

---

# Getting Interactive With Authorware 5 Attain:

---

Building Simulations and Games



---

Lloyd P. Rieber  
The University of Georgia — Athens

Copyright ©1999 Lloyd P. Rieber. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Lloyd P. Rieber.

Authorware Professional is a registered trademark of Macromedia, Inc.  
Director is a registered trademark of Macromedia, Inc.  
ClarisWorks is a registered trademark of Claris Corporation.  
PhotoShop is a registered trademark of Adobe Systems, Inc.  
QuickTime is a registered trademark of Apple Computer Co.

The reader bears total responsibility for the time, cost, and labor of downloading the electronic files comprising this book from the world wide web, for the subsequent time, cost, and labor of printing, duplicating, and binding of the text in printed form, and for any other miscellaneous costs that might be incurred. The reader is given no assurances that they will be able to successfully download, open, and print the electronic files.

While every effort has been made to ensure accuracy of the material contained in this text, the reader agrees not to hold the author liable for any errors. Readers are encouraged to check Lloyd Rieber's web site periodically for updates to these chapters: <http://www.nowhereroad.com>

For every full or partial copy made of this text, electronic, printed, or otherwise, the reader agrees to pay the author the specified royalty amount in U.S. dollars. Contact Lloyd Rieber directly for more information via email ([LRIEBER@COE.UGA.EDU](mailto:LRIEBER@COE.UGA.EDU)) or this address:

Lloyd Rieber  
6114 Nowhere Road  
Hull, Georgia 30646  
USA

This page containing the copyright notice must be included as the first page in each legally reproduced copy.

Version: June 14, 1999

---

# Contents

---

<b>Preface .....</b>	<b>viii</b>
----------------------	-------------

---

<i>What is Authorware anyway?</i>	<i>ix</i>
<i>Who should learn Authorware?</i>	<i>x</i>
<i>The purpose of this book</i>	<i>xi</i>
<i>The project approach</i>	<i>xi</i>
<i>About the projects</i>	<i>xii</i>
<i>Why the emphasis on games and simulations?</i>	<i>xiii</i>
<i>What this book won't teach you</i>	<i>xiii</i>
<i>How to use this book</i>	<i>xiv</i>
<i>A word about platforms and versions</i>	<i>xv</i>
<i>An experiment in publishing</i>	<i>xv</i>
<i>A brief Authorware history lesson (that you can safely skip)</i>	<i>xvi</i>
<i>Acknowledgments</i>	<i>xviii</i>

<b>Chapter 1: Slide Show .....</b>	<b>1</b>
------------------------------------	----------

---

<b>Authorware as a Presentation Tool</b>	<b>1</b>
<b>The Problem</b>	<b>1</b>
<b>Introduction</b>	<b>2</b>
<b>Getting started</b>	<b>4</b>
<i>A word about Knowledge Objects</i>	<i>4</i>
<i>Say hello to the flow line</i>	<i>5</i>
<i>The tool box</i>	<i>8</i>
<i>Adding background colors to the display</i>	<i>13</i>
<i>Fine-tuning the position of the screen objects</i>	<i>19</i>
<i>Saving the file</i>	<i>21</i>
<b>Finishing the title sequence</b>	<b>21</b>
<i>Adding the subtitle</i>	<i>21</i>
<i>Running the program</i>	<i>23</i>
<i>Setting up the second subtitle</i>	<i>27</i>
<i>Modifying the Wait icon</i>	<i>35</i>
<i>Adding animation and sound to the title sequence</i>	<i>37</i>
<i>Adding some sound</i>	<i>43</i>
<i>Grouping the icons in the title sequence</i>	<i>46</i>
<b>Time out</b>	<b>49</b>
<i>What's next?</i>	<i>50</i>
<b>Constructing the slide shows</b>	<b>50</b>
<i>Adding hot spots to the display</i>	<i>56</i>
<i>Positioning the hot spots</i>	<i>58</i>
<b>Using the start flag</b>	<b>59</b>
<b>Adding slides</b>	<b>60</b>
<i>Adding a second slide</i>	<i>67</i>
<b>Rearranging your slides</b>	<b>71</b>
<i>A nonlinear slide tray</i>	<i>72</i>
<i>Testing the gaming slides</i>	<i>77</i>

