Authoring Interactive Media

Brian P. Bailey and Joseph A. Konstan

Department of Computer Science and Engineering

University of Minnesota

4-192 EE/CS Building

200 Union Street SE

Minneapolis, MN 5545

Introduction

Multimedia presentations utilize content derived from a variety of information sources in order to convey a message to a viewer. Examples of multimedia content include recorded and synthesized audio, video, still images, bitmap and vector graphics, animations, and text. Although a message could be conveyed through any one of these media, combining them into a synchronized, coherent presentation creates a more appealing, compelling, and entertaining display. For example, watching an entertaining television commercial, which creatively uses synchronized audio, video, and special effects, creates a more compelling presentation than a print advertisement or radio commercial. A well-designed multimedia presentation continually makes use of the viewer's auditory and visual senses through dynamically changing displays, and as a result, offers a more interesting and effective medium for conveying information (19, 20, 33).

The difference between a non-interactive and interactive multimedia presentation involves the participatory role of the viewer. In a non-interactive presentation, the role of the viewer is that of a passive observer. The viewer is external to the display of information and is forced to experience it from the beginning to the end; e.g., a television viewer can neither skip to the end of a show to see how it ends, nor influence the way it ends. In an *interactive* presentation, the role of the viewer is transformed from a passive observer to an active participant enabling a unique, individual experience. To create this unique experience, an interactive presentation may allow the viewer to actively navigate among its content, periodically respond to questions, or play a more central role such as controlling the main character of a virtual murder mystery (7). Regardless of the interaction technique, it is the inclusion of the viewer as an active participant that differentiates an interactive from a non-interactive multimedia presentation. For example, in the popular adventure game Myst (32), a viewer controls their path through a mysterious island, and along the way interacts with various objects in order to unlock the secrets of the island. Interactive multimedia has successfully been applied within many application domains such as computer-based training, instructional manuals, distance learning, electronic books, corporate presentations, and computer games. Thus, the term *interactive multimedia presentation* collectively refers to each of them, rather than to a specific application domain.

The goal of this article is to define the processes, roles, and tools used to build an interactive multimedia presentation. *Authoring* is the process of designing and implementing an interactive presentation. The *author* collectively refers to the persons fulfilling roles throughout the authoring process. And *authoring tools* are the software and hardware tools used by an author to design or implement a presentation. In addition to further explaining these concepts, this article also provides a discussion of three popular authoring tools; HyperCard, Director, and Authorware. The history and authoring environment of each tool is discussed, and then the use of each tool is demonstrated by authoring a sample presentation. Finally, this article concludes with a discussion of current research shaping the future of interactive media authoring.

The Author

Although the term author usually implies a single person, it is rare that a single person designs and implements an innovative interactive presentation. In fact, designing a presentation usually requires a great deal of time, skill, and effort from a diverse community of professionals including writers, graphic artists, user interface designers, content editors, and programmers. Each profession offers a unique talent and perspective, that when combined, produces innovative presentations well beyond that which any one individual could produce alone. This diverse set of professionals involved in the authoring process is commonly and collectively referred to as *author*.

The Authoring Process

Authoring an interactive multimedia presentation is a process that combines creative talent along with computer expertise. The creative talent is used to design and organize media content and to provide innovative interaction. The computer expertise is used to transform these creative ideas into a working presentation. As with learning other skills, learning to author an innovative interactive multimedia presentation requires a great deal of practice and experience.

Authoring an interactive presentation consists of several tasks. First, an author must clearly define the intended audience and the message that needs to be conveyed to them. Second, an author must design the media content necessary to convey the intended message. Third, an author must organize the media content in terms of both space and time. Finally, an author must design the interaction with the viewer, which may affect both content design and organization.

Define the Intended Audience and Message

Every multimedia presentation must have an intended audience and a clearly defined message. An author must characterize the audience and then define what information its members should come away with after viewing and interacting with the presentation. When characterizing the audience, several criteria should be considered such as the average age, relevant computer experience, content familiarity, and viewing environment such as individual or group. For example, an author must determine if the audience will consist of grade school students, professional businesswomen, or college undergraduates. After the audience is characterized, an author must define the message that needs to be conveyed to its members. For example, an appropriate message for grade school students could be how to multiply and divide numbers involving more than one digit. For businesswomen, an appropriate message could be how to use a new software product at the workplace. Once the audience and message have been defined, the goal of an interactive multimedia presentation is to convey the message to the audience in an entertaining, yet informative way using a variety of different content and interaction techniques.

Design the Media Content

Just as a film director must create the audio, video, and special-effects for a movie, an interactive media author must create the media content necessary for a presentation. To create the content, or media elements, an author may use one or more software tools such as Adobe PhotoShop, Adobe Premiere, or Macromedia's SoundEdit. Because an author must anticipate and provide for a variety of unique experiences, content creation is a challenging and time-consuming task.

An author has two primary objectives when designing content. First, an author must design content that conveys the intended message to the audience. For example, to explain the use of a new software product to a company's employees, an author may create several screen captures of the product's interface along with descriptive voice narrations explaining their use. Second, an author must design content that fits together coherently, e.g., a narration to accompany a video must have a length close to that of the video.

Organize the Media Content

Once the media elements have been created, an author must organize them in both space and time. In other words, an author must define where and when each media element will be displayed within the presentation. The organization of media elements in space is called *spatial layout* while the organization of media elements in time is called *temporal layout*.

Spatial Layout

Spatial layout refers to the physical location and relative positioning of visual media elements on the computer screen. For each visual media element to be displayed such as a video, image, or text element, the exact screen location where it is to appear must be specified. The spatial layout of media elements should be visually pleasing to the viewer and appropriate for the message being conveyed. To ensure an effective layout, an author should apply proven user interface and graphic design principles (20, 36, 37). For example, semantically related information and functions should be positioned close together in order to make the display more readable and to offer the correct affordances.

Temporal Layout

Temporal layout refers to when, for how long, and under what conditions each media element is displayed within a presentation as well as to the synchronization of multiple media elements within that presentation. For example, as a voice narration describes a country's culture to the viewer, the target country is highlighted on a displayed map. After the narration finishes, the highlight remains visible for a few seconds before being removed. Both the initial highlighting of the country and the delay before its removal are examples of how media elements must be synchronized within a presentation.

Defining temporal layout is a difficult task and has been the focus of much research over the past decade. In research literature, temporal layout is often expressed through the use of a synchronization model. A synchronization model is a formal technique for representing, or modeling, the temporal relationships among the different media elements of a presentation. Synchronization models can generally be categorized as (9):

- *Timeline*. Events are placed at specific time points along a timeline. An event might represent the starting or stopping of a voice narration, or it might represent the display or removal of an image.
- *Hierarchic*. Special synchronization operators, parallel and serial, are applied to the endpoints of different media elements. By using these operators, an author constructs a presentation tree where the interior nodes represent simultaneous or sequential playback of their children and the leaf nodes represent the individual media elements.
- *Reference point*. Synchronization points are placed between media elements. These points can either inhibit or cause the display of one or more media elements (44).
- *Event based*. Programming scripts, known as event handlers, are attached to specific events that may be generated *within a presentation such as the starting or stopping of a voice narration. Whenever an event occurs, the attached script is invoked.*

In addition to modeling temporal relationships, a synchronization model may also attempt to detect temporal inconsistencies (15) or automatically generate runtime playback schedules (10) from those relationships. However, using a synchronization model to author an interactive multimedia presentation is difficult because only a few support viewer participation (7, 23).

Design the Interaction

Interaction defines the participatory role of a viewer within a presentation. Without the ability to interact, viewing a presentation is similar to viewing a television show or listening to a radio broadcast. A goal of interaction design is to enable a unique experience for each presentation viewer. A unique experience is created by enabling the viewer's interests, desires, or skills to influence the display of information. Designing innovative interaction is the key to creating appealing, compelling, and effective multimedia presentations.

Within a multimedia presentation, interactions can generally be categorized as:

- *Control*. Represents the operations a viewer can use to directly manipulate the playback of a presentation. For example, a viewer could use VCR operations such as pause and fast-forward to control presentation playback.
- *Navigation*. Represents the links a viewer can traverse to jump to the information that most interests them. For example, a virtual tour of a home might provide the viewer with links to different sections of that home such as the kitchen, living room, and bedroom.
- *Participation*. Represents the responses a viewer can make throughout a presentation. Based on these responses, a presentation can determine the next set of content to be displayed. For example, an interactive math lesson might request the viewer to enter one more numbers at a strategic location within *a problem. The next set of content to be displayed depends upon whether the viewer entered the correct numbers or not.*

In addition, interactions can also be categorized as *synchronous* or *asynchronous* within a presentation. A synchronous interaction requests viewer input at a specific time point within a presentation, and the presentation halts until a response is given. An asynchronous interaction may occur at any point within a specified temporal interval, where that interval may span from only a few seconds to the entire presentation.

Authoring Tools and Languages

An author uses a variety of tools to accomplish each task within the authoring process. These tools may include word processors for script writing and documentation, graphic design tools for creating and editing visual content, and sound tools for voice recording and editing. Traditionally however, the term *authoring tool* has referred only to those tools facilitating the implementation of a multimedia presentation such as HyperCard, Director, and Authorware. In this article, the term authoring tool is used in the traditional sense; however, it is important to realize that professional authors often rely on many other tools during the authoring process.

An authoring tool simplifies the implementation of a multimedia presentation by providing abstractions for spatial and temporal layout, media playback, and interaction. These abstractions are often made available to an author through either a scripting or visual authoring language integrated as part of the authoring tool. Although there are no hard and fast rules, choosing the right authoring tool and language depends primarily on the temporal and interactive requirements of the presentation being defined, the abstractions directly supported by the authoring tool, and the author's ability to understand and specify the requirements using those abstractions.

Scripting Languages

A multimedia scripting language is a general purpose scripting language equipped with commands suitable for implementing multimedia presentations such as for controlling sound and movie clips, positioning media elements on the screen, and creating transitional effects. Examples of multimedia scripting languages are HyperCard's HyperTalk and Director's Lingo. Similar to building a user interface with a user interface toolkit (38), authoring an interactive presentation with a scripting language requires the use of an event-based programming model. In this model, a presentation is not implemented or executed in a linear fashion as with a traditional C or C++ program; rather, a presentation is implemented by attaching short scripts to specific events generated from the viewer or runtime system. An example of a viewer event is the pressing of a mouse button while an example of a system event is the reaching of a specific playback point within a sound or movie clip. Whenever an event is generated, the runtime system searches for and then executes the script attached to the event, if any. Execution of the script may affect the global state of the presentation. The primary advantages of multimedia scripting languages are:

- *Expressive power*. Because multimedia scripting languages support general programming constructs such as variables, functions, branching, looping, and file I/O, they can be used to implement innovative interactive presentations. In addition, almost any attribute of a presentation or media element can be manipulated through the scripting language.
- *Rapid development*. Scripting languages are interpreted which alleviates an author from the details of memory allocation, provides high-level programming abstractions such as lists, sets, and associative arrays, and supports interactive development. An interpreted language enables faster development than a traditional language such as C or C++ (38).
- *Specialized commands*. Multimedia scripting languages are equipped with multimedia specific commands such as those for playing and controlling audio and video clips. Leveraging these commands also speeds the implementation of a presentation.

Although multimedia scripting languages can be used to implement almost any presentation, they are best utilized for implementing a presentation involving innovative interaction. For more information on multimedia scripting languages and the event-based authoring paradigm, see the sections on HyperCard and Director in this article.

Visual Languages

Visual authoring languages allow an author to use direct manipulation to define the spatial and temporal layout, and interactive behavior of a presentation. Spatial layout is defined by interactively positioning media elements within a presentation window. Temporal layout is defined using a visual metaphor and editor such as a timeline editor (16), synchronization graph editor (10, 11), or flowline editor (24, 42). Temporal relationships are defined by relative positioning, drawing behavioral arrows, or placing specific display and erasure commands, respectively. Interactive behavior is defined by reusing existing interaction objects or by editing detailed property sheets associated with generic interaction objects. Using a visual authoring language provides several benefits:

- *Fewer programming concepts*. An author is not required to know about detailed programming concepts such as scope, functions, or memory allocation. As a result, an author can focus more on the high-level aspects of the authoring task rather than on the low-level aspects of the implementation.

