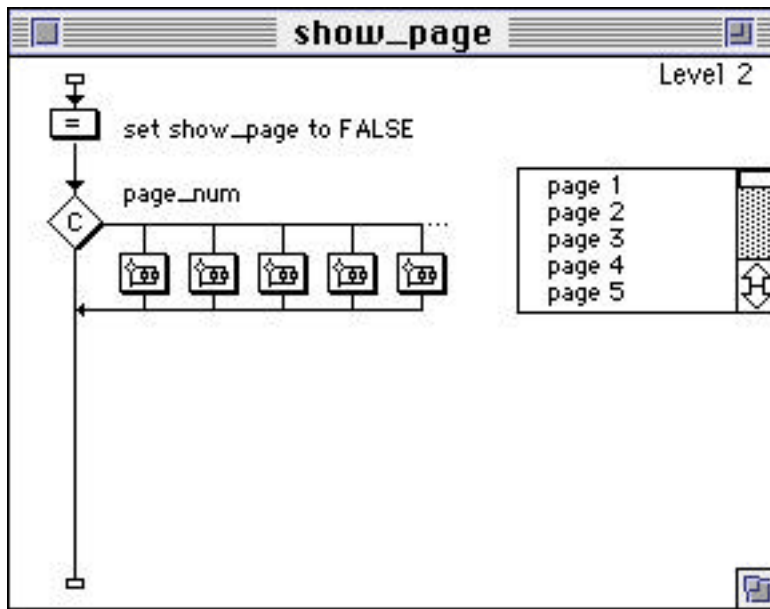
The graphics, sounds, and text for the course you are assembling are all stored in a library called "multdev.lib". These items were prepared in advance and put into the library for you. Of course, Authorware provides tools for editing text, drawing graphics, creating

animations, and (on the Macintosh) recording sound, as well as letting you import graphics, sound, and movies prepared in other packages. However, the true power of Authorware lies in its ability to let the author create fairly complex interactions and integrate different media elements, not in creating the actual media elements themselves. Therefore, it was decided to free you from having to create or import the media elements, and instead just demonstrate how they can be brought together to create an interactive multimedia course.

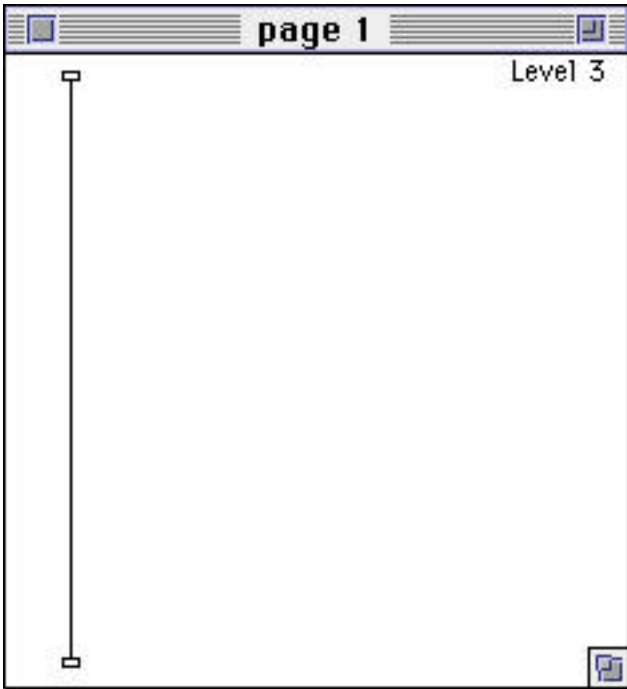
4.1 Assembling the First Page

Now, we'll obtain the actual content of the first page of our course from a library.

(1) Scroll to the top of the scrolling icon title list.