<b>Improving the user interface</b>	<b>79</b>
<b>Using digital movies</b>	<b>81</b>
<b>Summary</b>	<b>88</b>
<b>Other projects</b>	<b>88</b>
<b>Knowledge Object Activity</b>	<b>88</b>
<b>Chapter 2: Basketball Camp .....</b>	<b>91</b>
<hr/>	
<b>Building a simple simulation with data-driven animation</b>	<b>91</b>
<b>The Problem</b>	<b>91</b>
<b>Introduction</b>	<b>92</b>
<b>Getting started</b>	<b>92</b>
<b>Setting up the displays</b>	<b>93</b>
<b>Setting up the data-driven animation for the offensive player</b>	<b>98</b>
<i>Creating variables</i>	98
<i>Creating data-driven animation</i>	101
<b>Setting up the data-driven animation for the defensive player</b>	<b>110</b>
<i>Creating variables</i>	110
<b>Concurrent animation</b>	<b>115</b>
<b>Making the animation run continuously in a loop</b>	<b>116</b>
<b>Summary</b>	<b>119</b>
<b>But is it interactive?</b>	<b>119</b>
<b>Other projects</b>	<b>120</b>
<b>Knowledge Object Activity</b>	<b>120</b>
<b>Chapter 3: Mystery Number .....</b>	<b>125</b>
<hr/>	
<b>Building a simple game with data-driven animation</b>	<b>125</b>
<b>The Problem</b>	<b>125</b>
<b>Introduction</b>	<b>126</b>
<b>Getting started</b>	<b>127</b>
<i>Setting up the file structure</i>	128
<i>Setting up the game structure</i>	129
<b>Building the interaction</b>	<b>133</b>
<i>Setting up a text entry response branch</i>	134
<i>Setting up a push button response branch</i>	135
<i>Setting up the Interaction icon's display</i>	137
<i>Showing embedded variables in displays</i>	138
<i>Adding the final branch</i>	141
<i>Changing the flow</i>	143
<b>Completing the game</b>	<b>145</b>
<i>Performing a quick test</i>	148
<i>Debugging the file</i>	153
<b>Constructing data-driven animation as feedback</b>	<b>155</b>
<i>Testing the game</i>	162
<b>Mopping up</b>	<b>162</b>
<i>Adding a final frame</i>	163
<i>Fixing one final bug</i>	166
<b>Summary</b>	<b>167</b>
<b>Other projects</b>	<b>167</b>

## Chapter 4: Space Shuttle Commander ..... 169

---

<b>Building a physics simulation</b>	<b>169</b>
<b>The Problem</b>	<b>169</b>
<b>Introduction</b>	<b>170</b>
<b>Getting started</b>	<b>170</b>
<i>Changing the background color</i>	172
<i>Setting up the game structure</i>	173
<b>Constructing the shuttle display and its animation</b>	<b>177</b>
<b>Setting up the simulation’s “mathematical engine”</b>	<b>182</b>
<i>Testing and calibrating the simulation loop and mathematical engine</i>	184
<b>Making the simulation interactive</b>	<b>186</b>
<i>Finishing the interaction</i>	191
<i>Creating a “wrap-around” universe</i>	193
<i>Improving the user interface</i>	195
<b>Adding a game context to the simulation: Rendezvous</b>	<b>197</b>
<i>Creating a space station</i>	198
<i>Constructing the “final frame”</i>	203
<i>A mathematical approach to determining if the displays overlap</i>	205
<b>Next steps</b>	<b>209</b>
<b>Summary</b>	<b>210</b>
<b>Other projects</b>	<b>211</b>

## Chapter 5: Amazing Mazes ..... 213

---

<b>Building a Maze Game</b>	<b>213</b>
<b>The Problem</b>	<b>213</b>
<b>Introduction</b>	<b>214</b>
<b>Getting started</b>	<b>214</b>
<i>Setting up the game structure</i>	216
<b>Setting up the game background and maze objects</b>	<b>221</b>
<i>Constructing the maze blocks</i>	222
<i>Constructing the maze gates and maze exit</i>	226
<b>Constructing the player’s game piece</b>	<b>229</b>
<i>Setting up the animation for the player’s game piece</i>	230
<i>Resizing the player’s game piece</i>	235
<b>Setting up the player’s game controls</b>	<b>235</b>
<i>Testing the interaction and animation</i>	241
<b>Setting up the game engine loop</b>	<b>242</b>
<i>Programming Authorware to check for overlapping</i>	243
<b>Maze gates</b>	<b>249</b>
<i>Branching to a question at just the right time</i>	249
<i>Constructing a generic question as a template</i>	254
<i>Adding the feedback and actions</i>	263
<b>Adding the timer</b>	<b>270</b>
<b>Debugging the game</b>	<b>275</b>
<i>“Disarming” the game buttons</i>	276
<i>Telling Authorware to ignore unprocessed keystrokes</i>	280
<b>Next Steps</b>	<b>282</b>
<b>Summary</b>	<b>283</b>
<b>Other projects</b>	<b>283</b>

