

Personalizing Hypermedia: The Role of Adaptive Multimedia Scripts

by

Laura Hallie Robin

Bachelor of Arts in Computer Science
Boston University
Boston, Massachusetts
May, 1984.

Submitted to the Media Arts and Sciences Section,
School of Architecture and Planning
in Partial Fulfillment for the Requirements of the Degree
Masters of Science in Visual Studies

Massachusetts Institute of Technology
September, 1990.

©Massachusetts Institute of Technology, 1990. All rights reserved.

Author:

Media Arts and Sciences Section.
August 3, 1990.

Certified by:

Muriel Cooper, Professor of Visual Studies, Thesis Supervisor.

Accepted by:

Stephen Benton, Chairman, Department Committee for Graduate Students.

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

OCT 04 1990

LIBRARIES
Rotch

Personalizing Hypermedia: The Role of Adaptive Multimedia Scripts

by
Laura Hallie Robin

Submitted to the Media Arts and Sciences Section,
School of Architecture and Planning on August 3rd, 1990,
in Partial Fulfillment for the Requirements of the Degree
Masters of Science in Visual Studies.

Abstract

This thesis describes a system for the authoring and temporal adaptation of multimedia scripts. The system has two components; an authoring component and a hypermedia component.

With the authoring component, a user can design multimedia presentations using text, still images, sound, video and animation objects. Using graphic interfaces, the user can format these in objects in space and time, modify their media-specific attributes, and delineate the objects semantically as pertaining to specific levels of detail. This delineation creates hierarchical structures which can then be selectively parsed by the system to create variations on the original presentation. The user can save all formatting and chunking decisions in a script, creating a persistent record of the presentation.

These scripts can then be associated with the nodes of a hypermedia system, so that when a hypermedia-user invokes a node, some version of the scripted presentation is played back. The hypermedia component is responsible for determining version content. Based on user-imposed constraints, (hypermedia-users can dynamically input how much time they have to spend with the system, and interest levels associated with represented concepts), the system can adjust the amount of detail shown in a given presentation, thus contracting or expanding the presentation over time.

This thesis further discusses temporal allocation strategies, graphical constraint specification, and the notions of graphic and semantic adaptation.

Thesis Supervisor: Muriel Cooper, Professor of Visual Studies.
The work reported herein was supported by NYNEX and Hewlett Packard.

Table Of Contents

Abstract

1. Introduction	4
1.1 Thesis Roadmap	5
1.2 Related Work	6
2. System Overview	12
2.1 A Sample Session with the System	12
2.2 Authoring Component	14
2.3 Hypermedia Component	21
2.4 System-Level Control	24
3. Adaptation Strategies	25
3.1 Script Versioning	25
3.2 Dynamic Script Generation	26
3.3 Detail Hierarchies	27
3.4 Hierarchies and Concept Formation	28
3.5 Detail Hierarchy Specification	30
4. Constraint Management	32
4.1 Temporal Distribution	33
4.2 Interest Distribution	34
4.3 Design Constraint Specification	35
4.4 Script Modification Strategies	36
5. Scripting Tools	39
5.1 Browsing Environment	39
5.2 Scripting Environment	41
5.3 Playback Environment	43
5.4 Save/Load Criteria	44
5.5 Playback Control Mechanism	45
6. Conclusions and Future Directions	47
Appendix A -- Design in the VLW	50
Appendix B -- Implementation	53
Appendix C -- Sample Script	56
Bibliography	60
Acknowledgements	62

Chapter 1. Introduction

Imagine a hypermedia system which addresses the research being conducted in each laboratory of MIT's Media Lab. Given some large amount of time with the system, one could traverse all the levels of nested information, including published papers, videotaped classes, dynamic demonstrations, and multimedia presentations. Chances are slim, however, that a user would ever spend the time or have the interest required for this complete-traversal scenario. It is more likely that a user would only traverse a small subset of this information, based on their specific goals for using the system.

For instance, one user might be interested in getting an introductory view of the research areas, while another might have a very pointed research goal related to a specific project. Another user might want to learn as much as possible about each lab, yet have only a small amount of time to spend with the system. Further, each of these users might change their initial goals while using the system. The introductory-level user might become more interested in a particular area of research and decide to delve into a more detailed account of that work. The user with pointed research directives might find that

related research is being done by another group within the lab and change his research goals accordingly.