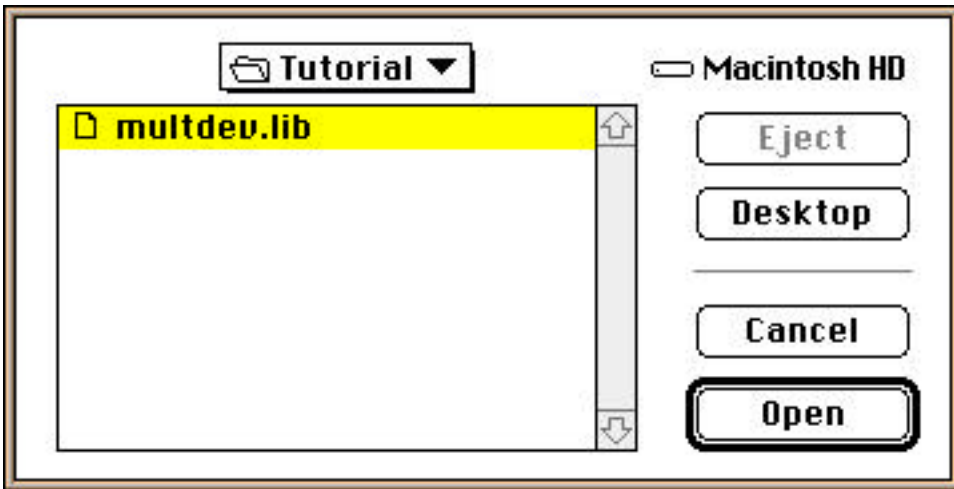


(2) Open the "page 1" Map window



(3) Choose “Open Library...” from the “Libraries” menu

You will see a window similar to the one shown below:



(4) Select “multdev.lib” and click on the “Open” button

(“multdev.lib” should already be selected, since it is the only library in your Tutorial folder.)

You will see the following window:




This is a library window. The window for a library consists of a scrolling list of all the icons contained within the library.

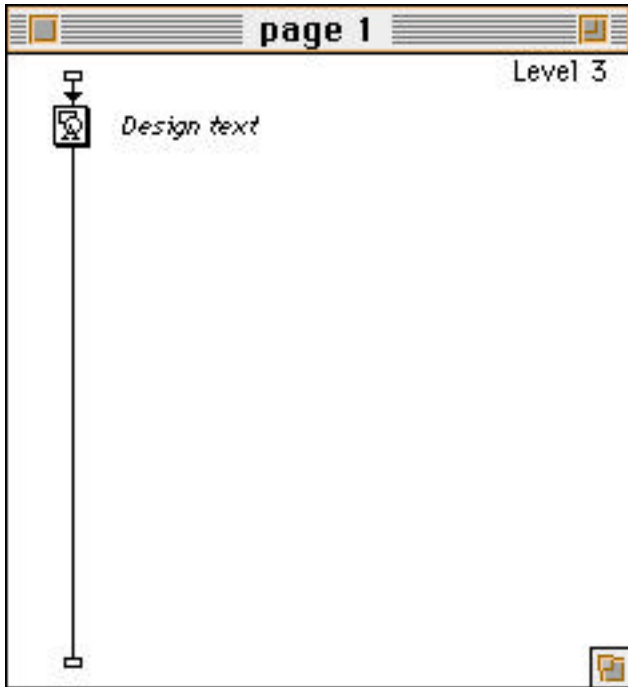
If you ever accidentally close the library window, or just want to bring it to the front, do the following:

1. Select the "Libraries" option from the "Libraries" menu.
2. Keep the mouse button held down. To the right of the highlighted "Libraries" option you should see a menu of libraries for this application - in this case, there is just one library, so the menu has just one item - the "multdev.lib" library.
3. Select the "multdev.lib" library by moving the mouse to the right (while keeping the button held down), and letting go of the mouse button when the library is selected. Now, the library window should open.

(5) Drag the Display Icon titled "Design text" from the "multdev.lib" library window to the "page 1" Map Icon's flowline.

(Reminder - A Display Icon looks like this: . A Display Icon is used to display graphics and/or text on the screen.)

The "page 1" Map window should now look like this:




Note that the icon title for the Display Icon on the “page 1” flowline is in italics. This signifies that the Display Icon is really just a “link” to the original (in the “multdev.lib” library).

(6) To see the results of what you’ve done, run the program.

Select “Run” from the “Try It” menu, or press Command R.