---

<b>Special techniques to enhance your Authorware programs</b>	<b>285</b>
<b>Technique 1. Giving the user control over sound</b>	<b>286</b>
<i>Introduction</i>	286
<i>Getting started</i>	286
<i>Giving the user control over the sound</i>	290
<i>Turning the sound back on</i>	293
<i>Improving the user interface</i>	295
<i>One last sound trick</i>	296
<b>Technique 2. Rotating screen displays</b>	<b>297</b>
<i>Introduction</i>	297
<i>Getting started</i>	298
<i>Setting up the interaction</i>	302
<i>Testing the program</i>	305
<i>Getting the airplane to fly around the screen</i>	305
<i>Time for another test</i>	308
<b>Technique 3. Creating a “click and hold” response type</b>	<b>309</b>
<i>Introduction</i>	309
<i>Getting started</i>	310
<i>Setting up the user interaction</i>	313
<i>Testing the response branch</i>	316
<i>Modifying the screen coordinates to match the movie</i>	317
<i>Improving the user interface</i>	319
<i>Next steps</i>	321
<b>Technique 4. Building a response palette that users can move</b>	<b>323</b>
<i>Introduction</i>	323
<i>Getting started</i>	324
<i>Setting up variables</i>	325
<i>Creating a hot spot with a relative position</i>	326
<i>Testing the program</i>	334
<i>Updating the variables when the user moves the palette</i>	335
<i>Next steps</i>	338
<b>Technique 5. Creating slide bars and other movable data objects</b>	<b>339</b>
<i>Introduction</i>	339
<i>Getting started</i>	340
<i>Manipulating data with the slide bar</i>	344
<i>Transferring this data to the variable “temp”</i>	346
<i>Using the data to alter conditions in a simulation</i>	349
<i>Another strategy</i>	350
<b>Technique 6. Collecting and analyzing data with a spreadsheet</b>	<b>351</b>
<i>Introduction</i>	351
<i>More about the data structure</i>	353
<i>Getting started</i>	354
<i>Constructing the survey’s file structure</i>	356
<i>Constructing the survey questions</i>	357
<i>Setting up a multiple-choice question for gender</i>	358
<i>Setting up a text entry question for age</i>	360
<i>Setting up a text entry question for game playing</i>	363
<i>Concatenating the survey data</i>	365
<i>Testing the program</i>	366
<i>Saving the data to disk</i>	367
<i>Thanking the respondent and restarting the file</i>	368
<i>Checking the data file, then copying and pasting it into a spreadsheet</i>	371

<b>Appendix A: Packaging .....</b>	<b>373</b>
<b>Appendix B: Libraries .....</b>	<b>380</b>
<b>Appendix C: Jumping to other files and applications.....</b>	<b>383</b>
<b>Appendix D: Compacting, printing, memory, cross-platform .....</b>	<b>387</b>
<b>Appendix E: Shockwave .....</b>	<b>392</b>
<b>Appendix F: Copyright.....</b>	<b>398</b>
<b>Index .....</b>	<b>401</b>

# Preface

---

Frankly, I had always been wary of writing a book like this one. I had come close to doing it many times in the past, but writing any “how to” computer book is full of perils. For example, once begun I felt sure that the software would become obsolete just at the moment I finished. But several recent events finally convinced me to do it.

The first concerns Authorware itself. This authoring tool has both a respectable history and relatively safe future — a rare combination in the computer industry. It has been used by developers since 1987 and the recent significant upgrade to version 5.0 is good evidence of Authorware’s stability. Second, as a member of the faculty here in the Department of Instructional Technology at the University of Georgia I am continually looking for ways to improve my teaching. Authorware is and will continue to be one of the most important tools learned by students here at UGA. Our courses expect a great deal from students, many of whom are juggling full-time jobs and families. For example, for years I taught a class where students were expected to learn Authorware from scratch well enough to produce a stand-alone piece of courseware in the span of about 8 weeks (with two weeks for field testing and documentation preparation). Not surprisingly, the best that most students could muster was a narrowly designed tutorial, heavy in content but shallow in meaning and experience for the person for whom the software was intended. I decided time had come to write a book where students could learn about the concepts of interactive design while learning a powerful authoring tool in a much more flexible way than is possible in traditional course-based instructor-led approaches. Also, our department recently reconceptualized the entire curriculum leading to a masters degree in instructional technology as our university made the transition from 10-week quarters to 15-week semesters. We have tried is to restructure the multimedia development courses in such a way as to get students learning about interactive design in more authentic ways. To pull this off requires more adequate and flexible learning resources. This book figures prominently in this effort.