- *Concreteness*. An author can directly manipulate the spatial and temporal layout, and the interactive behavior of a presentation.
- *Immediate visual feedback*. An author can immediately visualize the behavior of a presentation, which aids in the debugging process and increases the general understandability of the presentation design.

In contrast to scripting languages, visual authoring languages are best utilized for smaller presentations involving less innovative interaction; otherwise, visual languages can become cumbersome to use and comprehend. For more information on visual authoring languages see the section on Authorware in this article.

HyperCard: Authoring Hypermedia Using Cards

HyperCard is an authoring tool used to build hypermedia presentations. In contrast to multimedia authoring tools, hypermedia authoring tools focus more on the structure and linking of semantically related information and less on expressive temporal models. The concept of hypermedia dates back to a 1945 paper written by Vannevar Bush entitled “As We May Think” (12). In his paper, Bush discussed the difficulty of collecting, organizing, and later retrieving the vast amounts of scientific literature being generated by other scientists. To help overcome this emerging problem of information overload, Bush proposed a mechanical device called Memex, short for memory extender. As Bush stated, “a Memex is a device in which an individual stores his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility.” Bush emphasized that the most important aspect of this device was the ability to create associative links, or *information trails*, among documents of particular interest or importance. Information trails would provide a storage and retrieval model that more closely matches how the human mind works. Thus, as a scientist browsed information and discovered pertinent documents, or documents that were especially informative on a certain subject, their trail of knowledge discovery could always be re-traced by other scientists. Although the Memex device was never actually built, the idea of creating associative links among semantically related information is paramount to all hypermedia authoring tools.

Although it was built nearly 40 years after Bush first described his Memex system, the HyperCard authoring tool embodies many of the same ideas. HyperCard uses an intuitive card metaphor to facilitate the collection, organization, and retrieval of related information. Authors can establish links (trails) between the different cards enabling HyperCard users to browse and retrieve related information. In this section, we give a brief history of the HyperCard authoring tool, an overview of its authoring environment, and demonstrate its use by creating a digital photo album.

History

HyperCard began as an extension to the original MacPaint software, the first commercial product available for the Macintosh computer. At that time, the creator of MacPaint, Bill Atkinson, wanted to enhance the software to support interactive behavior enabling a user to click on a portion of an image in order to bring up a related textual description. As he continued to explore and refine his idea, Atkinson began to realize the huge potential of this new interactive medium, and convinced Apple to fully support his endeavor. In 1986, Apple computer released the first commercial hypermedia authoring tool named HyperCard. The name HyperCard reflects the purpose of the tool; to organize information into a network of interconnected cards.

Because early Mac computers shipped without a programming language, utilizing them to create new applications such as address books, file system shells, or interactive games, was virtually impossible. To address this issue as well as to enable the authoring of more complex interactive presentations, Apple developed an interpreted scripting language called HyperTalk. Because the language was intended to be used by people unfamiliar with programming, HyperTalk was based on an English-like syntax and command structure. In 1987, Apple began shipping HyperCard bundled with HyperTalk on every Macintosh computer free of charge. Due to the power and availability of the HyperTalk language, HyperCard quickly became the programming tool of choice. However, as other multimedia authoring tools, such as Director and Authorware, became more powerful, and as Apple introduced a new scripting language called AppleScript, the usage of both HyperCard and HyperTalk began to fade. Today, HyperCard is mainly used as a robust prototyping environment for both graphical user interfaces and hypermedia presentations.

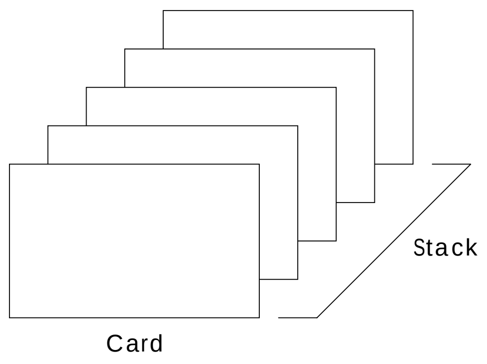


Figure 1. A HyperCard card represents an organizational unit of information. A group of cards sharing a common theme is a HyperCard stack.

Authoring Environment

HyperCard provides a simple and intuitive authoring environment for collecting, organizing, and retrieving information existing in a variety of forms such as text, images, sound, and QuickTime movies. HyperCard documents are known as *stacks*, and are made up of identically sized *cards*. A card represents the fundamental unit of information display, organization, and addressing within a stack. The relationship between a stack and card is depicted in Figure 1. To further illustrate the stack and card metaphor, consider a personal address book implemented using HyperCard. The address book itself would be represented as a single HyperCard stack. The information associated with each entry in the address book, such as a person’s name, address, and telephone number, would be organized and displayed using a single HyperCard card. To support navigation among the different entries, each card would contain several links, such as to the next, previous, first, and last cards.

Card Elements

A HyperCard card supports three media element types; a text field, image, and button. The text field and image types have their obvious meanings; however, a button provides more functionality than just navigation. A button can be used to create visual effects, launch other applications, or play QuickTime movies. To place an instance of one of these media element types onto a card, an author chooses the appropriate element type, sets the related properties, and then visually positions the instantiated element onto the card. Because HyperCard does not support media types for audio or QuickTime movies, an author must write HyperTalk scripts to achieve this functionality.

Card Layers

A HyperCard card consists of two layers, a background layer and a foreground layer. The background layer is used to organize and display information content and related behaviors common to all cards within a stack. By using this layer, an author saves both time and space as the content and behavior in the background only has to be specified and stored once. The foreground layer, also known as the card layer, is used to specify content and behavior that changes from card to card. Before placing a media element onto a card, an author must choose the appropriate layer into which the element is to be placed.

HyperTalk

HyperTalk is the English-like scripting language enabling authors to incorporate dynamic behavior and interactivity into their HyperCard presentations. An author uses HyperTalk to write numerous scripts, which are relatively short pieces of programming code structured in the form of message and function handlers. A message handler defines how the presentation responds to the occurrence of a particular message such as a mouse click or the opening of a card, whereas a function handler defines a supporting routine. Scripts can be attached not only to specific card elements such as a button or text field, but also to the background layer, card layer, current stack, and home stack.

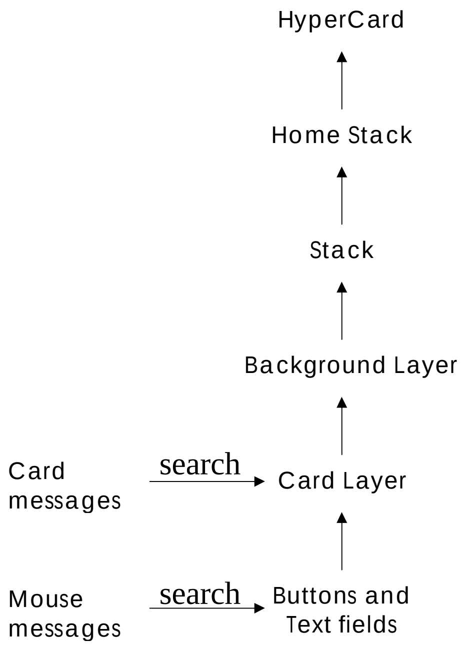


Figure 2. HyperCard’s object hierarchy determines the order in which scripts are searched for a matching message handler. The type of the message generated determines where in the hierarchy HyperCard begins its search. Once a matching handler is found, HyperCard automatically invokes it.

A HyperTalk message is generated in response to a user interaction such as a mouse click, or to a system event such as the opening of a card. For each message generated, HyperCard searches the attached scripts in a pre-defined order looking for a matching message handler, and if a match is found, HyperCard automatically invokes it. Scripts are searched based on the objects to which they are attached, starting with the most specific such as a particular instance of a button, and continuing towards the more generic such as the current stack. The object hierarchy defining the script search order is illustrated in Figure 2. The location within the object hierarchy at which HyperCard begins searching for a matching handler depends on the type of the message generated. For example, the search for an *openCard* message handler begins at the card level and then moves up the hierarchy until a matching handler is found. A script attached to an object lower in the hierarchy is never searched and can therefore never respond to this particular message. Although a complete description of HyperTalk's message dispatching behavior is beyond the scope of this article, the reader may refer to (2, 3) for additional details.

Case Study: Authoring a Digital Photo Album

To demonstrate the use of HyperCard's authoring environment, a case study involving the creation of a digital photo album is presented. The goal of this case study is to simulate a traditional photo album environment in which a user can annotate, organize, store, and later

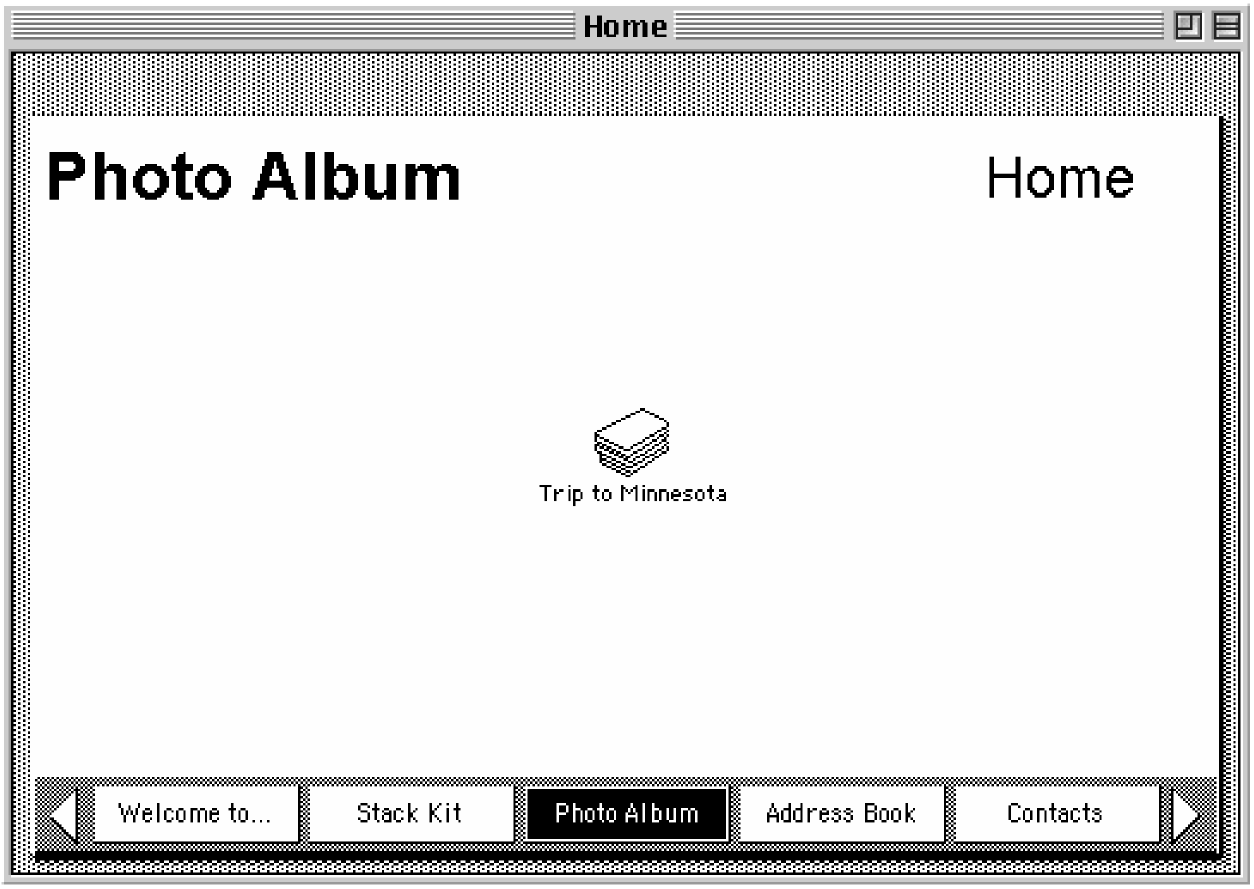


Figure 3. One of the empty home stack cards provided by HyperCard has been adapted for the index of the photo album. The album index contains a link to each stack representing a particular event, such as a "Trip to Minnesota."

retrieve photographs. Specifically, the interactive photo album created here enables a user to:

- Include a title and textual description for each photograph
- Hear the textual description read aloud
- Organize a set of photographs resulting from an event, such as a recent vacation
- Navigate among the photographs resulting from a particular event
- Organize photographs from multiple events into a single album

When authoring a presentation using HyperCard, the first step is to determine the organization of information content in terms of stacks and cards. For this example, the photo album itself is represented using one of the home stack cards that come standard with HyperCard. Unlike other cards, a home stack card may contain links to other stacks, thus providing an additional level of organization for multiple stacks. In this example, the home stack card is entitled "Photo Album" and contains a link to each event in the album. Each event is then represented as a distinct stack containing the individual photographs. This structure is illustrated in Figure 3. Although the album only contains a single event, additional events with their related photographs can easily be added.