How can a hypermedia system efficiently adapt its information display to the differing and dynamically changing viewing goals of its users? How can it optimize its information display so that users are not wasting time, caught in webs of seemingly unrelated information? In a broader view, how can hypermedia systems learn to adapt their information displays in a style consistent with the form and content of the originally designed material? Can hypermedia systems learn and correctly apply various adaptive styles?

1.1 Thesis Roadmap

This thesis examines these questions in the context of a *script-based* hypermedia system - a system where the invocation of a node starts a playback of a multimedia script. *Multimedia scripts* are files of playback-specific data, recording the sizes, positions, durations and attributes of text, image, sound, narration, video and animation objects. Playing back a script results in a linear presentation of these objects, analogous to the traditional multi-format slide show, yet backed with the compute power to perform real-time graphics operations and simulations. The hypermedia system described in this thesis has the capability to dynamically modify these scripts, and thus the display of information, in response to user time and interest constraints. (For the purpose of this thesis, user interaction is limited to time and interest specification, though in the future, the degree of interaction shall expand.)

Design criteria for affecting this adaptive hypermedia system will be addressed in the following chapter -- *A System Overview*. These criteria will be discussed in the context of the functional modules of the system. Chapter 3 -- *Adaptive Modification Strategies*, reviews strategies available for adaptive modification of multimedia information, including a discussion of the strategy chosen for this system. Chapter 4 -- *Constraint Management*, addresses the system's constraint management capabilities, as well as resource distribution strategies. The system's authoring tools for multimedia script creation are discussed in Chapter 5 -- *Scripting Tools*, and future directions are covered in Chapter 6 -- *Conclusions and Future Directions*. A more detailed view of the motivations for this work can be found in Appendix A - *Design in the VLW*, and a discussion of the implementation details can be found in Appendix B -- *Implementation*.

1.2 Related Work

This system lies at the intersection of interactive hypermedia systems and scripted multimedia documents (see Zellweger [27], [28]). Like most hypermedia systems, it allows its users to browse and navigate through the information space with little or no path specification. Yet like scripted multimedia documents, the information structure asserts control over what the user sees because the material at the nodes of the hypermedia system is scripted. This material can be adaptively modified by the system based on the changing goals and interests of different users at different times.

Another related area of study is that of personalization in computing environments.

Examples of computer-based personalization are plentiful. Systems that take an active

role in affecting change (based on some set of calculations or rule invocation) are coming into a more popular existence due to both a demand for responsive systems, and the fact that we are rapidly coming to grips with the technology required to model rules for “intelligent environments”.

MIT’s Media Lab has always held close ties to the notion of personalization. The research foundations for MIT’s Media Lab were laid during the early seventies at MIT’s Architecture Machine Group. Work conducted there formed the basis of a vision which would come to hold *personalization* as one of the most important factors in future electronic environments.

The Spatial Data Management System (Bolt [3]), created at the Architecture Machine Group, looked at personalization in the context of human interface. According to Nicholas Negroponte, Director of the Media Lab:

“...we believe future systems will not be characterized by their memory size or processing speed. Instead, the human interface will become the major measure, calibrated in very subjective units, so sensory and personalized that it will be evaluated by feelings and perceptions. Is it easy to use? Does it feel good? Is it pleasurable?”

Spatial Data-Management (1979)

Personalization in this context evolved into research on the use of “natural” modalities for communicating with the computer (such as gesture, eye movements, etc.), and having these forms of communication met with “perceptive” system response.

Network Plus (Bender & Chesnais [2]) is a system used for personalized news retrieval, consonant with individuals' preference profiles. A software agent, acting on behalf of the viewer, is programmed to watch all television channels, listen to all radio transmission, and read all magazines, newspapers and journals. It then can filter and format this information, based on what the viewer has dubbed as personally news-worthy. *Network Plus* suggests that intelligence for personalizing news broadcasts will be local to a software receiver, as opposed to a "news-editor", thus implying new degrees of freedom in the relationship between picture, data and programming.

The Interactive Cinema group in the Media Lab puts personalization to use in electronic editing environments (Mackay & Davenport [15]). Users can perform virtual video editing (recombining and playing back video clips in real time) using digital video manipulation techniques with multimedia authoring tools. Though this work nods to *manual* personalization using editors, it leads the way to the need for "intelligent" editing capabilities which make use of expert system technology to aid in editing processes.