My final motivation for writing this book deals with my current interest in the concept of “learning by designing.” I feel that the design process is a powerful mediator for learning. In a project called “KID DESIGNER,” I have worked

*Our new curriculum is structured around a studio model, similar to that used in schools of art and architecture. To learn more, visit our web site at: <http://itech1.coe.uga.edu/studio>*

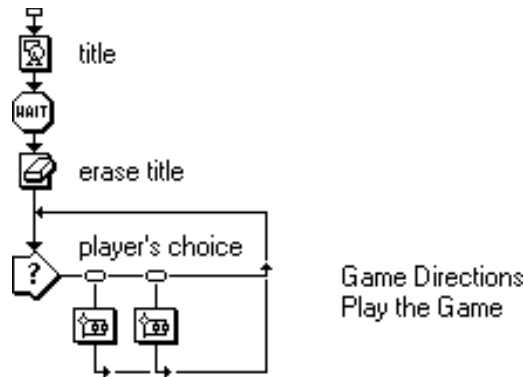
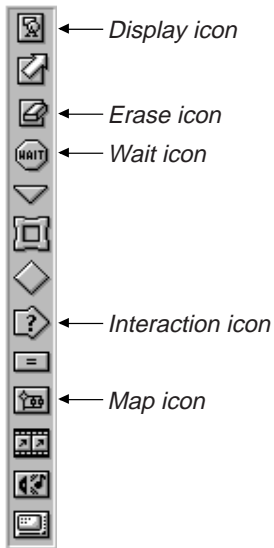
with elementary school children as they designed their own computer games about subjects they were learning at school. The games produced so far have been in science, language, mathematics, mythology, and electricity. These children did not learn Authorware. Instead, my graduate students and I worked for the children as their programmers. They designed the game goals and rules, created the graphics, wrote game questions and game directions, etc. and we put the pieces together using Authorware. (You can learn more about this project, and even download the children's games, by reading a recent article about the project at "<http://www.ncsu.edu/meridian/jan98/index.html>"). In the next stages of the project, I hope to get high school students working together with the elementary school students. The idea is for the older students to handle the programming of the younger children's game ideas. For that reason, the projects in this book are also meant to be able to be completed by students as young as 15 or 16. The idea that Authorware could be used by such a wide range of ages for such different purposes speaks to the concept of "no floors, no ceilings" — good ideas for learning span many boundaries. Most anyone can begin using this tool and continue to use it in increasingly sophisticated ways.

*Prominent writers in this field include David Perkins, Seymour Papert, Idit Harel, and Mitch Resnick.*

## **What is Authorware anyway?**

The simplest answer is that it is a computer authoring tool designed for those who want to produce educational software. Despite the fact that the word "programming" has largely fallen out of favor, you essentially use Authorware to program computer software. Most writers today use the term "authoring" as a euphemism for programming. This is not without justification. In traditional programming languages, one has to learn commands (i.e. vocabulary) and the rules to use those commands (called syntax). Programming in Authorware is different. Instead of typing commands line after line, one creates a flow chart where each symbol performs a specific function. In Authorware, a flow chart is not just a tool for designing a program, it actually *is* the program! Consequently, more attention can be placed on the *design* of the program and less on the mechanics of programming.

For example, even if you have never used Authorware before, you can probably make some sense out of the following graphic:



You build flow charts like these simply by “dragging and dropping” icons from the palette on the left. For example, this little segment begins by displaying the game’s title page. The program then waits until the player clicks the mouse button, after which the next icon erases the title screen. The next icon, an Interaction icon, allows the player to make a choice between reading the game directions and playing the game via two buttons that are automatically displayed on the screen when the program is run. The two Map icons are used to group or hold all of the necessary icons that make up the game directions and the game itself (including, probably, more Map icons). Notice how the flow chart clearly shows that the player will again be given these two choices after reading the game directions or completing the game (just follow the flow line as it exits each of the two Map icons).