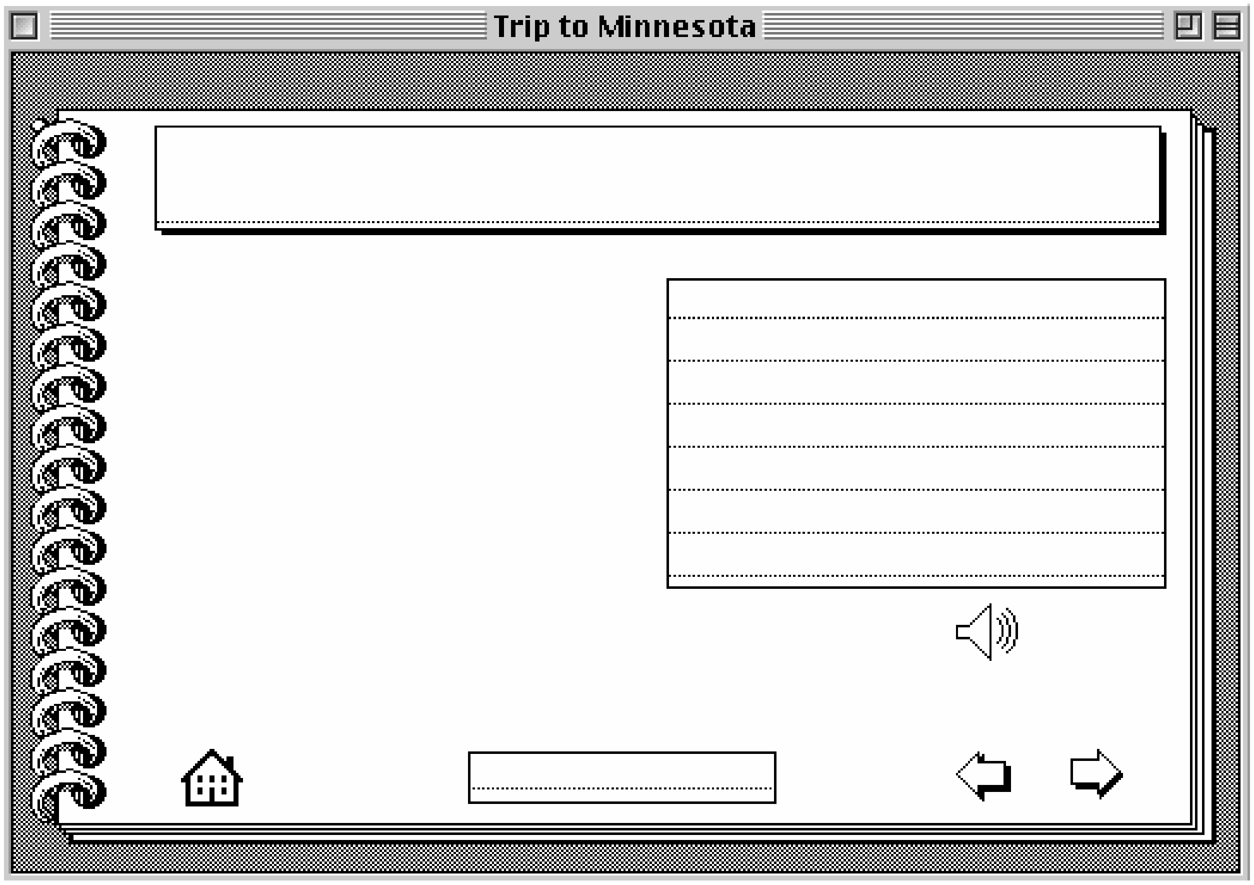


Figure 4. Three text entries representing the title, description, and card count information, and four buttons representing the previous, next, read aloud, and home stack behaviors have been placed into the card background layer.

Each photograph entry in the album is represented using a single HyperCard card containing the following information and behaviors:

- *Background image*. Simulates the appearance of a physical photo album.
- *Photograph*. The actual photograph taken with a digital camera or digitized with an image scanner.
- *Title*. A heading for the photograph.
- *Description*. Additional information associated with the content of the photograph such as its historical significance, related stories, or when and where it was taken.
- *Narration button*. Reads the description of the photograph aloud when selected.
- *Home stack button*. Allows the user to navigate back to the index of the photo album.
- *Card count*. Displays the current card number along with the total number of cards associated with this event.
- *Navigation controls*. Allow the user to navigate to the previous and next entry.

Because each photograph entry shares much of the same information structure and behavior, the common elements are factored out and placed into the card's background layer. Utilizing the background layer guarantees a consistent look and feel among the existing entries and serves as a template for creating new ones. In the background layer the title, card count, and description elements are added as empty text fields. When a new card is created from this background, the appropriate information can be entered into each text field. Similarly, the icons and related behaviors have been added to the background layer as illustrated in Figure 4.

Behavior is defined by attaching a HyperTalk script to a button. For example, in order for the narration button to retrieve and then read aloud the textual description, a HyperTalk script is attached to it. Within the script, a *mouseUp* message handler has been defined and will be

```
on mouseUp
        speak (background field “Description”)
end mouseUp
```

Figure 5. The HyperTalk message handler that reads the photograph's description aloud. The script first retrieves the contents of the text description field and then sends HyperCard a request to read it aloud. The script is attached to the narration button shown in Figure 4.

```
on openCard
        put the number of cards into len
        put "Card " && the number of this card && " of  "
            && len into background field "Counter"

end openCard
```

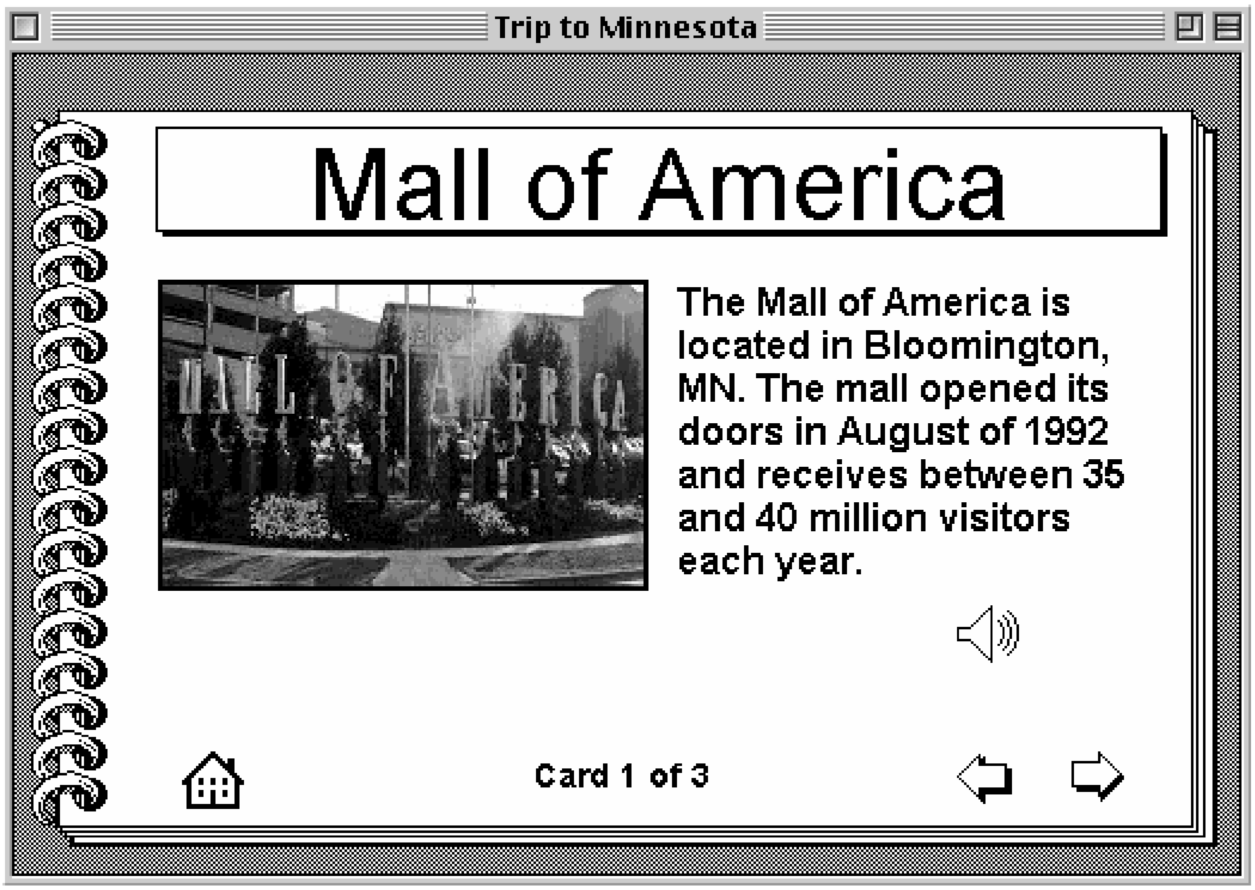
Figure 6. The HyperTalk message handler that displays the card count information at the bottom of each card. The script retrieves the current card number as well as the total number of cards in the current stack and then inserts these values within a meaningful structure. The script is attached to the current stack and is invoked each time a card is opened.

invoked whenever the user clicks on the narration button. When invoked, the script retrieves the contents of the text description field and then sends a request to HyperCard to read the text aloud. The programming code for this message handler is shown in Figure 5.

Displaying the correct card count information is more complex because new cards can be added to or removed from the stack at any time. Thus, the card count information cannot be statically defined; rather, it must be dynamically placed into the text field each time a card is opened. This behavior is accomplished by attaching a script to the current stack and defining a handler for the *openCard* message. The programming code for this message handler is shown in Figure 6.

Each of the home, previous, and next buttons also has a script attached to it which responds to the *mouseUp* message. The programming code for each of these message handlers is a single statement instructing HyperCard to *go home*, *go prev*, and *go next*, respectively.

By placing each of these common elements into the background layer and attaching the appropriate behaviors through scripts, new cards created from this background will automatically share its structure and defined behavior. Adding a new photograph entry to the album only requires choosing this background, and then entering the title, text description, and photograph for the entry. Each of the defined behaviors such as the narration button, card count, and



The Mall of America is located in Bloomington, MN. The mall opened its doors in August of 1992 and receives between 35 and 40 million visitors each year.

Figure 7. The photo album entry for the Mall Of America. Because the background layer shown in Figure 4 was used to create this card, only the photograph and text for the title and description fields need to be included. Each of the behaviors located in the background layer continue to function correctly without modification.

navigation controls function correctly without modification. To illustrate this concept, a new card containing a photograph of the Mall of America has been created and inserted into the photo album. The resulting entry is shown in Figure 7.

Macromedia Director: A Timeline-based Authoring Tool

Macromedia Director is one of the most widely used multimedia authoring tools today. Director has been successfully applied within a variety of application domains such as kiosks, games, and computer-based training. The success and widespread use of Director can be attributed to several factors. First, Director is a mature authoring tool whose movie-based metaphor makes it is easy to learn and use for non-programmers. Second, it offers a mature, powerful, and extensible scripting language called Lingo that can be used by professional authors to implement innovative interactive presentations. Finally, Director is a cross-platform authoring tool supporting Windows and Macintosh computers as well as a variety of publishing mediums including CD-ROM, videotape, and the Internet. To support Internet publishing, Macromedia has developed a set of software products called Shockwave. Shockwave enables a Director presentation to be embedded within a web page and subsequently viewed within a web browser. In this section, we give a brief history of the Director authoring tool, an overview of its authoring environment, and demonstrate its use by re-creating the interactive video game Pong.