You should see a screen similar to that shown below:



Design

- As with any software project the design phase should be completed before production begins.
- The objectives of the multimedia software must be identified first. For example, if the software is intended to instruct, then the material that should be learned as a result of using the software should be identified.
- Then, design of the project may commence. Often, the design is split into high and low level design, just as a usual software project.
- However, the design of a multimedia project involves other tasks, which should be noted:
 - (1) If video is to be incorporated, a video script must be produced (much like the script for a motion picture or TV show).
 - (2) If narrative audio is to be included, an audio script should be produced. This script will be used when the audio is to be recorded.
 - (3) Graphics to be used must be identified, and often sketched out.

Previous Page 1 of 6 Next

You've created the first page of your course! Now, all that remains is to obtain the content of the rest of the pages of your course from the "multdev.lib" library.

4.2 Adding Content to the Rest of the Pages

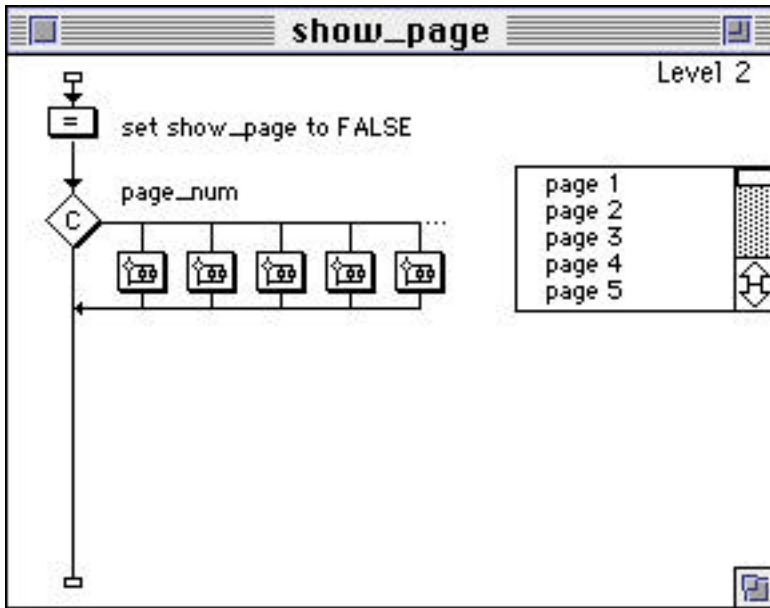
By now, you should be comfortable with opening and closing Map (and other) icon windows, and dragging items from libraries to various flowlines. Therefore, instead of giving you detailed instructions on how to assemble the rest of the pages, you're simply shown what each Map icon's window should look like when you're finished. (All icons shown in the Map Icon windows are to be dragged from the "multdev.lib" library. If you accidentally close the library window, reference the instructions given for reopening it earlier in this document.)

(1) Return to the Design Window

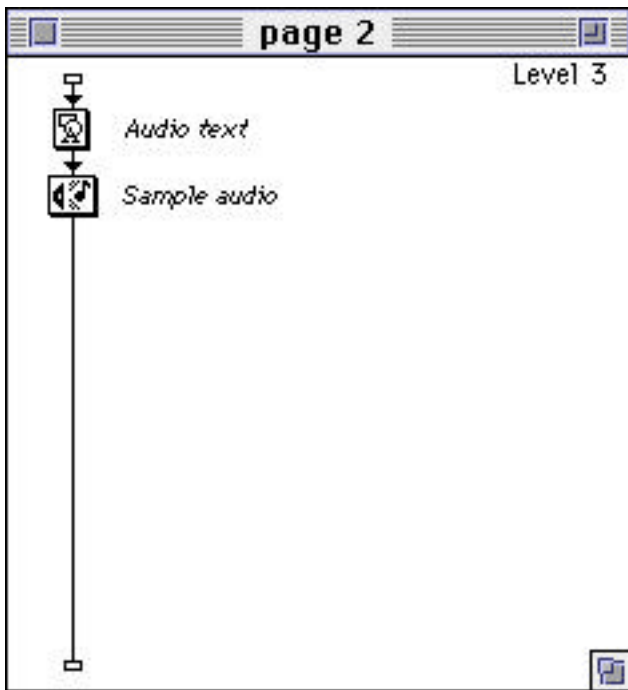
(Press Command J)