## Who should learn Authorware?

The “official” answer is that Authorware is intended for instructional designers and developers interested in creating computer courseware. My answer is that it is perfect for anyone wanting to take control of the computer to design their own creations. Yes, it is a great tool for teams of educators, subject matter experts, and programmers wanting to design instructional materials for students, but it’s also a great tool for anyone who wants to enter the creative world of designing interactive computer software. It’s also a great tool just to have fun with — it can be very satisfying to see one of your creations come alive on the screen. Of course, this takes time, patience, and perseverance. Some people insist that they cannot learn programming because they are not the “logical, analytic, left brain type.” Actually, the “hard core, top-down, design everything first” way of programming is but one approach. The best designers I know are also highly creative and imaginative (and not all are adults — I have seen children as young as 10 use Authorware competently). In fact, I would argue that the so-called “right brain” attributes (like creativity) are far more important for anyone wishing to be a designer of educational software. The

*The term “courseware” generally refers to any instructional computer software — software intending to teach somebody something.*

easy part is learning the “how to’s” of programming, the hard part is putting it to use. If you want to learn Authorware, are willing to invest some time and effort to do so, and won’t bail out at the first sign of trouble, then I am confident that you *will* learn it. On the other hand, I’m equally convinced that you can talk yourself into believing that you can’t. It’s your choice.

## **The purpose of this book**

This book is meant to help you to learn Authorware well in a minimum amount of time. Although one can learn to do some things very quickly with Authorware, it is still important to learn and master fundamental programming concepts, such as variables and functions, in order to go beyond creating “click and browse” courseware. Rather than saving these features for later, this book has you put them to use right away. Hopefully, you will begin to understand Authorware’s rich collection of interactive features and capabilities from being immersed in their use at the very start.

This book uses what some refer to as the “project-based approach.” You will learn how Authorware works by constructing a series of fully functioning projects. Each chapter presents one project designed to be completed in one or two sittings (1-2 hours). These projects are meant to be fun, intriguing, and, when finished, will act as working demonstrations of the most sophisticated levels of Authorware programming.

## **The project approach**

The project approach simply means that one learns through building complete, stand-alone projects that are interesting and have clear goals and expectations. Good projects also have an easy-to-understand structure that makes sense at the start. Games and simulations make for good projects. One might say that this book was written much in the spirit of Norm Abrams on the American public television show “The New Yankee Workshop.” Norm is a master carpenter who shows his TV audience how to build from scratch a piece of furniture or other woodworking project. Norm always starts by carefully choosing a project worthy of the effort, often visiting American colonial sites for inspiration. Once he finds a suitable subject, he then builds the piece twice — once for himself (plus to show the audience at the start what the finished product will look like) and again in front of the camera. The design that Norm follows usually includes many changes from the original, mostly to keep the construction within the reach of the amateur carpenter at home.

Likewise, all of the projects in this book were chosen from among dozens of games and simulations I have built for teaching, research, and just plain personal enjoyment over the last 20 years or so. The projects I finally chose were among the most successful in helping others understand

how to program with Authorware as well as understand some fundamental concepts in designing simulations and games. It turns out they are also among the most fun to show and build. I reconstructed the projects in a way that a novice could understand and build them quickly within one or two sessions. After all, the main purpose of this book is, of course, to learn Authorware. So, the chapters were written as I built the projects myself again from scratch, much like Norm in front of the camera. I was careful to note every keystroke and mouse click (with words and pictures) so that you could retrace my steps to build the same projects yourself.

## About the projects

Here are brief descriptions of the projects you will build (note that the simulation/gaming projects start with chapter 2):

**Chapter 1: Slide Show.** This chapter demonstrates how to use Authorware as a flexible and easy-to-use presentation tool; meant as a fast introduction to Authorware. It ends with a brief introduction to one of the latest features of Authorware — Knowledge Objects.

**Chapter 2: Basketball Camp.** A simple animated simulation of one of the fundamental rules of basketball defense: keep yourself between the ball and the basket. Good introduction to the use of data-driven animation. This chapter also ends by considering how to use another Knowledge Object to make the simulation interactive.

**Chapter 3: Mystery Number.** A "guess the number" math game built with data-driven animation. (Remember playing the 'hot/cold' game as a kid?)

**Chapter 4: Space Shuttle Commander.** A simulation of Newton's laws of motion. Pretend you are piloting the space shuttle!

**Chapter 5: Amazing Mazes.** A template for games that use mazes; players have to answer questions as they proceed past numerous gates laid out on a maze background (programmed so that you can't cheat). Also includes directions on how to program a timer as a scorekeeping feature.

**Chapter 6: How'd they do that?** A collection of various interactive features you might like to have in your software. Examples:

- Allowing users to turn a program's sound on and off;
- Rotating screen objects by manipulating digital movies constructed with BMP sequences;
- Building a "click and hold" interaction response type;
- Building "response palettes" that users can move freely around the screen;
- Creating "slide bars" that control data functions;
- How to save a program's data in a format that can be opened and analyzed by a spreadsheet.

