Technical Issues in Hypermedia Authoring Systems* Extended Abstract

Brian M. Dennis and Michael A. Harrison †
Computer Science Division
University of California, Berkeley

Abstract

We present key issues that hypermedia authoring systems must address to be successful. Rapid growth in consumer interest and computer capabilities have increased the expectations from hypermedia software. Meanwhile, hypermedia authors are scrambling to make the best use of new media, new authoring metaphors, new interaction techniques, and new delivery concepts. Only extensible systems designed to accommodate hypermedia scripting issues will survive.

Towards this end, we discuss a number of technical concerns important to hypermedia authoring. We compare and contrast a number of currently available authoring systems. This analysis will help in the design of future systems.

Introduction

There has been a tremendous explosion in interest in multimedia [7]. While by some measures, not a new field, this area of research has recently been energized by breakthroughs in media capture, computer storage, and networking connectivity. Meanwhile, commercial authoring tools have been developing more slowly.

The focus of this paper is on the needs of title *authors*. There are a host of technical and economic issues that an authoring system influences, over and above the final hypermedia product [6, 3, 11]. Authoring systems are generally composed of an authoring environment and a scripting language. We are interested in how authoring systems address the concerns of those constructing titles. In particular, we are interested in critical elements of the scripting language, the media editors, and the total authoring environment.

The rest of this paper proceeds as follows. First, we present some hypermedia scripting issues. Then we begin to survey current commercial offerings. Next we compare and contrast the surveyed systems to illuminate the scripting issues. Lastly, we summarize the paper and discuss our future research directions.

Essential Elements of Authoring

The task of constructing a hypermedia title currently is one largely of design: retouching graphics, editing video, laying out text, devising interactions, etc. However, just as crucial to an effort's success is the programming process needed to actually build the title. This process Since the task of developing a hypermedia title is fundamentally programming, underneath any authoring system there must be a programming language. We term this language a *hypermedia scripting language*

The scripting language is the glue that combines an author's content into a title. The type of applications that are convenient to author is ultimately limited by the scripting language features that are easiest to use. Clearly, the authoring system can provide less flexible, less complex interfaces but any such system that does not have a well designed scripting language will eventually be outgrown.

To assist the programming task, a hypermedia authoring system should help manage the programming process. In short, it should be a software development environment which we shall term a *hypermedia authoring environment.* The complexity of hypermedia titles varies from simple personal projects to multimillion dollar studio efforts. Even simple tools though, should provide support for debugging, prototyping, preview-

*This research has been sponsored in part by the Advanced Research Projects Agency (ARPA) under Grant MDA972-92-J-1028, by the NSF under grant CDA-8722788, and an IBM Fellowship. The content of this paper does not necessarily reflect the position or the policy of the United States Government or IBM.

† {xjam,harrison}@cs.berkeley.edu

ing, etc. More sophisticated systems could add features related to profiling, cataloging content, version control as well.

Authoring environments and scripting languages are highly co-dependent but we believe that they are not inseparable. Indeed, it is useful to make a distinction between the environment and the responsibilities of the scripting language. The CommonLisp [13] and Scheme [4] languages illustrate the ends of the spectrum for this tradeoff.

Media Types

There is a fundamental set of media types that define the content that appears in a hypermedia title. These media types will have to be manipulated and presented by the authoring system. Ideally they would be fundamental datatypes in the scripting language of the system.

Digitized video currently represents the most glamorous content element in multimedia titles. While the engineering constraints of delivering digital video are considerable, video is no more important than any media type from our point of view. Incorporating video special effects is an unexplored area.

Digitized sound presents hurdles similar to digitized video. However, current systems have CD quality sound for a reasonable price. Research is proceeding in THX quality sound for desktops which would raise three dimensional spacing considerations. Multimedia authors will soon have to consider *where* a sound is to be placed just like movie directors.

Despite the attractions of sound and video, text will remain an important element in hypermedia titles. Presentation of text is already at a high level. Integrated tools for structuring, indexing, summarization and cross referencing large bodies of work are not yet available. While the images and sound of an electronic encyclopedia are enticing, dealing with the large body of text that carries much of the information is just as important.

Structured graphics are also a key element in hypermedia scripting. These elements represent a compact, quickly renderable means of incorporating graphic information. They are also a major way in which to interface to external tools such as stand-alone drawing and graphing packages.

Images are cheaply captured and manipulated by current systems. Users expect to be able to incorporate such bitmap graphics as they can be a high density means of information as well as aesthetically pleasing.

Animation is an important element as well. The capability to specify motion over time of graphical objects can be essential for good presentations. As an alternative to video capture, movies can be made through three dimensional modeling and rendering.