Efforts in the machine-learning domain in the Visible Language Workshop have created systems responsive to design activity, capable of dynamically building and maintaining models of design style (Ishizaki [13]).

While there are seemingly endless models of systems which implement some form of personalization, these models stop short at the hypermedia frontier. Historically, the lack of personalization in hypermedia systems is evident. Bush's "memex" environment [4], first described in 1945 described a device which would act as an "enlarged intimate supplement" to the user's memory -- basically a huge storage device, which speedily and flexibly allowed access to all the user's knowledge and records. Doug Englebart, developer of NLS [8], introduced some structure to the memex idea with the development of structured multimedia documents, and by discussing the notions of linking and collaborative work. Ted Nelson's Xanadu system formally defined the term "hypertext" (as "non-sequential writing") [20], as well as the notions of links and connections. In all of these cases, the focus is on the associative nature of information and how we can exploit that structure for more intuitive information processing -- neither Bush nor Nelson discussed adapting that information based on specific user goals.

More recently, hypermedia models have been extended to include multiple representations of link-types (hot links, warm links, navigational links), node types (anchors, regions) and navigational methods. *Intermedia* (Meyrowitz [16]), developed at Brown University, incorporates a very rich model of hypermedia documents, including the aforementioned linking and navigational structures.

Educational hypermedia systems, such as Apple Multimedia's *Life Story* series on Watson and Crick or *Visual Almanac* (Hooper [12]) are quite good at providing integrated

multimedia displays and methods for navigating them, but like *Intermedia*, the content and format of these presentations are fixed.

Some hypermedia systems approach, but do not reach, a fine-grained level of customization. The Hypertalk programming language allows for the specification of filters for conditional branching within Hypercard stacks. But when the number of imposed constraints for information display gets large, this method of filtering rapidly becomes incredibly inefficient.

Feiner's work on generated multimedia displays [9, 10] strikes the closest to the work described in this thesis. It uses expert system technology to generate the layout and content of text and graphics, based on presentation models developed for specific scenarios. The application they are using to illustrate the system is that of a radio repair manual. Although this type of system requires very little adaptive modification (since the domain is basically small, with a limited set of graphical elements), user experience again plays an important role in being able to format the information. A technician might just need to look up a part number, whereas the home-fixit-whiz might require a bit more hand-holding.

Andries van Dam, in his vision of the Memex of the Nineties [25], emphasizes that in order for hypermedia to secure its place as a standard computing mechanism, hypermedia features must be integrated into the desktop. Some of those features he discusses are

creating, referencing, and associating content, referencing dynamic media, wayfinding, filtering, querying, searching, automatic processing, collaborative efforts, providing semantic structure, and providing standard services. If the eventual goal is to have “a hypermedia infrastructure supporting an international network of information and knowledge for an enormous variety of audiences”, it seems curious that tools to create adaptive information for hypermedia are not being discussed as being in the critical path of hypermedia integration. Is this not an issue essential to desktop hypermedia?

Chapter 2. System Overview

2.1 A Sample Session With the System

Suppose you've been hired to design and produce the hypermedia "tour" of MIT's Media Lab mentioned in the Introduction. This task involves compiling many years of interdisciplinary research into linked multimedia presentations. These presentations must cover the work of all twelve groups housed in the Media Lab -- spanning research areas from Epistemology to Holography to Advanced Visual Design. (See Figure 2.1.)