## **Why the emphasis on games and simulations?**

Starting with chapter 2, all of the projects you will build can be classified as either games or simulations (or both). There are two main reasons for the decision to focus on gaming and simulation. First, constructing a computer game or simulation requires mastery of many programming concepts. Despite the tendency to think of a game as “child’s play,” the programming of a game is among the most sophisticated of all computing projects. You’ll be amazed at how much Authorware you will learn just by correctly programming the simplest of games. While games are demanding in terms of programming expertise, they are also at the same time fun to construct. You get instant feedback while you work. Once you know how a game is supposed to be played, you know within seconds if someone (or the computer) is “breaking the rules.” Hence, you will know if your programming works just by playing the game (e.g. if a screen object goes through a wall instead of bouncing off of it as it should, you know where the programming “bug” is to be found).

But perhaps the most important reason for the emphasis on games and simulations is that I believe they remain among the most compelling learning environments yet devised. I think this is especially true for games. Through all my experiences as a teacher, researcher, and learner, I keep coming back to the creative use of gaming features as a potent learning strategy for all people. Well-designed games come as close as anything I have found to matching all of the complex requirements for a successful learning environment (including the cognitive, motivational, and social components). Arguments in favor of gaming can come from both the perspective of a teacher and student. The traditional strategy is to give students educational games as a learning approach. However, I find the idea of students constructing their own games to be far more compelling. Indeed, much of the current thinking in the field of instructional technology centers around “learning by building” or “learning by designing.” Game design is perfectly suited to this point of view.

## **What this book won’t teach you**

While no prior experience with Authorware is assumed, you are expected to already have much experience using a computer. This book will not teach you about general functions of the computer or other applications. In particular, you are expected to be completely comfortable with the Windows operating system. Although all necessary steps involved in manipulating files, icons, and graphics will be described thoroughly in this text, these explanations assume you already have experience with basic computing concepts (e.g. computer memory, file/disk management and security, etc.) and the graphical user interface (GUI) techniques (e.g. “copying and pasting” within and between applications).

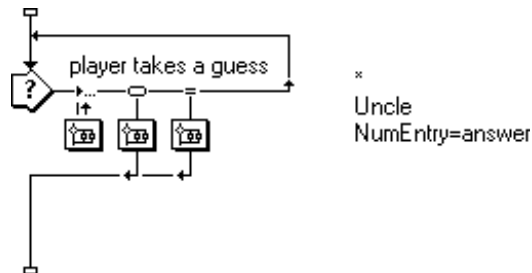
## How to use this book

In each chapter, you will be asked to perform a series of steps to complete the project. Each step to be performed will be preceded by a check box, such as this:

- Click once on the Map icon titled “game directions”:**



Follow the instructions carefully and check off the step when it is completed. This will help you from getting confused and disoriented. In the event that you make a mistake, these check boxes will help you to retrace your steps. Explanations and illustrations will accompany these steps. The illustrations represent what you should actually be seeing on your screen. You should strive to ensure that critical elements of illustrations match exactly what you see on your screen, such as the directions of the arrows showing the flow of a particular group of icons:



However, you will encounter lots of situations in which general features of illustrations will never exactly match what you see on the screen, such as the relative positions of windows. The graphical user interface (GUI) of Authorware results in all kinds of variations that have no bearing on the final result. Rest assured that the text will try to alert you to those times when absolute accuracy is required.

Of course, it is possible that you will follow and check off each and every step and still feel hopelessly lost at some point. When the inevitable happens, try not to panic. Instead, carefully retrace your steps starting at the last point where things seemed to be working. Try to approach these situations with the attitude of a creative problem-solver confronted with a challenge.

Finally, special instructions and reminders, plus informal commentary on design issues will be provided off to the side.

*Look for some lighthearted commentary over here. I'll try to give you some insights into my own ideas and experiences about design without boring you.*

## **A word about platforms and versions**

This book was written exclusively for Authorware 5 Attain. If you are using an earlier version of Authorware (e.g. versions 2.0, 3.x, or 4.0), consult my web site (<http://www.NowhereRoad.com>) for information about an edition of this book specifically written for those versions. Although the logic of the projects is essentially the same, there are significant differences in the interface design of each version. It is also important to note that starting with Version 5, Authorware is only available for the Windows platform. However, once a file is authored, it can be repackaged to run on Macintosh systems (this is explained in full in Appendix D).

Version 5 comes with significant improvements and enhancements. One of the most significant is the addition of Knowledge Objects, special Authorware icons that include step-by-step instructions, called wizards, which guide you through the process of creating sophisticated programming techniques. With a good basic understanding of Authorware, Knowledge Objects can greatly facilitate the production of interactive educational multimedia.