History

In 1984, Marc Canter founded the Macromind company with a vision of putting high-end audio, video, and graphics technology into the hands of artists and musicians. The company's first product, SoundVision, was a multimedia authoring tool useful for integrating voice and animation on the early Macintosh computer. Soon after the release of their SoundVision product, the company separated out the animation and sound tools creating two new products; MusicWorks and VideoWorks. MusicWorks was an enhanced version of the company's sound tools while VideoWorks was an enhanced version of the company's animation tools. In 1987, Macromind integrated an interpreted scripting language called Tiny Basic into their VideoWorks product forming a new product called VideoWorks Interactive. As its name implied, this new product was designed for authoring *interactive* multimedia presentations. As development continued on this new product and language, they both received new names. VideoWorks Interactive was renamed *Director* while Tiny Basic was renamed *Lingo*. In 1992 Macromind merged with a small company named Authorware to form the present-day company Macromedia. The Director authoring tool along with its interpreted scripting language Lingo have set the standard for professional authoring tools in use today.

Authoring Environment

Macromedia Director's authoring environment is based on a movie metaphor using components such as the cast, stage, and score. The movie metaphor was intended to make the authoring tool intuitive to use and learn by its target audience; artists, musicians, and video editors. However beneath this metaphoric cover, Director is a timeline-based authoring tool offering a timeline partitioned into discrete temporal units for authoring. A timeline simplifies the specification of time-based behavior such as animations and transitional effects. For example, an author can animate an object by incrementally adjusting one or more of its attributes, such as its location, scale or color, over a finite period of time. Similarly, an author can define a transitional effect such as a wipe, fade, or dissolve by first choosing the appropriate effect and then stretching it across an interval proportional to the desired transition time.

Although it simplifies the creation of time-based behavior, a timeline complicates the creation of an interactive presentation because it does not directly support asynchronous interaction (7). An asynchronous interaction may occur at any point within a specified temporal interval, where that interval may span from only a few seconds to the entire presentation. Because time is progressing during the playback of a presentation, defining a precise interval in which an asynchronous interaction may occur is usually not possible. For example, assume a presentation offers an "exit" button which the viewer can select at any time to end the presentation immediately. Because the length of the presentation is unknown, defining a finite interval in which the interaction may occur is not possible. As a result, either the duration of the interaction must be set to infinity or the progression of time within the presentation must be stopped. Director supports the latter technique through the Lingo command *go to the frame*, which is demonstrated in the case study.

In Director, a presentation is a collection of one or movies created using the Cast, Stage, and Score components of the authoring tool as well as numerous Lingo scripts.

Cast

The Cast represents the set of media elements used within a presentation. Each media element within the Cast is known as a *cast member*. Example cast members include audio clips, graphic images, geometric shapes, video clips, and film loops (animations). Each cast member is named and maintains a set of default properties such as size, display location, and color. In order for a cast member to be visible during the presentation, it must be placed onto the Stage. An instance of a cast member on the Stage is called a *sprite*. A sprite maintains a distinct set of properties initialized to the values of the underlying cast member. Thus, multiple instances of a single cast member can appear on the Stage simultaneously, yet can be individually controlled.

Stage

The Stage represents the display area of a presentation, typically defined as a 640 x 480 window centered on the computer screen. In order for a cast member to be visible during a presentation, an instance of it (sprite) must be placed onto the Stage. Each sprite placed onto the Stage is represented as a sequence of filled frames within one of the Score's sprite channels. The filled frames of the sprite channel determine when, and for how long the sprite is visible on the Stage.

Score

The Score is a visual timeline and editor used to specify the temporal layout of a presentation. The Score's editable timeline is subdivided into both horizontal and vertical units. A horizontal unit is called a *frame* and represents a discrete unit of time. The collection of frames spanning an entire horizontal row is called a *sprite channel*, or just channel for short. Within a channel, the starting point and duration of a sprite appearing on the Stage is represented as a sequence of one or more filled frames. Vertical units represent different sprite channels, enabling multiple sprites to be active on the Stage simultaneously. The collection of frames spanning a vertical column; i.e., the same frame number within each sprite channel, represents the same instant in time. Vertically aligned sprites are drawn on the Stage in ascending order of the occupied channel number.

When a cast member is either added to the Stage or dragged onto the Score directly, a sprite representing the cast member is inserted into the first available channel. An author can then move the sprite horizontally or vertically to manipulate when and in what order it appears on the Stage. An author can also shrink or stretch the sprite across channel frames in order to manipulate the sprite's duration. Although many of its frames may not be used, a channel is typically occupied by a single sprite for the entire presentation.

Lingo

Lingo is the English-like scripting language enabling authors to incorporate dynamic behavior and interactivity into their Director presentations. Using Lingo, an author can also control almost any aspect of the presentation including cast member properties, sprite properties, frame location, and playback rate. Similar to HyperTalk, an author uses Lingo to write numerous scripts, which are relatively short pieces of programming code structured in the form of event and function handlers. An event handler defines how the presentation responds to the occurrence of a particular event such as a mouse click or entering of a frame, whereas a function handler defines a supporting routine. Director supports four types of scripts:

- *Sprite scripts*. These scripts allow behavior to be attached to individual sprites. For example, suppose a "click" sound should be heard whenever a viewer selects a specific graphic element on the Stage. To create this behavior, an author would define the *mouseUp* event handler within a script and then attach it to the graphic element's sprite in the Score.
- *Cast member scripts*. These scripts allow behavior to be attached to cast members, thus providing default behavior for sprite instances. For example, suppose a "click" sound should be heard whenever a viewer selects an instance of a particular cast member on the Stage. Rather than defining and attaching a separate script to each sprite instance, an author could write a single script defining this behavior and then attach it directly to the cast member.
- *Frame scripts*. These scripts allow behavior to be attached to a specific frame within the Score. For example, suppose a presentation reaches a playback point where a viewer response is required before it can continue. Using the appropriate Lingo code to hold the presentation until a response is received, an author would define the *enterFrame* event handler within a script and then attach it to the appropriate frame in the Score.
- *Movie scripts.* These scripts usually define functions globally accessible to any of the sprite, cast member, or frame scripts. However, a movie script can also be used to define behavior associated with the beginning or ending of a presentation. For example, suppose a QuickTime movie needs to be pre-fetched at the beginning of a presentation to ensure smooth playback. Using the appropriate Lingo code to pre-fetch the QuickTime movie, an author would define the *prepareMovie* event handler within a script and then attach it to the current movie.

An event is generated in response to a user interaction such as a mouse click, or to a system event such as the entering of a frame. For each event generated, Director searches the attached scripts in a pre-defined order looking for a matching event handler, and if a match is found, Director automatically invokes it. Scripts are searched based on the elements to which they are attached, starting with the most specific such as a particular sprite, and continuing towards the more generic such as the current movie. The hierarchy defining the script search order is illustrated in Figure 8. The location at which Director begins searching for a matching handler within the hierarchy depends on the type of the event generated. For example, the search for an

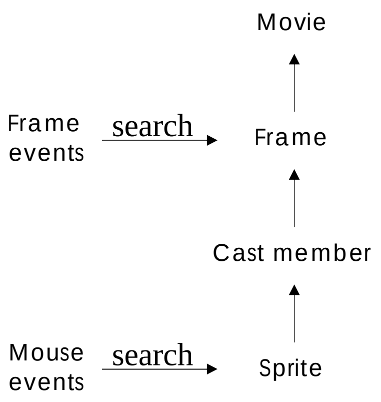


Figure 8. Director's object hierarchy determines the order in which scripts are searched for a matching event handler. The type of the event generated determines where in the hierarchy Director begins its search. Once a matching handler is found, Director automatically invokes it.

enterFrame event handler begins at the frame level and then moves up the hierarchy until a matching handler is found; thus, scripts attached to elements lower in the hierarchy are never searched and can therefore never respond to this particular message. Although a complete description of Lingo's event dispatching behavior is beyond the scope of this article, the reader may refer to (28) for additional details.

Case Study: Recreating the Game of Pong

In this section the Director authoring tool is used to recreate the world's first video game Pong. The single-player version of Pong involves three objects; a paddle, ball, and 2-D court. The court consists of three walls; a bottom, left, and top wall. The fourth (right) wall of the court does not exist. Located on the right side of the court, the player-controlled paddle can be moved vertically but must remain within the confines of the court. The ball, which is controlled by the game itself, always moves in a straight line. Anytime the ball touches a wall or the paddle, it bounces off in a new direction computed from the negative of its incoming slope plus a small noise factor. The player's goal is to keep the ball within the confines of the court by moving the paddle vertically and bouncing the ball away from the missing right wall. If at anytime the player misses the ball

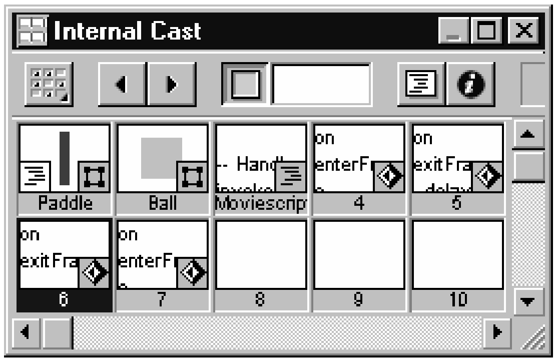


Figure 9. The Cast window showing the different cast members used to create the game Pong. Both the Paddle and Ball shapes are cast members as well as the Lingo scripts that define the interactive behavior.

with the paddle, the ball escapes the court and the game ends. To challenge the player, the speed at which the ball travels around the court is slowly increased.

The first step in authoring this interactive game is to decide upon the appropriate cast members. For this game two filled rectangles will suffice; one representing the paddle, and the other representing the ball (the first two cast members displayed in Figure 9). The 2-D court is represented by the Stage itself, and thus, does not require an explicit cast member. Both the

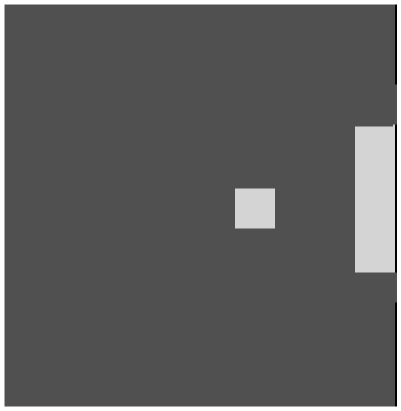


Figure 10. The Paddle and Ball cast members positioned on the Stage.

```
          (a)
on mouseDown
          global startV  -- starting vertical position of the mouse

          set startV to the mouseV
end

          (b)
on mouseWithin
          global paddleX, paddleY     -- (x,y) of the paddle
          global paddleHeight, startV
          global paddleChannel        -- Sprite channel of the paddle

          if the mouseDown then
                    set deltaY to (the mouseV - startV)
                    set newY to (paddleY + deltaY)
          else
                    set paddleY to newY
                    set the locV of sprite paddleChannel to paddleY
                    set startV to the mouseV
          end if
end
```

Figure 11. The cast member script for the Paddle defines two event handlers. The *mouseDown* handler in (a) records the vertical position of the mouse when its button is pressed. The *mouseWithin* handler in (b) updates the paddle location based on how far the mouse has moved vertically. The Lingo code enforcing that the paddle stays within the court has been omitted.

paddle and ball are created using Director's shape tool, which automatically inserts each shape into the Cast. From the Cast window, the paddle and ball are dragged onto the Stage and positioned as shown in Figure 10. By dragging the cast members onto the Stage, Director automatically inserts sprites into the Score for each one. The next step is to write the Lingo scripts defining the behavior for both the paddle and ball. Although this game does require a few other scripts to be written, only these two are described here due to space limitations and because they represent the main function of the game.

The player must be allowed to click on the paddle and then drag it up and down. However, the paddle must always remain within the confines of the Stage. Because only a single instance of the Paddle cast member is ever placed onto the Stage, defining this behavior within either a sprite or cast member script would suffice. This example uses the latter.