(2) Close the "page 1" window

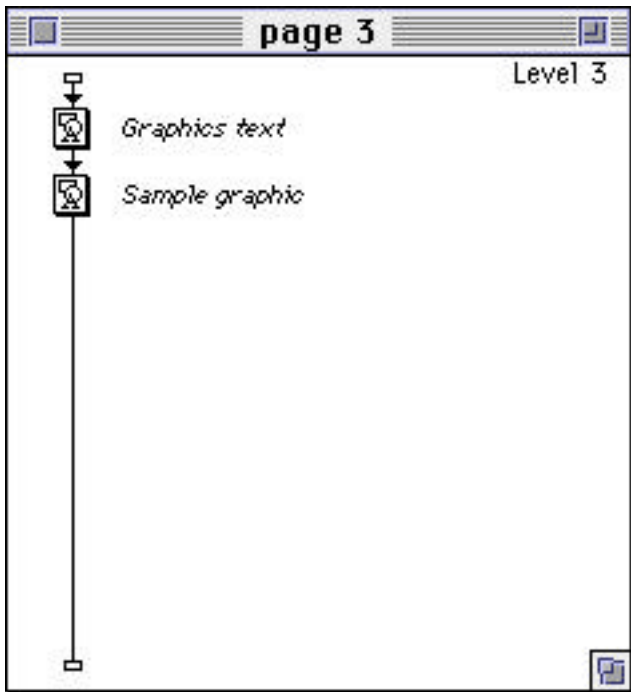
The "active" window should now be the "show_page" Map Icon window.



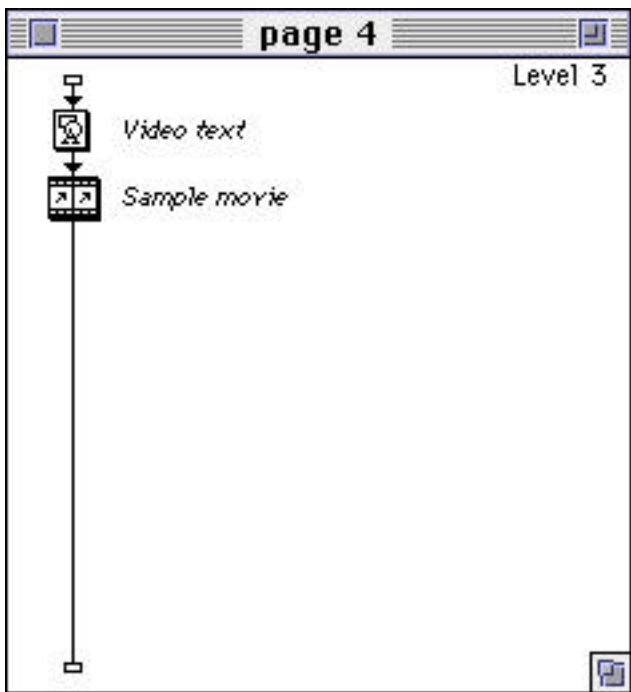
(3) Edit each of the “page 2” to “page 5” Map icons so that their windows look like those shown below:




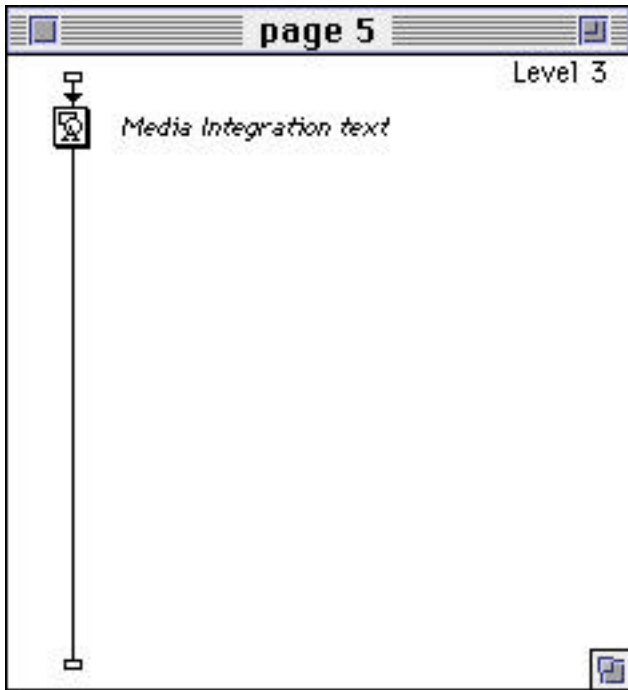
(The icon that looks like this  is a Sound Icon. It is used to play audio.)