## **An experiment in publishing**

As any graduate student will tell you, books are overpriced. Of course, a publisher will quickly point out that books are expensive to produce (with paper the largest single cost). Authors will complain that their piddly royalties often don't justify the hard work. Surely, there must be a better way. I think there is and it's called the Internet.

With this book, I am experimenting with the world wide web as a distribution outlet. More than likely, you yourself downloaded and printed this preface from my web site (<http://www.NowhereRoad.com>). If you were given a photocopy of the book to use (perhaps in a course you are taking), probably your instructor downloaded and printed it. (Indeed, the fact that you might be reading a photocopy of the book is no cause for concern — the quality is the same, if not higher, than in a “real” book.)

*I really live on Nowhere Road — this is no joke!*

Transferring the costs and labor of printing to the end user allows me to offer this book at a low cost which is still fair to me as its writer. I also expect the Internet to allow more access to the book by more people at the time they really need it (e.g. “Our development team just bought a copy of Authorware, now what?”). I also hope the electronic format of the book will allow me to continually update and improve the book. We'll see.

## A brief Authorware history lesson

(that you can safely skip)

When the Macintosh came on the scene in 1984, it quickly revolutionized many aspects of computing, most notably desktop publishing. Although it fast became the preferred platform for many people working exclusively with computer tools such as word processing and graphics, its use as a platform for computer-based learning environments was slow to catch on. At that time, the Apple II family of computers was the platform of choice for instructional software (drill and practice, tutorials, games, and simulations). The Apple II had been around for about 5 years and IBM had yet to make much of a dent in the educational market. The Apple II came with a built-in programming language — Applesoft BASIC. Granted, learning BASIC was no picnic, but it did allow a lot of people to become a special kind of “edutech” mutant — part educator and part computer hobbyist. It also allowed, for better or worse, the chance for elementary and high school students to get experience in computer programming (along with languages such as LOGO and PASCAL).

In the Macintosh world, in contrast, there just weren’t any accessible means that an average Joe or Jane could do courseware development for the first few years of its existence. Ironically, the Mac’s graphical user interface made programming unruly for everyone but the most dedicated and skilled programmers. Little was available for “just plain folks” like us. Colleges and universities that offered courses or curricula in educational computing tended towards labs of Apple IIe or IBM-PC computers for most course activities and clusters of Macintoshes for graphics, desktop publishing, and special demonstrations of emerging commercial courseware specially suited to its graphical interface. The advent of HyperCard (originally called WildCard) in 1986 was a significant milestone. With it, one could easily construct hypertext and hypermedia — textual and graphical documents in which users could explore in a nonlinear fashion. Most of the HyperCard software that still gets developed resembles easy-to-navigate databases of information in specific content areas. Of course, one can also develop educational games and simulations with HyperCard, but learning a significant amount of programming (called scripting) is necessary. When Authorware was first released around 1987, it represented something very new.

I first started using Authorware in 1988 when it was called *Course of Action*. It was the principal product of a new company (called Authorware, Inc.) headed by Michael Allen. Its creative use of a graphical programming environment based on an interactive flowchart was amazing to many of us. Although the term “multimedia” had yet to cause people’s eyes to roll back into their heads, Authorware made the integration of text, graphics, animation, sound, and video virtually seamless. Most remarkable to me was the range and

depth of interactions that one could produce (with surprisingly little “programming” in the traditional sense). However, the first few versions of Authorware were slow and cantankerous. (The first version I used was very prone to crashing.) Implementing one’s courseware was also awkward. You actually had to buy “student disks” from the company in order to distribute the software you created. Fortunately, the company soon shook off the quirks in both the product and their approach. Those who get into educational computing today and start with programming tools such as Authorware fail to appreciate its contrast with the programming tools available up to the time of its release.

Authorware and HyperCard were long viewed as tools in competition. This was never an accurate view. Both did different things well and other things poorly. If both were screwdrivers, one was a phillips and the other a flathead. HyperCard was designed for hypertext (with graphics) whereas Authorware was designed to build instructional courseware. Each could do the other, but only with a struggle. On pure economic grounds, there was no competition — HyperCard was literally given away by Apple with every Macintosh that was sold. Many people bought a Macintosh just to get HyperCard. It was truly a “Killer App”! Authorware, in contrast, was (and still is) extremely expensive, even for the academic community. Single copies of Authorware originally cost about \$8000 for corporate customers. Higher education had it a little better — my university department shelled out over \$2000 for our first copy back in 1988. Fortunately, prices have fallen substantially since that time.