The paddle's cast member script defines two event handlers; *mouseDown* and *mouseWithin*. The *mouseDown* handler records the vertical position of the mouse when its button is pressed. The *mouseWithin* handler updates the paddle location based on how far the mouse has moved vertically and also enforces that the paddle remains within the court (Stage). The Lingo code implementing these handlers is shown in Figure 11.

```
        (a)
on exitFrame
        updateBallLocation
        go to the frame
end

        (b)
on updateBallLocation
        global ballX, ballY, ballWidth, ballHeight
        global ballChannel, paddleChannel
        global wallsHit
        global slope, increment

        if (ballX <= 0) then
                set slope to ( (-1 * slope) + noise())
                set increment to (-1 * increment)
                set wallsHit to (wallsHit + 1)
        else if ( (ballX + ballWidth) >= 640) then
                ....
        ....
        endif

        set ballX to (ballX + increment)
        set ballY to (ballY + (slope * increment))
```

Figure 12. A portion of the frame script which animates the ball and detects if the ball touches the paddle or a wall. The event handler in (**a**) is invoked each time the playback head exits the current frame. Because the handler forces the playback head to continuously re-enter the current frame, the progression of time has been stopped. The function handler in (**b**) displays the partial code determining whether or not the ball is touching the left or right side of the court. If so, the new direction and speed of the ball is computed and then used to update the ball's location.

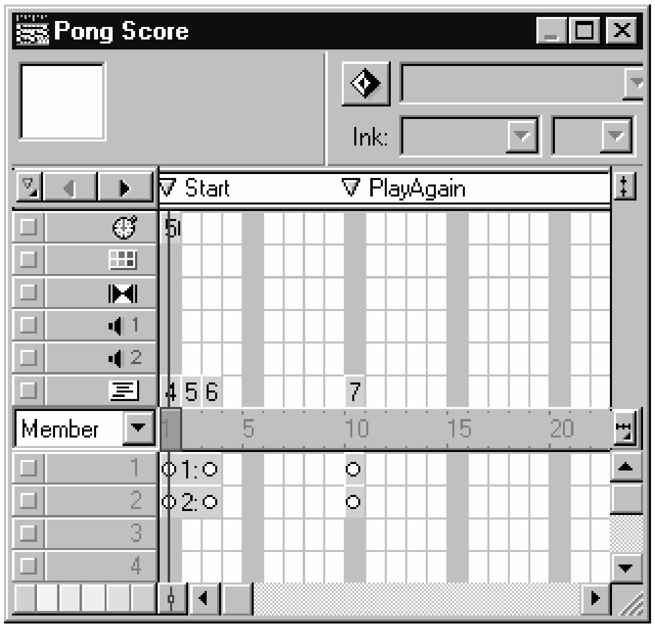


Figure 13. The Score of the Pong game after implementation. The sprites representing the Paddle and Ball cast members are numbered 1 and 2, respectively. The sprite representing the frame script of Figure 12 is numbered 6 in the Score.

The script defining the behavior of the ball implements the main logic of the game. Because the player's interaction with the paddle represents an asynchronous interaction, the script defining the ball's behavior must be written as a frame script. The frame script uses a technique common among Lingo programmers; a continuous loop created with the Lingo command *go to the frame*. By using this loop, the behavior of the ball is integrated with the player's ability to interact with the paddle. The frame script implements the following game logic:

- *Paddle and wall detection*. Upon touching the paddle or the bottom, left, or top wall of the court, the ball bounces away and continues in a new direction. If the player misses the ball with the paddle, the game ends and then asks the player to play again.
- *Speed increment*. The speed of the ball is periodically increased to challenge the player.
- *Adjusted bouncing*. When the ball bounces away from the paddle or a wall, its new direction includes a small noise factor causing it to traverse different paths.

This game logic is shown in Figure 12 while the Score of the finished Pong game is shown in Figure 13.

Authorware: A Flow-based Authoring Tool

Authorware is an authoring tool specifically designed for creating interactive courseware presentations. Examples of these types of presentations include computer-based training, educational learning, situational testing, and drill and practice. Like other multimedia presentations, the goal of an interactive courseware presentation is to convey a message to the viewer using a variety of different media. However, to ensure that the viewer has understood the presented message, an interactive courseware presentation periodically judges the viewer's knowledge related to the presented information. For example, consider an interactive courseware presentation designed to help undergraduate students learn the concept of mathematical integration. Assume the instructor has already prepared several primary lessons, supplemental lessons, and interactive exercises for the students. After each primary lesson, a student is asked to complete several exercises on their own and respond with the calculated answers. If the student answers a high percentage of the exercises correctly, then the next primary lesson will be presented, otherwise, the supplemental material associated with that primary lesson will be presented next. By judging their understanding of the material, the presentation allows a student who needs additional instruction to receive it, while allowing a student who has correctly grasped the concept to continue on. An authoring tool designed to facilitate the creation of interactive courseware presentations must support the storage, manipulation, retrieval, and reporting of viewer responses; which is exactly what Authorware provides.

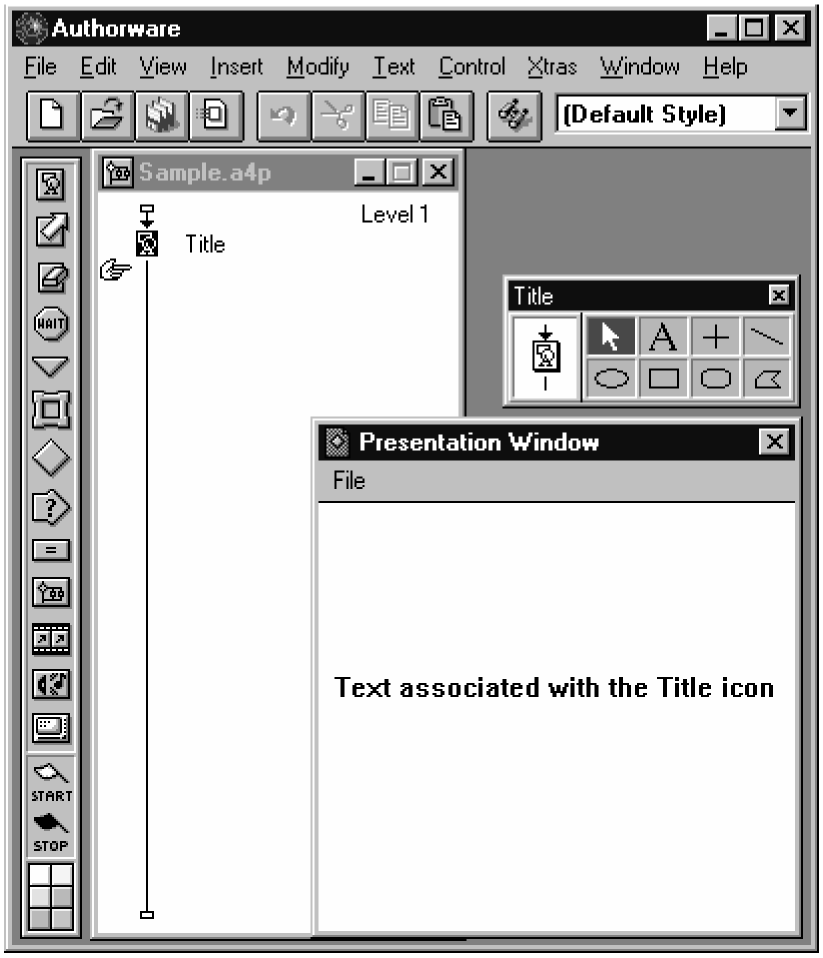
Authorware provides a unique authoring environment in that building interactive presentations requires little or no programming. To build a presentation, an author drags pre-defined behavior icons onto a visual flowline and then fills out a set of related properties. Authorware provides icons for playing media such as audio and video, as well as for creating viewer interactions such as hyperlinks and multi-choice questions. The flowline provides a visual representation of the presentation's behavior, or flow over time. Once implemented, an Authorware presentation can be published on CD-ROM, DVD, or the Internet. In this section, we give a brief history of the Authorware authoring tool, an overview of its flow-based authoring environment, and demonstrate its use by building an interactive atlas.

History

The origins of Authorware are firmly rooted in a system named PLATO (**P**rogrammed **L**ogic for **A**utomatic **T**eaching **O**perations) developed by Don Bitzer at the University of Illinois in the early 1960s (8). Professor Bitzer was interested in applying computer technology to learning environments, and subsequently founded the **C**omputer-based **E**ducation **R**esearch **L**aboratory (CERL) at the University. Along with several other engineers and students, Bitzer designed the PLATO hardware and software system at this laboratory. The hardware consisted of a time-sharing mainframe computer while the software consisted of an authoring language called TUTOR. Although the system originally supported only a small classroom of users, it eventually supported up to one thousand users simultaneously.

Realizing the potential commercial value of this new technology, Control Data Corporation began turning PLATO into a commercial product in 1975. One of the principal designers of this commercial product was Dr. Michael Allen. By 1985, over 100 commercial PLATO systems were operating at sites around the world. However, the mid 1980s also marked the beginning of the microcomputer revolution. Because microcomputers were becoming a more cost-effective platform than mainframes, Control Data Corporation sold or closed many of its mainframe businesses, including those that supported the PLATO systems. Recognizing the need for state-of-the-art multimedia authoring tools in this emerging PC market, Dr. Allen left Control Data Corporation and founded a new company named Authorware in 1985.

In 1987 Authorware released its first authoring product called Course of Action, which was designed to facilitate the creation of instructional courseware. The company continued to enhance and support this product until 1992, at which time Authorware merged with the Macromind company. The merger resulted in the formation of a new company called Macromedia. Because Macromind was already an established producer of multimedia tools, the merger helped solidify Authorware's place in the interactive multimedia market.



Text associated with the Title icon

Figure 14. The Authorware environment consists of three main components; the Icon palette, Design window, and Presentation window. The Icon palette, located on the far left, provides the pre-defined behavior icons an author uses to construct a presentation. The Design window (entitled Sample.a4p) contains the flowline onto which these behavior icons are placed. The Presentation window displays the output of icons as they are executed along the flowline, provides tools for creating text and geometric shapes, and allows an object's visual properties to be modified.

Authoring Environment

The Authorware tool has three distinguishing characteristics setting it apart from other authoring tools. First, Authorware is based on the concept of a *flowline*. A flowline represents the behavioral "flow" of a presentation over time. However, time is represented at a more abstract level and is not partitioned into discrete time units as in a timeline-based authoring tool. To build a presentation, an author drags pre-defined behavior icons from a tool palette onto the flowline. The flowline represents not only the visual design of the presentation, but also the implementation of the presentation. Second, Authorware provides direct support for interaction with little or no programming. In fact, programming is considered an advanced feature of the tool and most introductory texts on Authorware do not even cover this topic. Third, Authorware provides several built-in mechanisms such as response properties and system variables for controlling and judging viewer responses. Response properties enable an author to control the number of answer attempts and enforce time limits regarding a viewer's response. System variables enable an author to retrieve the response time, number of correct or incorrect responses, total number of judged responses, and more. Together, these mechanisms expedite the process of creating interactive courseware presentations.

The Authorware environment consists of three primary components; the Icon palette, Design window, and Presentation window. These components are shown in Figure 14. The Icon palette provides the pre-defined behavior icons an author uses to construct a presentation. The Design window contains the flowline onto which these behavior icons are placed. The Presentation window displays the output of icons as they are executed along the flowline, provides tools for creating text and geometric shapes, and allows an object's visual properties such as location, size, and color to be modified. The next three sections describe each of these components in greater detail.

Icon Palette

The Icon palette provides the pre-defined behavior icons used to construct a presentation. Specifically, Authorware provides the following behavior icons:

- *Display*. Displays text, graphics, and images on the screen. These media elements can be imported from an existing file or created using Authorware's text and shape tools.
- *Motion*. Animates a visual media element on the screen. An author can define the speed, path, and direction of the animated element.