Interaction is the component of hypermedia authoring that separates a hypermedia title from a "mere" movie. That the user can control the flow of events instead of passively receiving the data is what computers add to the content. Note here that we are talking about the interaction that an author incorporates into the delivered title, not the author's interaction with the authoring system. Authors need new tools to prototype, implement, reuse, and evaluate interaction mechanism. For many systems this implies tools for dealing with graphical user interfaces.

To summarize, a hypermedia authoring system must be able to support text, sound, video, images, structured graphics, animation, and some form of interaction. Text, sound, graphics and images are all easily and cheaply captured by non-programming individuals and for the immediate future will make up the bulk content of hypermedia titles. Any hypermedia authoring system should strive to handle these elements well.

Hypermedia authors already have content editing tools that they understand and put to good use. A hypermedia authoring system must be open to interfacing to such tools in some fashion. For example, high quality stand-alone digital video editors are available for video professionals. Output from these systems tend to conform to industry standards. At a minimum the authoring system should be able to accept as many industry standard formats of the required content types as possible.

Programming Paradigms

While control flow and procedural abstraction are fairly uninteresting, scripting languages have interesting data abstraction requirements. The focus on ease of use and code reuse almost forces all such languages to have message passing and object oriented features. Systems can vary on how user code is incorporated into the object hierarchy. Applications can be built based on a toolkit or set of libraries. Alternatively, the author constructs an application by augmenting an application framework. The data abstraction facilities of the scripting language clearly influence how such toolkits and frameworks are constructed.

Combining elements that have a time dimension such as audio and video along with non-linear navigation makes temporal coordination increasingly important. How then do we allow authors to express such coordination in an effective manner? The design and implementation of programming idioms and semantics to coordinate and sequence content is little researched. Along those lines, scripting language support for such features as multithreading and synchronization constructs might allow authors to easily build such abstractions.

The field of multimedia is rapidly changing and hypermedia authoring systems must be adaptable. The scripting language must be malleable and open to new programming idioms and paradigms. Incorporating such concepts as object oriented programming, macro systems, dynamic linking and loading, etc. can help authoring systems survive longer.

Many operating systems incorporate interapplication communication mechanisms. Examples are Microsoft's Object Linking and Embedding and traditional UNIX interprocess communication. Authoring systems should let authors take advantage of such features. Preferably, such capabilities would be accessible from the scripting language although interfacing to user written C extensions seems to be current norm. We believe that future scripting languages will have to incorporate language level constructs that abstract interprocess communication.

Authoring Environment Issues

The authoring of a given title is often based around an *authoring metaphor* which helps organize the temporal flow, interaction, and graphical design of the title. There are a number of authoring metaphors currently in use today including score based (Macromedia Director), stack based (Hypercard), flowchart based (Icon Author), etc. These are all essentially frameworks into which a user adds code to build an application An authoring system might want to support more than one such framework. If the system is extensible as we have outlined then it should may be possible for experts to construct new special purpose frameworks as appropriate, e.g. building new libraries in a system like GainMomentum.

Preview and prototyping facilities are necessary in the authoring environment. The current development cycle of hypermedia titles involves an iterative process in which the title is first drafted as a rough cut and then iteratively refined to its final form. A system for authoring must allow the preview of incomplete titles to support this notion. Incremental compilation and linking techniques can be used to shorten development turnaround time.

While interesting hypermedia work can be done by a single person, titles above a minimal level of complexity require a group effort. As part of its development environment a hypermedia authoring system should explicitly support collaborative work. This both entails support for composition of individual efforts and group efforts on individual components of the title.

As hypermedia titles become less like collections of video clips and more like software they will need to have design errors (bugs) repaired and features upgraded. As is the case with software, we expect maintenance to become one of the largest factors in the success of hypermedia titles. Hypermedia authoring systems must provide for both robust titles as well as long term maintenance.

One critical choice that hypermedia authors will have to make is how to actually deliver their titles. Currently, CD-ROM is the preferred format but the future leaves open the possibilities of read/write formats and even higher bandwidth delivery mechanisms. we envision high storage writable formats increasing in importance as they become cheaper, denser and detachable. As well, network delivery will garner a larger share of interest as the Internet, interactive television, remote kiosks, corporate database applications, and computer based training become even more prominent. Other than CD-ROM, authoring systems tend not to address specifically how the title will be delivered.

A constant request from title developers is the ability to develop titles on one platform (combination of hardware and operating system) and have the title be presentable on others. This mainly comes from a desire to reach the widest audience possible. Cross platform support may also allow a developer to take advantage of a powerful and effective development platform already available.

Many electronic devices, coupled with a suitable delivery mechanism, can provide interfaces to hypermedia titles. A hypermedia title can be presented by a wide variety of players including hand held units, settop boxes connected to TVs, up to workstations as well as special purpose kiosks. We have yet to see an authoring system that facilitates the construction of hypermedia titles that are presentable over a range of environments.