Then a couple of things did and didn’t happen in the early 1990’s that gave Authorware a decided edge. First, Authorware was bought out by Macromedia. A lot of us had seen software tools come and go quickly on the Macintosh. Macromedia’s purchase seemed to assure us that Authorware would be around for awhile. Plus, Authorware was now supported by a company with the resources and reputation for serious multimedia development (Director is one of their other well known authoring products). Next, despite persistent rumors to the contrary, a complete overhaul of HyperCard never happened. In contrast, Authorware upgrades were fairly significant improvements (the latest upgrade to 4.0 is a good case in point). Another important difference was Authorware’s commitment to cross-platform development and delivery. Up to version 4.0, one could develop courseware on either the Macintosh or Windows platform and convert to the other with little or no reprogramming (although serious cross-platform development still demands much planning). Things have changed. Starting with version 5.0, Macromedia decided to make Authorware a Windows only application. Frankly, this was disappointing news to those of us with strong ties to K-12 schools (and those of the opinion that the Macintosh is a superior platform for multimedia development). Fortunately, due to a strong response by the Macintosh community of Authorware users,

*Prices and versions have fluctuated quite a bit, though. In 1994 a “real” version of Authorware was released specifically for individual educators at a reasonable price, called Authorware Academic. This meant that ordinary classroom teachers might be able to get a copy or two of Authorware for their classrooms, not to program courseware for their students, but to give to their students to design and program their own projects. Getting powerful tools like these in the hands of students is an exciting concept that I want to support. Unfortunately, Macromedia decided recently to stop publishing Authorware Academic — a poor decision, in my opinion.*

Macromedia has made available a Macintosh Packager program that allows files created on the Windows platform to be easily converted to Macintosh (see Appendix D for details). Authorware has kept up with advances in the computing industry while retaining virtually the same interface that it had in 1987. Among the most notable advances is its continued compatibility with Macromedia's ShockWave technology permitting courseware developed with Authorware to be distributed over the Internet.

*Fortunately, and ironically, the Macintosh is rather Windows-friendly. For example, I wrote this edition using an iMac with Virtual PC installed. Authorware 5 ran great.*

Authorware is not perfect, however. There are still lots of things that frustrate developers (besides its cost). The decision to end its affiliation as an authoring tool on the Macintosh figures prominently on the list. To those of us interested in simulation and game design, our wish list includes the ability to rotate screen objects, scale objects (i.e. increase or reduce a graphical object's size through programming), and improved collision detection between dynamic screen objects. Another example is Authorware's inability to allow easy user interaction with text fields. There is also the misconception that one can pull Authorware out of the box and start developing sophisticated courseware immediately. I think this perception was actually promoted (inappropriately and perhaps unintentionally) by the company early on. Yes, in contrast to programming in BASIC, PASCAL, or C, the learning curve is slight. But believe me, there *is* a learning curve to Authorware. There are two consequences to this attitude. The first is just the general frustration of not being able to do the things promised right away. The second is the tendency to create courseware that merely scrapes the surface of what is possible with Authorware. With some devoted time and effort, one can create very imaginative and interactive courseware with Authorware. But there is no such thing as a free lunch here either — you get out of it what you put into it.

## Acknowledgments

Although the number of people to whom I need to express my thanks continues to grow, I'd like to thank several people here. First, I want to thank the many students here at the University of Georgia who patiently struggled through early drafts of each edition of this book and helped spot an amazing number of errors that I missed. I also thank Simon Hooper, a friend and colleague at the University of Minnesota, for his helpful encouragement. Simon is also a creative Authorware user, teacher, and writer. I'd also like to thank someone I have never met — George Beekman. My first attempts at learning HyperCard years ago were very successful mainly due to a book that George wrote called *HyperCard in a Hurry*. I found his approach to be very effective. The step-by-step project approach used here largely mimics his style.

I'd also like to thank the following individuals who patiently tried out early drafts of the chapters and provided me with

valuable feedback: Brett Bixler, The Pennsylvania State University; John Braidwood, Huntly College, Hamilton New Zealand; Gail Fitzgerald, University of Missouri-Columbia; and Pam Miller, University of Pretoria, South Africa.

Finally, special thanks go to my daughter, Becky. In 1996 (when she was 17 years of age), she graciously served as the book's first beta tester. Her experiences and comments on early drafts of this book have been extremely helpful. She has also reinforced my idea that Authorware is a wonderfully creative tool for anyone to learn and should not be reserved only for professional designers and developers. Expert Authorware programmers don't have to have letters after their names.

LPR  
June 14, 1999

*Notes:*