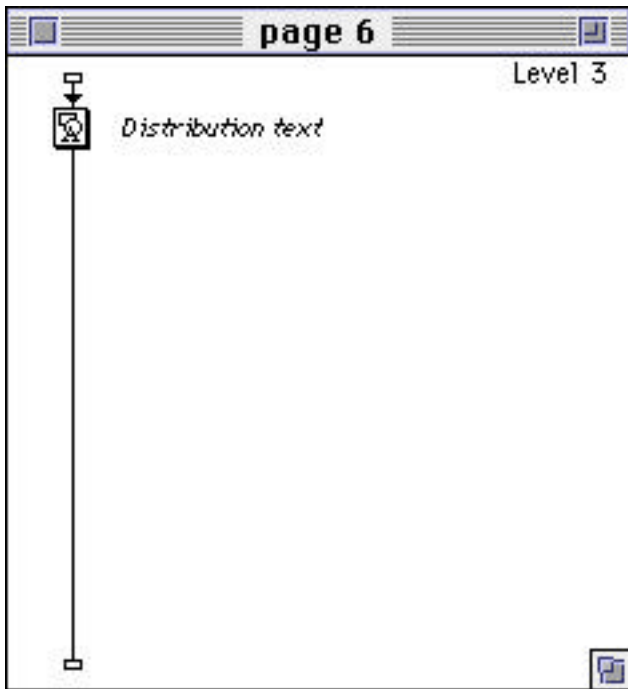


(Note that here we see the use of Display Icons to display both graphics and text).



The icon that looks like this  is a movie icon. It can be used to display movies on the screen (QuickTime movies, PICS animations, Director movies, etc.).





You've finished creating all the pages! Now, check to make sure your program works correctly.

(4) Run your program (select "Run" from the "Try It" menu or press Command R)

(5) Flip through the pages of your course by pressing the "Next" and "Previous" buttons

Remember, DON'T DOUBLE CLICK - reference the instructions given earlier if you accidentally do. Test out your program. Take a look at each of the screens you've created. Notice how the "Next" and "Previous" buttons become active and inactive, depending on the page you're on. The pages should look like those shown below:

page 1

Design

- Before any production begins, as with any software project, the design phase should be completed.
- The objectives of the multimedia title must be identified first. For example, if the software is intended to instruct, then the material that should be learned as a result of using the software should be identified.
- Then, design of the project may commence. Often, the design is split into high and low level design, just as a usual software project.
- However, the design of a multimedia project involves other tasks, which should be noted:
 - (1) If video is to be incorporated, a video script must be produced (much like the script for a motion picture or TV show).
 - (2) If narrative audio is to be included, an audio script should be produced. This script will be used when the audio is to be recorded.
 - (3) Graphics to be used must be identified, and often sketched out.

[Previous](#) Page 1 of 6 [Next](#)

page 2**Audio****Spoken/narrative, requires:**

- 1) Script, script writers (must be familiar with content)
- 2) Recording and digitizing hardware/software
- 3) Professional voice (much like a radio announcer)

Music/sound effects, requires:

- 1) Source, can be either
 - a) Stock music CD's
 - b) Professional musicians
- 2) Recording and digitizing hardware/software