Hypermedia Scripting Languages

In this section we present a short survey of authoring systems. Further work will include more hypermedia authoring applications in detail not allowed by the space constraints of this paper.

The HyperCard Family

There are a variety of hypermedia authoring systems that are oriented around a card stack metaphor. This concept originated with Apple's Hypercard and continued with Allegiant SuperCard (formerly from Silicon Beach Systems), and Metacard.

This group of systems is defined by their common authoring metaphor, the card stack. At any given point in the hypermedia title, the author can present one card which contains multimedia content. Multiple cards can exist in a given stack which is essentially the entire title. These cards can be connected with links that let the end user traverse the stack non-linearly.

The authoring environment of these systems tends to be direct manipulation based. The author is presented with a visual representation of the stack and arranges links and graphical user interface elements, which contain multimedia content, graphically. Application specific semantics are specified by the system's scripting language Hypertalk [1]. Hypertalk code is incorporated almost exclusively as handlers for events. These events can be generated by the user or Hypertalk code and are analogous to SmallTalk [5] messages.

Hypertalk syntax attempts to be English like in nature with English verbs representing actions and language objects such as buttons, graphics, and integers standing in for nouns. Hypertalk semantics are much like that of SmallTalk with communicating objects sending messages to each other to execute a program. The original Hypercard was a fairly closed object oriented system only allowing users to create objects, alter some of their data attributes, and send messages to objects. The scripting languages for all of these systems descends from Hypertalk.

GainMomentum

GainMomentum [8] is a hypermedia authoring system which runs on a variety of platforms; its scripting language is called GEL for the Gain Extension Language. While many aspects of GEL are similar to HyperTalk, there are differences in that the underlying metaphor is much richer. Instead of stacks, cards, and fields, etc, GEL employs applications, pages, and other objects more complex than those found in HyperTalk. Because GainMomentum provides access to the major commercial data base systems, there are aspects of GEL that are similar to Fourth Generation Language and the system is so described in [2].

As is typical of successors of Hypercard, GEL keeps the same Hypertalk scripting base but extends the language to accommodate new datatypes and more complex scripting demands. GEL is a complete multimedia database application programming environment. Even the syntax is chameleon-like in that a verbose mode is quite similar to HyperTalk while a concise mode is more procedural. A rich collection of functions make GEL very powerful but the language lacks the polish of HyperTalk.

Director

Macromedia Director is a popular hypermedia authoring system with a scripting language entitled LINGO. Like GEL, LINGO is similar to HyperCard in its object orientation, but the authoring metaphor is quite different. Director adopts the language of movie production, with *casts*, *scores*, and a VCR-like control panel. The system provides for external functions, XCMDs, like HyperTalk.

Because of Director's movie oriented authoring metaphor, the system is particularly good at handling timing and sequencing. A score is a collection of timelines that organize the temporal behavior of the cast. New objects such as *sprites* and *puppets* help coordinate animations and sound.

ScriptX

ScriptX [12, 14] is an effort by Kaleida Labs to develop a platform independent multimedia scripting language and runtime player. While Kaleida itself, along with third party developers, are creating authoring environments based on the ScriptX scripting language there is no complete authoring system currently available.

At the core of ScriptX is an object oriented dynamic language influenced by Dylan [10] which is itself influenced by SmallTalk, CommonLisp, Scheme, and C++. The syntax is much like that of traditional Pascal programs. Meanwhile, ScriptX semantics are once again similar to those of SmallTalk involving a message passing paradigm.

While ScriptX could be used as a general purpose programming language, Kaleida has designed the language specification and standard libraries to be oriented towards multimedia scripting. Constructs and idioms in the language are oriented towards building multimedia applications. For example, the language itself has a fundamental concept of *time* that is incorporated into all objects. This allows developers to uniformly coordinate actions in an application.

ScriptX's major technical achievement is its cross platform capabilities. Some authoring systems such as GainMomentum and Director deliver applications that can run on multiple platforms. The ScriptX specification however *guarantees* that a title developed in the language will be executable on any platform that the Kaleida Media Player runs on.

Visual Basic

Visual Basic [9] is a popular general purpose development tool that is being turned to hypermedia purposes. Many other languages can and have filled this gap including C, C++, and Lisp. However, Visual Basic's ease of use and rapid application development provide some interesting lessons.

The Visual Basic product is a language interpreter with a direct manipulation based development environment. Developers construct visual *forms* built out of *controls*. Code to implement application specific semantics is written in the Visual Basic language itself. The application is developed and prototyped in the "authoring" environment but can be built standalone. The Visual Basic environment provides debugging tools, database interface support, and access to interapplication communication.

Building applications in Visual Basic has a remarkably similar feel to that of Hypercard style applications. Most of the time is spent in directly manipulating the controls and laying out the application interface. The user is called upon occasionally to implement application semantics as event handlers. These event handlers are written in the "scripting language" Visual Basic.