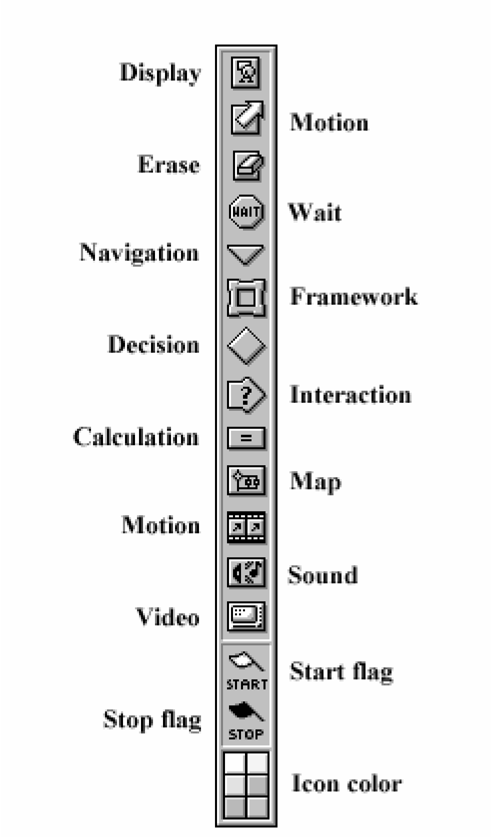


Figure 15. The behavioral icons available from the Icon palette.

- *Erase*. Removes one or more displayed media elements from the screen.
- *Wait*. Pauses execution of a presentation until either a specified amount of time elapses or the viewer clicks a “continue” button.
- *Navigate*. Jumps to a specific page created as part of a framework.
- *Framework*. Defines a set of pages along with navigation controls.
- *Decision*. Directs execution along one of several paths based on parameters such as viewer responses or variable values, or by following a sequential or random order.
- *Interaction*. Displays information or controls that a viewer may interact with either synchronously or asynchronously. Supported interactions include drag and drop, menu selection, hot spots, buttons, hyperlinks, and more. Whenever a viewer responds to an interaction, execution of the presentation continues along the path defined for that particular response.
- *Calculation*. Executes an attached programming script written in the Authorware language. A script can access and manipulate almost any aspect of the presentation.
- *Map*. Creates a hierarchical grouping of icons facilitating a modular design.
- *Movie*. Imports a video file existing in a standard format such as MPEG, AVI, or QuickTime.
- *Sound*. Imports an audio file existing in a standard format such as PCM, AIFF, or WAV.

Each icon within the Icon palette is labeled in Figure 15. Once an author places an icon onto the flowline, numerous properties of the icon can be configured specializing its behavior. In some cases, determining the proper configuration of an icon and understanding how the icon will behave in the context of the overall presentation can be difficult.

Design Window

The Design window is where an author visually constructs the temporal layout and interaction of a presentation. An author selects icons from the Icon palette and drags them onto the flowline contained in the Design window. The flowline begins at the top of the Design window and ends at the bottom, as shown in Figure 14. Although the flowline is not a timeline, it does impose a temporal ordering on the placed icons. An icon placed higher on the flowline is executed before an icon placed lower on the flowline. The flowline is not considered a timeline because it does not represent discrete points in time. Authorware executes each icon until it is “finished” and then moves sequentially to the next one. The execution time of an icon depends on its type and configured properties. For example, the execution time of a display icon is the amount of time required to draw the associated visual elements in the Presentation window. On the other hand, the execution time of an interaction icon depends on whether the interaction has been configured as perpetual or time-bounded, and on the viewer’s response time. Because the execution time of an icon is either unknown or can only be estimated, achieving a precise level of synchronization can be difficult, e.g., playing a sound effect at a precise point within an animation. Timeline-based authoring tools such as Director are much better at achieving this precise level of synchronization through the use of temporal layers.

Presentation Window

The Presentation window displays the output of icons as they are executed along the flowline, provides tools for creating text and geometric shapes, and allows an object’s visual properties to be modified. To associate visual content with an icon, such as a display or interaction icon, an author first places the icon onto the flowline and then selects it by double-clicking. After selecting the icon, any content imported, created, or otherwise placed into the Presentation window is associated with that icon, and displayed whenever that icon is executed. For example, in Figure 14 a display icon has been placed onto the flowline and selected. Next, a text message was created in the Presentation window causing it to be associated with the display icon. Each time the display icon is executed, the associated text message will be displayed. Any visual content displayed in the Presentation window remains visible until it is explicitly removed

Case Study: Designing an Interactive Atlas

To demonstrate the use of Authorware's authoring environment, a case study involving the creation of an interactive atlas is presented. The interactive atlas displays a map of the world's continents and then asks the viewer three different questions regarding them. For each question, the viewer must respond using a different interaction technique. For the first question, the viewer is presented with the name of a continent and must respond by dragging the name to its appropriate geographic location. For the second question, the viewer is presented with a circled continent and must respond by entering its name. Finally, for the third question the viewer is presented with a multi-choice question and must respond by selecting the button associated with their preferred choice. The viewer receives three attempts to correctly answer each of the first two questions, and only receives one attempt to correctly answer the third. After the last question, the total number of correct answers is displayed to the viewer.

When authoring a presentation in Authorware, an author should first outline the general flow of the presentation. For the interactive atlas, a reasonable outline is:

- Display the world map
- Present Question 1
- Present Question 2
- Present Question 3
- Tally the number of correct answers

This process of creating and further refining multiple levels of detail is known as *top-down* design. Unlike many other authoring tools, Authorware directly supports top-down design through its map icon. Although the first and last steps of the outline can each be accomplished by using a display icon, the middle three steps require that several substeps be completed, and thus, a map icon is used to represent each one of them. The result of placing these icons onto the flowline is shown in Figure 16. For the initial display icon, an image containing a map of the world's continents is imported and associated with it. For the map icons, each one must be further refined.

Question 1 - Drag the Name of a Continent to its Geographic Location

The first question presents the viewer with the name of a continent and then requests that the viewer drags that name to its matching geographic location. By selecting the *Question 1* map icon of Figure 16, Authorware brings up another Design window containing an empty flowline. In this new window, a display icon for the question text is placed onto the flowline. To create the drag and drop behavior, an interaction icon is also placed onto the flowline. Associated with the interaction icon is the name of a continent, Africa, which must be appropriately positioned within the Presentation window. To the right of the interaction icon, three map icons are placed

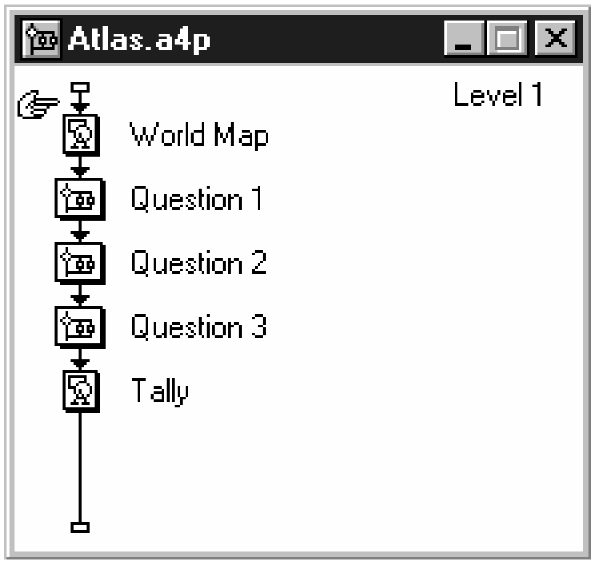


Figure 16. Two display and three map icons have been placed onto the flowline representing the outline of the interactive atlas. Each icon has also been given an appropriate title.

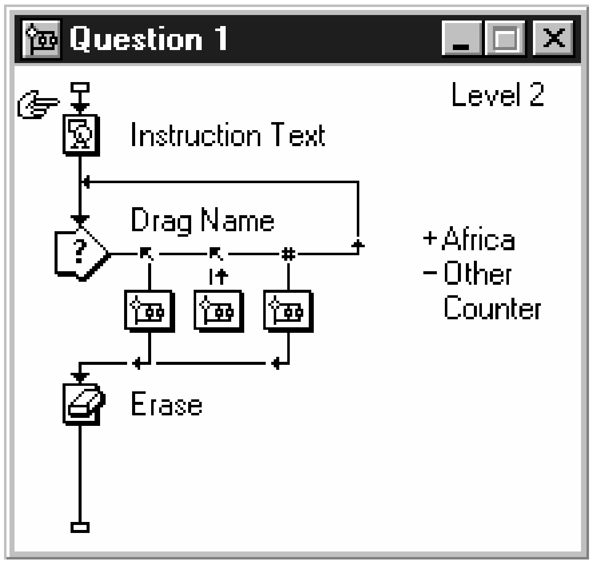


Figure 17. The refined flowline for Question 1. The display icon displays the instruction text to the viewer while the interaction icon displays the name of the continent the viewer must drag to its geographic location. Each map icon to the right of the interaction icon represents a possible viewer response path. The paths represent the correct response, incorrect response, and exceeded number of attempts, respectively.

representing the different response paths. The paths represent the correct response, incorrect response, and exceeded number of attempts, respectively.

When an icon is placed to the right of an interaction icon, Authorware prompts for the response type. The first two response types are set to *target area* while the third response type is set to *tries limit*. Target area represents a specific screen location where the viewer may drag and drop an object, while tries limit represents the number of attempts the viewer is allowed for this interaction. The branching selection for the first and third responses are set to *exit interaction*, while the branching selection for the second response is set to *continue*. Figure 17 displays the results of these steps.

If the viewer drags the name of the continent to the correct location on the map, the first response is matched and the contents of its associated map icon are executed. Afterward, the presentation exits the interaction. If the viewer drags the name of the continent to any other location, the second response is matched and the contents of its associated map icon are executed. Afterward, the presentation continues to the right, allowing the counter (third response) to increment by one. If the counter reaches the maximum value of three, then the contents of its associated map icon are executed and then the presentation exits the interaction. Thus, the interaction continues until the viewer either responds correctly or makes three incorrect attempts.

To complete the implementation of the first question, each of the map icons associated with a response path must also be further refined. After each response, the viewer should receive a message indicating if their answer was correct or incorrect, and whether or not they are allowed another attempt. Because refining each response behavior is similar, only the refinement of the correct (first) response path will be discussed here.

By selecting the map icon associated with the correct response path, Authorware brings up a new Design window. To remove the question text, an erase icon is placed onto the flowline and associated with the text. Next, an appropriate message indicating that the answer was correct is created, and then associated with a display icon. Finally, a wait icon is placed onto the flowline and configured to give the viewer adequate time to read the message. Figure 18 shows the results of these steps. After completing similar steps for the other responses, the first question is now complete and is shown in Figure 19.

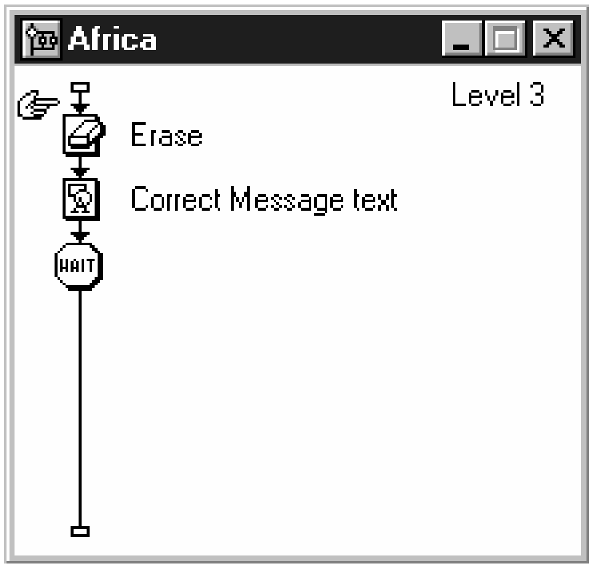
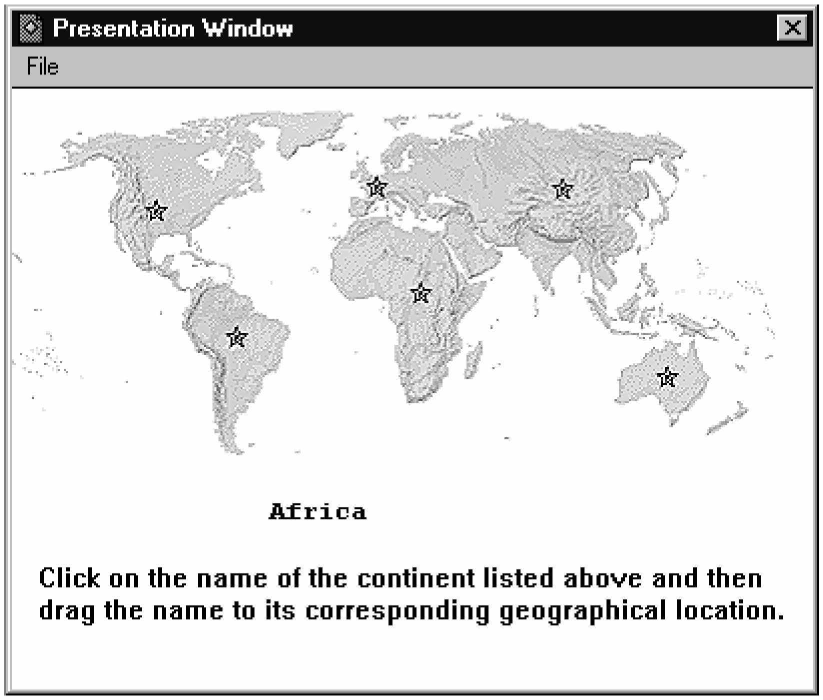


Figure 18. The contents of the map icon associated with the correct response path of Question 1. When a correct response is given, the current question text is erased and an appropriate answer message is displayed. After a short pause, the presentation continues to the next question.