[Previous](#) Page 2 of 6 [Next](#)

page 3



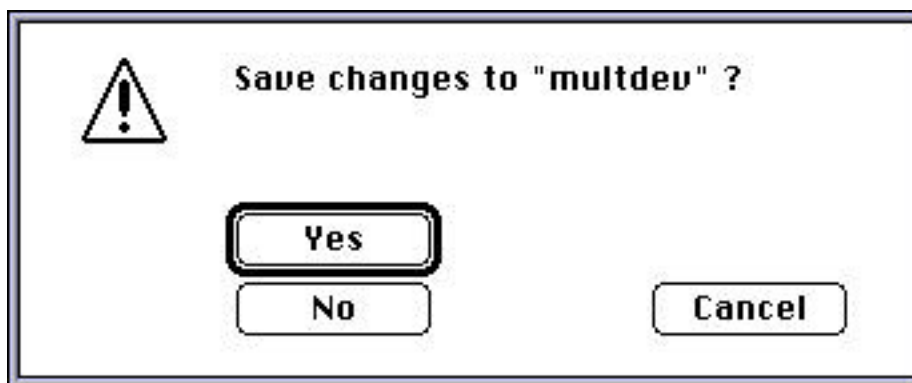
When you're done, press Command J to return to the Design Window.

5.0 Quitting Authorware

Now that you've finished this tutorial, you may exit Authorware.

(1) Select "Quit" from the File menu.

If you haven't saved your file recently, you may see a message like that shown below:



If you've made changes to your program since your last save and you try to exit, Authorware asks you if you want to save your file, so you don't lose those changes.

(If you haven't made any changes since your last save, Authorware will just quit.)

(2) (If you do see this message box) click “Yes”.

Authorware saves your program and exits.

6.0 Conclusion/Notes

Congratulations - you've created your first Authorware program!

Hopefully, you've gained some appreciation for the ease with which Authorware allow you to create user interactions and integrate media elements. In addition, you've seen an example of an “icon-based” multimedia authoring language - Authorware is one of several such languages. However, there are many other kinds of authoring languages - Director is based on a “score” and a “stage” with “actors”. Some are “card” based, like HyperCard and SuperCard.

Though we didn't see this feature of Authorware in this tutorial, you can also create standalone programs with Authorware, that do not require Authorware to be running to execute. Thus, finished applications may be distributed to end users without requiring them to have Authorware.

Appendix B - Unit materials

Exercises:

1) Run the Authorware demo program.

Make sure to learn the material presented in it. Note the different media elements incorporated in the demo.

2) Complete the Authorware tutorial.

In the process of doing so, you will use most of the Authorware icons. Try to be somewhat familiar with the function of each of them, and pay special attention to how program execution flow happens in Authorware (this includes understanding the Authorware "flowline").

Issues:

Authorware is an multimedia authoring language which allows for rapid development of interactive multimedia software. It is icon-based; programming is done by placing icons onto a flowline in a particular order. Thus, it is much easier for "non-programmers" to produce multimedia software with Authorware than with traditional programming languages, like C. (However, certain functions that can be performed in traditional languages, like binary file access and complex calculations, cannot be performed in Authorware, and require the use of low-level functions written in other languages).

Objectives:

- 1) To see how an authoring language like Authorware allows fairly rapid development of interactive multimedia software (in contrast to "traditional" programming languages like C)**
- 2) To be familiar with the "icon-based" programming paradigm employed in Authorware**
- 3) To learn the functions of most of the icons in Authorware (You will also learn the functions of the icons in the demo)**
- 4) To understand the "flowline" approach to program execution flow in Authorware (as opposed, to example, the "score" approach in Director)**
- 5) To be familiar with the two "modes" provided in Authorware (Presentation and Design)**
- 6) To be understand how Authorware provides the ability to create user interactions**

Study Questions:

- 1) Explain the function of each of the following Authorware icons:**
 - Display Icon
 - Erase Icon
 - Calculation Icon
 - Interaction Icon
 - Group Icon
 - Decision Icon

- Map Icon
- Sound Icon
- Movie Icon

2) How does Authorware provide code modularity? (Hint: what icon is involved?)

3) Compare/contrast Authorware to "traditional" programming languages (consider how program flow occurs, icons versus program language statements, code modularity, etc.) What are the advantages/disadvantages of Authorware?