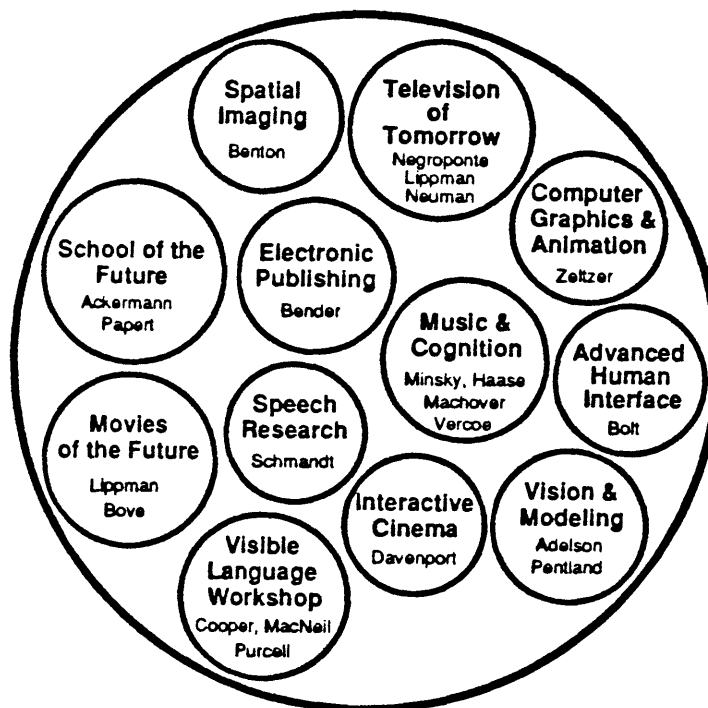


Figure 2.1 Organization of MIT's Media Lab

Given this wealth of information to work with, suppose the task specification stated that this must be an "adaptive" hypermedia system; in other words, it needs to be able to modify the length and display of its presentations based on what the user is currently most interested in, and how long they have to see it. This can be accomplished by designing a *script* for each presentation, including both graphic and semantic information which can be used by the system to affect display and content-based modifications. The system described in this thesis is an implementation of an adaptive hypermedia system and its associated tools, for creating scripts of graphic and semantic information. The following sections will describe the structure of such a system in the context of the Media Lab tour example.

The system structure can be divided into two areas; the modules with which the user interactively designs multimedia scripts (*Authoring Component*), and the module with which the system controls script modification based on dynamic input from a hypermedia-user (*Hypermedia Component*). See Figure 2.2.

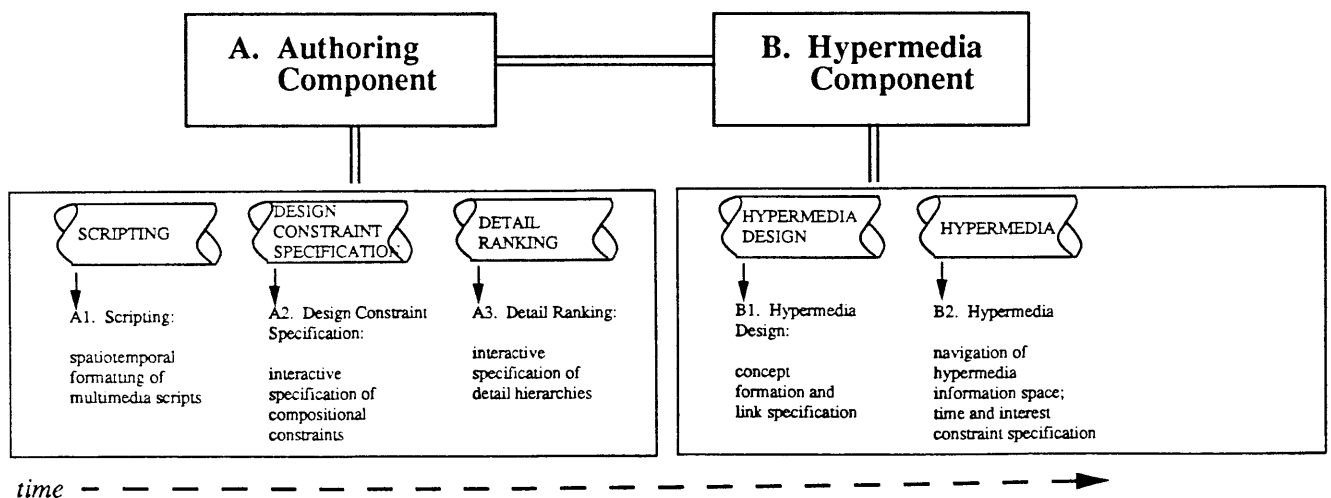


Figure 2.2. System Structure viewed as a pipeline of functionally separate modules.

2.2 Authoring Component

The authoring component of the system is a pipeline of modules with distinct functional differences. Control must proceed linearly through the pipeline to create a multimedia script which can be used by an adaptive hypermedia system. See Figure 2.3.