The Visual Basic language, while called Basic, most resembles Pascal in syntax and semantics. There are some vestiges of classic Basic syntax, but for the most part the language can be considered a subset of Pascal. Users only have available a record datatype for constructing their own datatypes. The weak support for data abstraction is overcome by a high level of extensibility. User can write custom controls in other programming languages such as C++ and cleanly make them available to the authoring environment. This technique has been so successful that there is a large market for these custom controls including many which make available multimedia support. This exemplifies how a relatively simple general purpose tool can successfully be bent to multimedia uses.

Discussion

Most of the systems that we have discussed in this paper and that we are looking at are somewhat similar. The development environment is tightly coupled with the scripting language almost to the point of inseparability. The authoring environment is direct manipulation oriented. There is a message passing system within the scripting language although not necessarily an object oriented abstraction facility. All of the systems we have surveyed can support incorporation of the essential media types.

The systems are radically different in a number of areas. The support for code development varies widely. GainMomentum and VisualBasic attempt to be complete software development environments. Hypercard provides minimal code development support. ScriptX currently has no development environment but has many features incorporated into the scripting language to support such environments. We noticed two distinct stances on how content is edited in the systems. Systems such as GainMomentum provide builtin media editors and little support for external tools. Meanwhile, tools like Hypercard and Visual Basic provided no media editing and only served as integrating tools. Content media, other than interaction, was expected to be edited and provided by external programs. Platform independence varied among the environments as well. ScriptX attempts to be completely platform independent while Hypercard is only found on one platform. This does involve some performance tradeoffs as the ScriptX player is fairly large and not as efficient as similar compiled native code.

All of the issues that we raised as essential elements were touched upon by at least one system. No system actually incorporated all of the elements. We found the support for application development while available in many systems simply not scalable to large projects. Along similar lines, there was little to no support for collaborative work and delivery specific presentation. If we discount ScriptX which has just recently been released, multithreading and multiple authoring metaphors are ignored.

Conclusion

We have presented several technical issues related to multimedia authoring systems. We have focused on the features of such systems crucial to the hypermedia author as opposed to the features of the delivered title. For our purposes, authoring systems are divisible into a scripting language and an authoring environment. The scripting language should incorporate standard media types, object oriented abstraction, message passing, support interprocess communication, be multithreaded and extensible. The authoring environment should allow for multiple authoring metaphors, aid in collaborative work, let authors make delivery tradeoffs, and strive to be cross platform. Many of the issues raised in this paper have been touched upon by at least one commercially available system, but no such system has addressed them all.

In the future, we will be exploring the incorporation of techniques from programming language and software development research into hypermedia systems. To satisfy all of the issues we have covered we will be exploring how to make general purpose dynamic languages much more amenable to hypermedia requirements. As our current work on Ensemble shows, there are very close parallels between general purpose programming environments and hypermedia authoring systems.

References

[1] Apple Computer Inc. *HyperCard Script Language Guide: The HyperTalk Language.* Addison Wesley, Reading, MA, 1988.

[2] Hilary Beech and Susan Renner. GainMomentum – an analysis of the software architecture and concepts. *SQL Forum*, September 1993.

[3] Jeff Burger. *The Desktop Multimedia Bible.* Addison Wesley, 1993.

[4] William Clinger and Jonathan Rees. Revised^4 report on the algorithmic language scheme, November 1991.

[5] Adele Goldberg and David Robson. *Smalltalk-80, the Language and its Implementation.* Addison-Wesley, Reading, MA, 1983.

[6] Michael A. Harrison. Defining hypermedia: The essential elements. Technical Report UCB/CSD-92-671, University of California, Berkeley, Berkeley, CA 94720, July 1992.

[7] GISTICS Inc. Interactive index. *Morph's Outpost on the Digital Frontier*, Ongoing 1994.

[8] Daniel D. McCracken. *Introduction to GainMomentum – Tutorial and Cookbook.* Addison-Wesley, Reading, MA, 1992.

[9] Microsoft Corporation, Redmond, WA. *Microsoft Visual Basic: Programmer's Guide*, version 3.0 edition, 1993.

[10] Ike Nassi, editor. *Dylan Interim Reference Manual.* Apple Advanced Technology Group, 1994.

[11] William T. Park. The future of authoring. *NewMedia*, 4(5):52–54, May 1994.

[12] Assaf Reznik. Character simulation with ScriptX. *Dr. Dobb's Journal*, pages 76–82, November 1994.

[13] Guy L. Steele Jr. *CommonLisp, The Language; Second Edition.* Digital Press, 1990.

[14] Ray Valdes. What's up at Kaleida. *Dr. Dobb's Developer Update*, pages 1–7, September 1994.