4) Explain how a Calculated Path Icon controls program branching

5) Explain how a Conditional Response can affect program execution (when it is present on an Interaction icon's flowline)

Appendix C - Quiz Questions

Set 1:

1. Give the number of the answer for each of the following three parts:
 1. Which of the following icons allows text to be shown on the screen?
 1. Text icon
 2. Type icon
 3. Display icon
 4. Word icon
 2. Which of the following icons is used to display a QuickTime movie?
 1. Display icon
 2. Movie icon
 3. Video icon
 4. QuickTime icon
 3. What is the name of the window in which Authorware program execution takes place?
 1. Presentation window
 2. Design window
 3. Display window
 4. Execution window
 5. Program window
2. What Authorware icon provides modularity and the capability for unlimited nesting? Explain how it accomplishes this.
3. Describe an Authorware program that causes one of three graphics (named graphic 1, graphic 2, graphic 3) to be displayed based on the value of a variable. Hint: Describe the icons involved and how they are "attached" to one another.

Set 2:

1. In Authorware, explain the effect of turning on "Automatch" for a conditional response. Discuss briefly the value of this feature.
2. Describe an Authorware program that causes the value of a variable X to be increased by one every time a pushbutton entitled "Add 1" is pressed. Hint: Describe all the icons involved and how they are attached to one another, and the contents of the icon which causes 1 to be added to X.

Set 3:

1. In Authorware, what is the effect of turning on the "Perpetual" option for a pushbutton response? Please explain briefly, and indicate what the value is of this feature.
2. Describe an Authorware program that causes the sound titled "Beep" to be played in the following two situations: when the pushbutton entitled "Play Beep" is pressed, and the first time the interaction to which this sound is attached is encountered. Discuss all the icons involved, and the contents of any Calculation Icons. Hint: A "conditional" response icon will be required. You will need a Boolean variable, therefore. Assume there is already one called play_beep that is initialized to false. Consider both the condition for the conditional icon, and if Automatch is turned on or off.
 - a) Text icon
 - b) Type icon
 - c) Display icon
 - d) Word icon

Appendix D - Quiz Answers

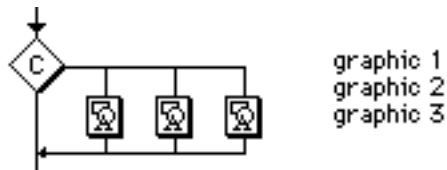
Set 1:

- 1.1 1. Display icon
- 1.2 2. Movie icon
- 1.3 1. Presentation Window

2. Authorware allows for modularity and nesting through the use of the Map icon. A Map icon provides its own flowline, which behaves in the same manner as the “main” (Level 1) Authorware flowline. Any icons may be included on the Map icon’s flowline, including other Map icons. Thus, Authorware provides for unlimited nesting in this manner. When a Map icon is encountered during program execution, execution proceeds to its flowline. When execution reaches the “bottom” of the Map icon’s flowline, execution proceeds to the icon following the Map icon.

This was demonstrated in the Tutorial - Map icons were used several times in it. For example, one was used to “contain” the icons used for each page.

3. The program should look like the diagram shown below:



It’s not necessary to remember exactly how the icons look - but if the correct icons aren’t drawn, names should be given (Calculated Path Decision icon, Display icon). And, it is necessary to show how they are attached to one another.

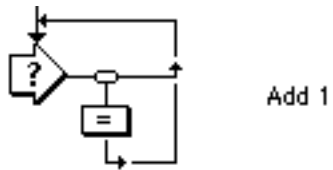
Set 2:

1. Normally, a Conditional Response can only be “matched” if some user action happens to cause program flow to proceed along the Interaction icon’s flowline (on which the conditional Response icon resides) and “hit” the Conditional Response icon. And, of course, to be matched, the condition associated with the Conditional Response would have to be true for the Conditional Response to be matched.

If, however, “Automatch” is turned on for the Conditional Response, if the condition associated with the Conditional Response becomes true, execution flow will proceed down the Interaction icon’s flowline (and match the Conditional Response when it is encountered) merely because of the truth of the condition (i.e. no user action is required.)

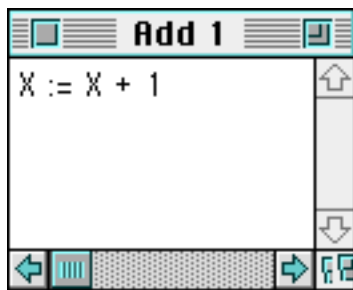
This was demonstrated in the tutorial - the Conditional Response had Automatch turned on so that the first page would automatically be displayed.