Africa

Click on the name of the continent listed above and then drag the name to its corresponding geographical location.

Figure 19. The display of Question 1 to the viewer. The viewer is asked to drag the name of the continent to its proper geographic location. If the viewer's response is correct, the presentation displays an appropriate message and then moves on to the next question. If the viewer's response is incorrect, the presentation displays an appropriate message and then gives the viewer another attempt. The viewer is given at most three attempts to respond correctly.

Question 2 - Identify the Circled Continent

The second question asks the viewer to identify the circled continent, which in this case is Australia. The steps necessary to design this question are similar to those of Question 1, and as result, will be given in lesser detail. By selecting the *Question 2* map icon of Figure 16, Authorware brings up another Design window containing an empty flowline. To build the question, a display icon is first placed onto the flowline and a circle around the Australian continent is associated with it. Next, an interaction icon is placed onto the flowline along with three map icons representing the different response paths as before. The response types for the first two map icons are set to *text entry* while the response type for the third map icon is set to *tries limit*. For a text entry response, the title of the icon represents the matching answer. Thus, the first (correct) map icon's title is set to *Australia* while the second (incorrect) map icon's title is set to a wildcard, signified by the "--" at the beginning of the title. Because Authorware matches the responses from left to right, the catchall response must be placed to the right of the correct response. The completed flowline for Question 2 is shown in Figure 20.

The map icons associated with the response paths of Question 2 are refined in a similar manner as those in Question 1, and therefore, will not be discussed further. The second question is now complete and is shown in Figure 21.

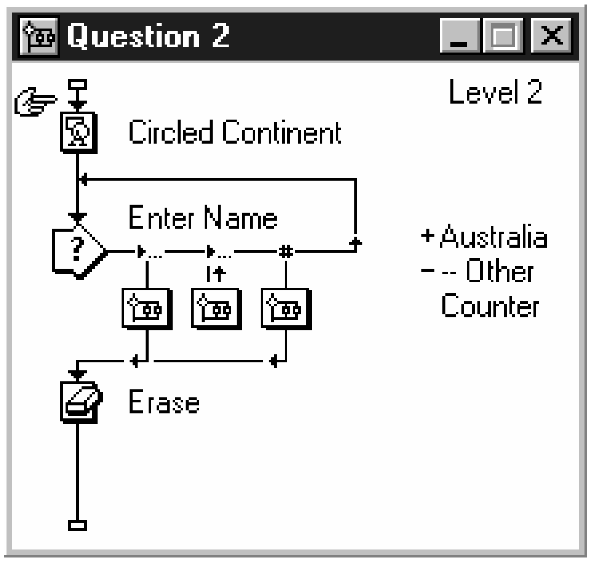


Figure 20. The refined flowline for Question 2. The display icon displays the circle around the Australian continent while the interaction icon displays the question text. Each map icon to the right of the interaction icon represents a possible viewer response path. The paths represent the correct response, incorrect response, and exceeded number of attempts, respectively.

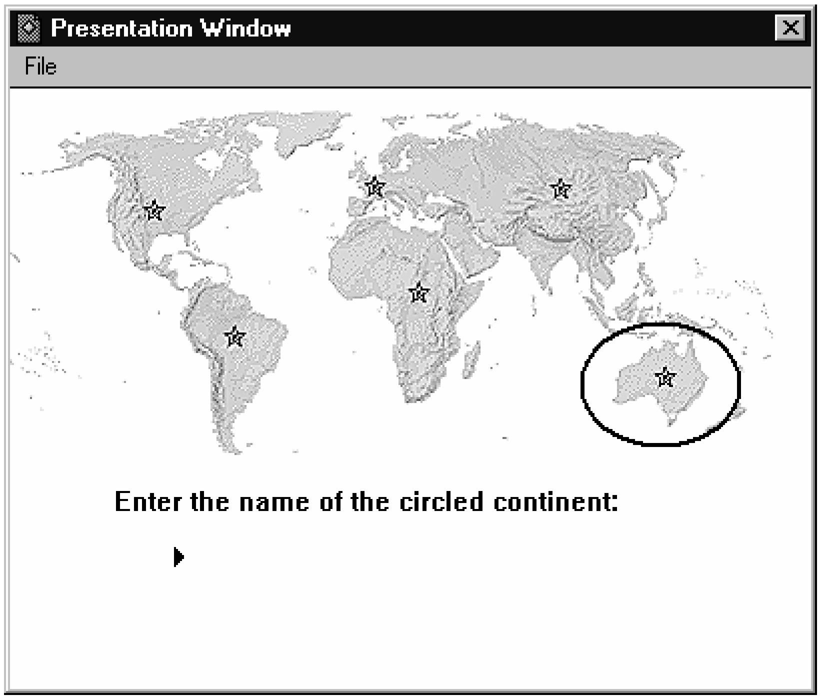


Figure 21. The display of Question 2 to the viewer. The viewer is asked to enter the name of the circled continent. If the viewer's response is correct, the presentation displays an appropriate message and then moves on to the next question. If the viewer's response is incorrect, the presentation displays an appropriate message and then gives the viewer another attempt. The viewer is given at most three attempts to respond correctly.

Question 3 – Select the Continent with the Largest Population Size

The third question is a multiple choice question asking the viewer to select the continent with the largest population size. Unlike the first two questions, this question offers the viewer only a single attempt to respond correctly. By selecting the *Question 3* map icon of Figure 16, Authorware brings up another Design window containing an empty flowline. To build the question, a display icon is first placed onto the flowline and the question text is associated with it. In addition, the text for each answer choice is also associated with this display icon. Next, an interaction icon is placed onto the flowline along with three map icons representing the different response paths as before. The second choice (B) is the correct response while the first and third choices (A and C) are the incorrect responses. The response type of each map icon is set to *button*, and in return, Authorware automatically creates a button in the Presentation window with its label set to the icon's title. Each button must be positioned within the Presentation window next to the answer choice that it represents. The completed flowline for Question 3 is shown in Figure 22.

The map icons associated with the response paths of Question 3 are refined in a similar manner as those in Question 1, and therefore, will not be discussed further. The third question is now complete and is shown in Figure 23.

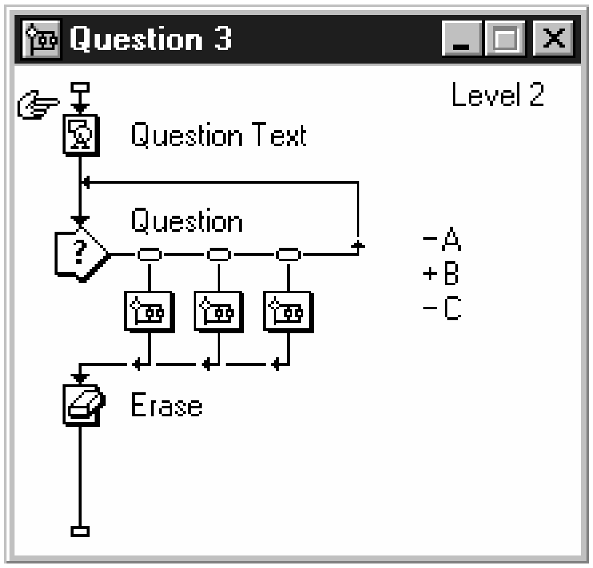
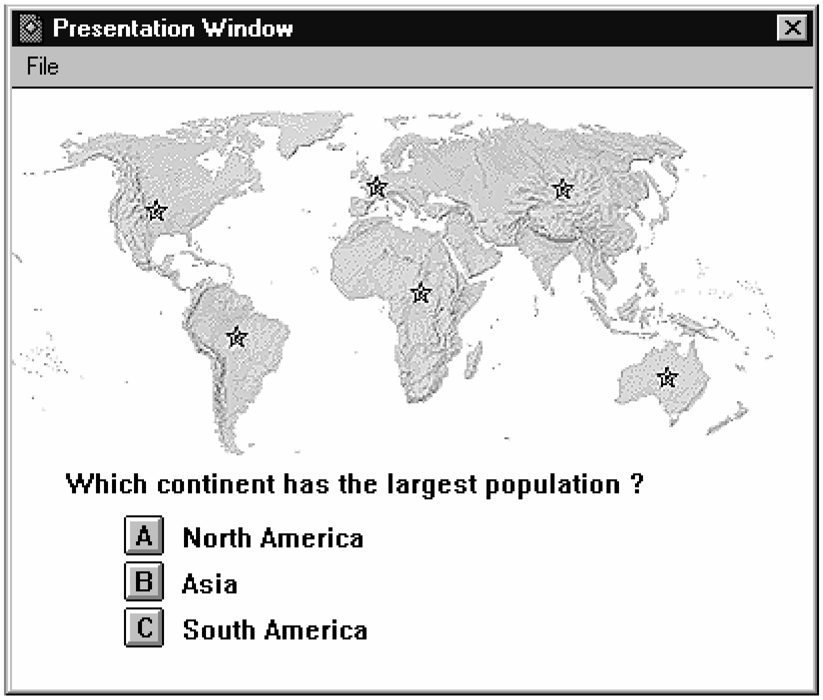


Figure 22. The refined flowline for Question 3. The display icon displays the question text as well as the three possible answer choices. The interaction icon has no visible elements associated with it. Each map icon to the right of the interaction icon represents a possible viewer response path. The paths represent the selection of buttons A, B, and C, respectively.



Which continent has the largest population ?

A North America
B Asia
C South America

Figure 23. The display of Question 3 to the viewer. The viewer is asked to select the name of the continent having the largest population. Upon receiving the viewer's response, the presentation displays an appropriate message and then moves on to the tally of correct responses.

Tally the Number of Correct Responses

The final task of the interactive atlas is to tally the number of correct responses and display this result to the viewer. Authorware provides two system variables, TotalCorrect and JudgedInteractions, which automatically provide the desired information. Along with the *Tally* display icon of Figure 16, the following text is associated with it:

You correctly answered {TotalCorrect} out of {JudgedInteractions} questions

The string names inside the braces represent the names of system variables. When Authorware displays the text, it will replace the system variables with their current values.

The Future of Authoring Interactive Media

Past research in the area of authoring interactive media has focused on building expressive temporal models and logics, providing semantically rich linking structures, and supporting innovative interaction. Future research in the area of authoring interactive media will focus on creating standardized encoding formats for multimedia presentations, building expressive authoring languages for the Internet, and integrating multimodal interactions into presentations.

Standardized Encoding of Multimedia Presentations

An authoring tool enables an author to specify the spatial and temporal layout, and interaction of a multimedia presentation. Once these specifications are made, an authoring tool must encode them into a format that can be distributed and later interpreted for playback or editing. Because each authoring tool uses its own proprietary encoding format, a presentation created with one tool cannot be interpreted by another. For example, a presentation authored with HyperCard cannot be edited or played using Director. To address this problem, the Multimedia/HyperMedia Information Coding Experts Group (MHEG; ISO/IEC 13522) is developing a standardized encoding format for describing the spatial and temporal layout, and interaction of a multimedia presentation (17). The encoding format provides a declarative syntax and associated semantics for describing a multimedia presentation. If MHEG is adopted by new and existing multimedia authoring tools, an author could use their favorite tool to build a presentation, and could then export it in the MHEG format. Another MHEG-compliant tool could then interpret this format for playback or editing. For example, a presentation authored using Director and converted into MHEG, could be played on any computing device equipped with an MHEG-compliant player, speakers, and display device. However, designing an encoding format that is expressive enough to capture innovative interactive presentations is a challenging and ongoing task.

Authoring Languages for the Internet

HTML is currently the primary authoring language for the Internet. Using HTML, an author can easily combine text and graphics into one or more documents, and can establish links between semantically related content. However, an author cannot use HTML to express the sophisticated temporal layout or innovative interactions required by most multimedia presentations. As a result, the World Wide Web Consortium (W3C) has proposed the Synchronized Multimedia Integration Language (SMIL) (48) enabling the creation of interactive multimedia presentations on the Internet. SMIL supports streaming audio and video, run-time adaptation, and a rich timing and linking semantics not found in HTML. Because SMIL is defined by an XML Document Type Definition, authoring a multimedia presentation using SMIL is similar to authoring a hypertext presentation using HTML.

In SMIL, spatial layout is based on the concept of a *region*. A region represents a rectangular area of the screen in which media elements are displayed, and controls the position, size and scaling of those elements. A SMIL presentation can specify any number of overlapping regions, however, a media element must be assigned to a specific region in order to be visible to a viewer.

Temporal layout is defined using the *parallel* and *serial* synchronization grouping tags. These grouping tags place a temporal structure on their child elements (children). A parallel tag starts all of its children at the same time while a serial tag plays its children in a sequential order. A child element may be either a media element such as an audio or video clip, or another synchronization tag. By nesting synchronization tags, an author can achieve complex temporal layouts.

Similar to HTML, SMIL supports interaction through hyperlinks. However, SMIL's hyperlinks have two additional features. First, SMIL allows the destination of a link to be a temporal subpart of the same or different SMIL presentation. A temporal subpart of a presentation is specified by including the synchronization tag identifier associated with that subpart as part of the link destination. This feature allows the viewer to navigate to a specific point within a presentation and have the presentation behave as if it had already played to that point. Second, SMIL allows an author to specify a context for link traversal. Link context defines how the source presentation containing the link behaves when the viewer follows the link. SMIL supports three contexts for link traversal:

- *Replace*. The source presentation is paused and then replaced with the destination presentation. If the viewer returns to the source presentation at a later time, it may resume from the state in which it was paused.
- *New*. The destination presentation starts in a new context and the source presentation is unaffected.
- *Pause*. The source presentation is paused at its current state while the destination presentation starts in a new context. When the destination ends, the source presentation resumes from the state in which it was paused.

Finally, SMIL supports adaptive playback of a presentation based on system parameters and user preferences specified through the *switch* element. The switch element allows an author to specify a set of alternative media elements from which only one element is chosen. For example, a switch element can be used to choose among different qualities of an audio segment based on the network bandwidth available at runtime.

Multimodal Input and Output for Multimedia Presentations

Today's computer looks little different from the workstations developed twenty years ago. Since the addition of the bitmapped screen and mouse to the standard keyboard, little noticeable change has been made in the exterior interfaces of computers. Now, just a few years after computers have commonly become powerful enough to display high-quality audio and video, several significant new input and output modalities are on the horizon.

The two input technologies that are most likely to change multimedia interfaces are speech and user sensing. Speech technology is already available for users who prefer to control a computer by speaking rather than typing. While accuracy is still a challenge, current commercial systems by IBM and Dragon Systems work well enough to satisfy many users. As improvements continue, we are likely to see more speech-controlled applications. User sensing uses various sensors, including cameras, smart cards, and even eye trackers, to detect the presence and actions 0of users. This technology can allow a presentation to reset when the viewer leaves, can adapt a presentation to the set of viewers, or can add humor to a presentation if it notices a drop in attention from the user. As new input technologies become pervasive, authors will have new types of interaction available for creating multimedia presentations.

Two exciting output technologies are virtual reality displays and force-feedback devices. Virtual reality displays combine high quality graphics with accurate detection of user position to create a more realistic sense of immersion in a space. Common displays include: headsets, which may block out the world or display computer graphics over it; cave displays that project contents on the walls of a room; and various special-purpose displays such as car and airplane simulators. Force feedback devices, also commonly used in virtual reality applications, allow users to feel a force through fingers, arms, or a body suit. With force feedback output, it is possible to add substantial reality to an otherwise computer-feeling application. For example, a force-feedback steering wheel adds the real sense of steering to a car simulator. Combining this with force feedback pedals and seats can give surprising realism. It remains to be seen how authors will use these new media for interactive presentations.

Further Reading

Several published books and research papers are available on many aspects of authoring interactive multimedia presentations. For a general introduction to multimedia systems and applications see (36, 46). For insight into the early influential work on hypertext, hypermedia, and multimedia systems see (8, 12, 18, 21, 34, 35, 49). For additional information regarding content selection and design see (6, 19, 20, 25, 29, 30, 33, 40, 41, 47). For additional information regarding temporal models see (1, 7, 9, 10, 13, 15, 23, 26, 39, 43, 44, 45, 48). And for additional information regarding authoring tools and languages see (27, 31).

BIBLIOGRAPHY

1. J.F. Allen, Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26 (11): 832-843, 1983.
2. Apple Computer, *HyperCard: Reference Manual*, 1993.
3. Apple Computer, *HyperCard: Script Language Guide*, 1993.
4. Apple Computer, *Inside Macintosh: QuickTime*. Addison-Wesley Publishing Company, 1997.
5. R. Baecker, A. Rosenthal, N. Friedlander, E. Smith, and A. Cohen, A multimedia system for authoring motion pictures. *Proceedings ACM Multimedia*, 1996.
6. P. Baggett, The role of temporal overlap of visual and auditory material in forming dual media associations. *Journal of Educational Psychology*, 76 (1984), 408-417.
7. B. Bailey, R. Cooley, J. Konstan, and M. DeJong, Nsync - a toolkit for building interactive multimedia presentations. *Proceedings ACM Multimedia*, 1998.
8. D. Bitzer, P. Braunfeld, and W. Lichtenberger, PLATO: an automatic teaching device. *IRE Transactions on Education*, E-4: 157-161, December, 1961.
9. G. Blakowski, and R. Steinmetz, A media synchronization survey: reference model, specification, and case studies. *IEEE J. Selected Areas Comm.*, January, 1996.
10. C. Buchanan and P. Zellweger, Scheduling multimedia documents using temporal constraints. *Proceedings NOSSDAV*, November, 1992.
11. C. Buchanan and P. Zellweger, Specifying temporal behavior in hypermedia documents. *Proceedings ACM ECHT Conference*, 1992.
12. V. Bush, As we may think. *The Atlantic Monthly*, July, 1945.
13. K. Candan, B. Prabhakaran, and V. Subrahmanian, CHIMP: a framework for supporting multimedia document authoring and presentation. *Proceedings ACM Multimedia*, 1996.
14. P.R. Cohen, M. Johnston, D. McGee, S. Oviatt, J. Pittman, I. Smith, L. Chen, and J. Clow, QuickSet: multimodal interaction for distributed applications. *Proceedings ACM Multimedia*, 1997.
15. J.P. Courtiat and R.C. De Oliveira, Proving temporal consistency in a new multimedia synchronization model. *Proceedings ACM Multimedia*, 1996.

16. G.D. Drapeau, Synchronization in the MAEstro multimedia authoring environment. *Proceedings ACM Multimedia*, 1993.
17. M. Echiffre, C. Marchisio, P. Marchisio, P. Panicciari, and S. Rossi, MHEG-5 - aims, concepts, and implementation issues. *IEEE Multimedia*, 5 (1): 84-91, 1998.
18. D. Engelbart, A conceptual framework for the augmentation of man's intellect. *Vistas In Information Handling*, Vol. 1, Spartan Books, Washington D.C., 1963.
19. P. Faraday and A. Sutcliffe, An empirical study of attending and comprehending multimedia presentations. *Proceedings ACM Multimedia*, 1996.
20. P. Faraday and A. Sutcliffe, Designing effective multimedia presentations. *Conference Proceedings on Human Factors in Computing Systems*, 1997.
21. F. Halasz, Reflections on NoteCards: seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31 (7): 836-852, 1988.
22. M. Helander, *Handbook of Human Computer Interaction*. Second Edition, Elsevier/North-Holland: Amsterdam, 1997.
23. J. Herlocker and J. Konstan, Commands as media: design and implementation of a command stream. *Proceedings ACM Multimedia*, 1995.
24. M. Holtz, *The Multimedia Workshop: Authorware Professional 2*.0. Wadsworth Publishing Company, 1995.
25. R.B. Kozma, Learning with media. *Review of Educational Research*, 61 (2): 179-211, 1991.
26. T. Little and A. Ghafoor, Interval-based conceptual models for time-dependent multimedia data. *IEEE Trans. Knowl. Data Eng.*, 5 (4): 551-563, 1993.
27. C. MacKnight and S. Balagopalan, An evaluation tool for measuring authoring system performance. *Communications of the ACM*, 32 (10): 1231-1236, 1989.
28. Macromedia, *Learning Lingo*. Macromedia Press, March, 1997.
29. M.T. Maybury (editor), *Intelligent Multimedia Interfaces*. AAAI Press, 1993.
30. R.E. Mayer, Aids to text comprehension. *Educational Psychologist*, 19 (1984), 30-42.
31. M.D. Merrill, Where is the authoring in authoring systems? *Journal of Computer-Based Instruction*, 12 (4): 90-96, 1985.
32. Myst (computer software distribution), Broderbund Software.
33. L.J. Najjar, Multimedia information and learning. *Journal of Educational Multimedia and Hypermedia,* 5 (1996), 129-150.
34. T.H. Nelson, *Literary Machines*. Swarthmore, PA: Self-published, 1981.
35. T.H. Nelson, On the Xanadu project. *Byte*: 298-299, September, 1990.
36. J. Nielsen, *Multimedia and Hypertext: The Internet and Beyond*. AP Professional, Feb., 1995.
37. D. Norman, *The Design of Everyday Things*. Doubleday, 1988.
38. J.K. Ousterhout, *Tcl and the Tk Toolkit*. Addison-Wesley Publishing Company, 1994.
39. B. Prabhakaran and S.V. Subrahmanian, Synchronization models for multimedia presentation with user input. *Multimedia Systems,* 2 (2): 53-62, 1994.
40. R.A. Reiser and R.M. Gagne, Characteristics of media selection models. *Review of Educational Research*, 52, 499-512.

41. L.P. Rieber, Animation in computer based instruction. *Educational Technology Research & Development*, 38 (1990), 77-86.
42. L.P. Rieber, *Getting Interactive with Authorware: Building Simulations and Games*. http://www.nowhereroad.com/authorware/index.html.
43. K. Rothermel and T. Helbig, Clock hierarchies: an abstraction for grouping and controlling media streams. *IEEE J. Selected Areas Comm.*, January, 1996.
44. J. Schnepf, J. Konstan, and D. Du, Doing FLIPS: flexible interactive presentation synchronization. *IEEE J. Selected Areas Comm.*, January, 1996.
45. R. Steinmetz, Synchronization properties in multimedia systems. *IEEE J. Selected Areas Comm.*, April, 1990.
46. R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications, and Applications*. Prentice Hall, July, 1995.
47. A.G. Sutcliffe, Task-related information analysis. *International Journal of Human Computer Studies*, 47 (2): 223-257.
48. Synchronized Multimedia, http://www.w3.org/AudioVideo.
49. N. Yankelovich, B. Haan, N. Meyrowitz, and S. Drucker, Intermedia: the concept and the construction of a seamless information environment. *IEEE Computer*, 21 (1): 81-96, 1988.