3. The program should look like this



It's not necessary to remember exactly how the icons are drawn, but if they aren't drawn, names should be given (e.g. Interaction icon, Calculation icon, Pushbutton Response icon). Furthermore, the Pushbutton Response should be called "Add 1", since this is what will appear on the button. And, it is necessary to show how the icons are attached to one another.

The window for the Calculation icon would look like this:



It's just necessary, of course, to give the contents of the window as an answer:

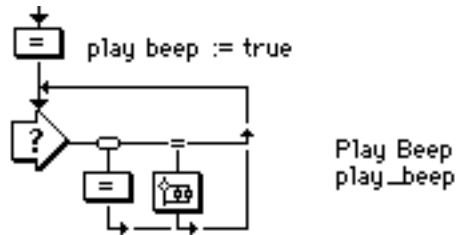
$X := X + 1$

Set 3:

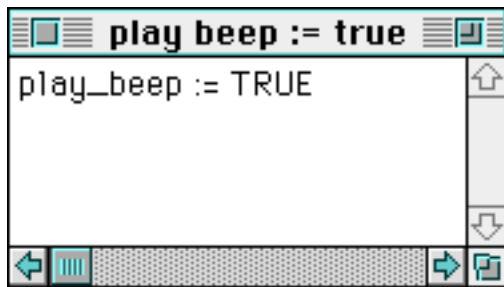
1. If a Pushbutton Response has the “Perpetual” option turned on, it will be available at any time during the program in which the Response is located (i.e. the Interaction Icon’s flowline (to which it is attached) does not necessarily have to be traversed to cause the Pushbutton Response to be matched.

This was demonstrated in the tutorial - the “Previous” and “Next” buttons were made perpetual so they would be active all the time (for example, during the playing of the movie or the sound.)

2. The program should look the one shown below:



The Calculation icon “play beep := true” window looks like this:



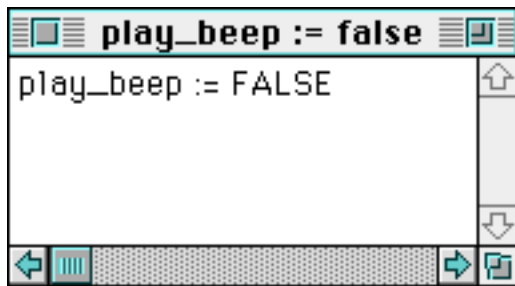
The Calculation icon “Play Beep” window looks like this:



The contents of the Group icon should look like this:



The contents of the window for the Calculation Icon titled “play_beep := false” should look like this:



Note: It's not necessary to exactly draw the icons. But, if they aren't drawn, names must be given (Interaction icon, Pushbutton Response icon, Conditional Response icon, Calculation icon, Map icon, Sound icon). However, it is important to show how the icons are connected to each other. Also, the only icon titles that must be as shown are “Play Beep”, “play_beep”, and “Beep”. And, of course, only the contents of the Calculation icon windows must be given - the windows themselves don't have to be drawn. (In the same vein, the Map icon's window doesn't have to be drawn, just the flowline with the appropriate icons on it.)

Vita

David Robert DeVaux was born in Frederick, Maryland on February 6, 1970. During his senior year of high school at Governor Thomas Johnson High School, he was employed by Galaxy Conferences as a computer programmer.

Upon graduation from high school in 1988, David enrolled at Virginia Polytechnic Institute and State University in Blacksburg, Virginia. During his undergraduate career, he worked summers as a programmer for the Navy Medical Logistics Command, located at Fort Detrick in Frederick. David obtained his Bachelor of Science degree in computer science and a minor in mathematics from Virginia Tech in 1992, graduating magna cum laude. He subsequently enrolled as a Master of Science candidate in the Computer Science Department at Virginia Tech. While obtaining his Masters degree, David served as a graduate teaching assistant.

David DeVaux is married to the former Jeannette Louise Jones. He is currently employed as a multimedia programmer at Interactive Design and Development in Blacksburg, Virginia.