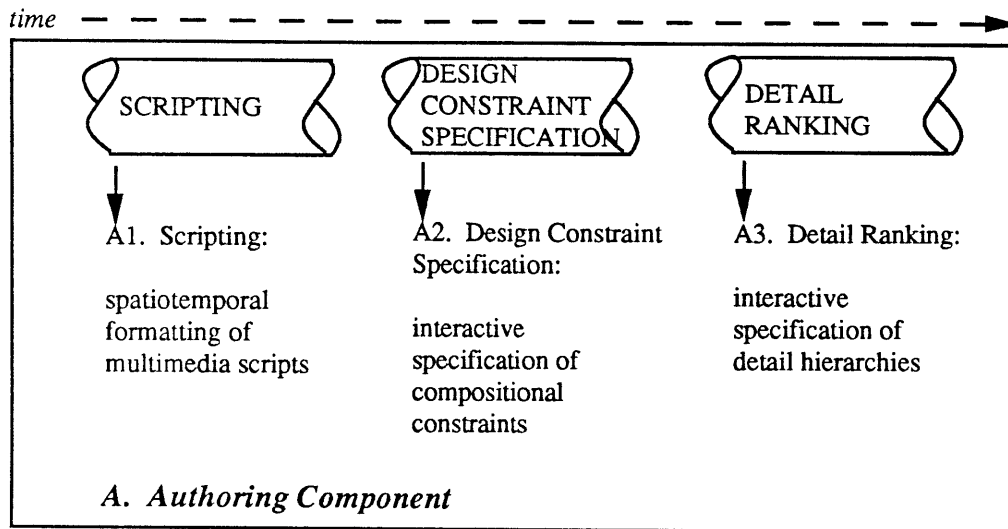


Figure 2.3 Pipeline of functionally separate modules comprising the Authoring Component.

A1. Scripting Module

Suppose that our hypothetical user is ready to start designing the multimedia scripts for the system. She has chosen to focus first on the research of the Visible Language Workshop, and more specifically, on the area of typography. As a starting point, let us suppose that she chooses to first address issues of dynamic font-filtering.

Note, first, that all multimedia resources for script generation have been scanned, sampled, typed, or otherwise logged into the system prior to system start time. She needs to browse and select from this collection of multimedia objects (text, still images, sampled sound, video and animation) any information related to dynamic font-filtering. Next, she needs to sequence these objects along a timeline, set their durations, and determine the spatial layout of these objects as they would appear in the actual presentation.

To do this, she uses a set of generalized multimedia authoring tools provided by the system. These tools are organized into three environments -- one for browsing a database of multimedia objects (the *Browsing Environment*), one for sequencing and specifying durations (the *Scripting Environment*), and one for laying out the objects and modifying their media-specific attributes (the *Playback Environment*). (These environments are discussed and illustrated in full detail in Chapter 5 -- *Scripting Tools*.) Collectively, these tools comprise the *scripting module*. The end-product of an interaction with the scripting module is a file of human-readable spatiotemporal formatting data which can be played back by the system in the form of a multimedia presentation. See Figure 2.4.

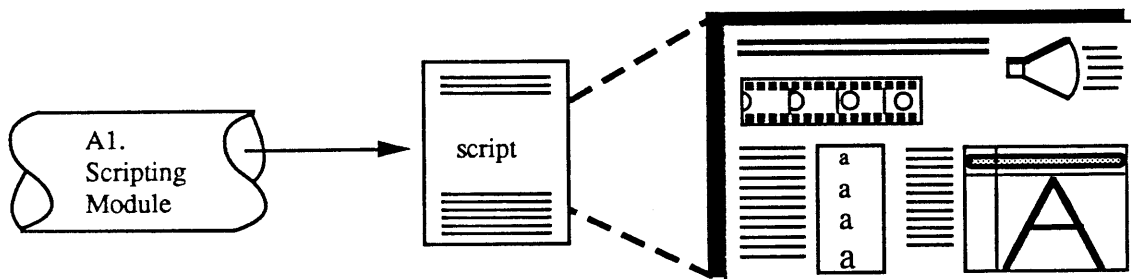


Figure 2.4. Using the authoring tools in the scripting module, a script designer can produce a script -- a file of spatiotemporal formatting data, which can be "played back" as a dynamic multimedia presentation.

Let's say our user selects two bitmaps (grabbed from font-filtering demos), three text blocks (a title, and two paper excerpts), a video clip (of a dynamic demonstration), and a piece of explanatory narration from the browsing environment. She sequences these objects with respect to the presentation timeline using a track-based storyboard. The following is an illustration of how the storyboard would look for this example:

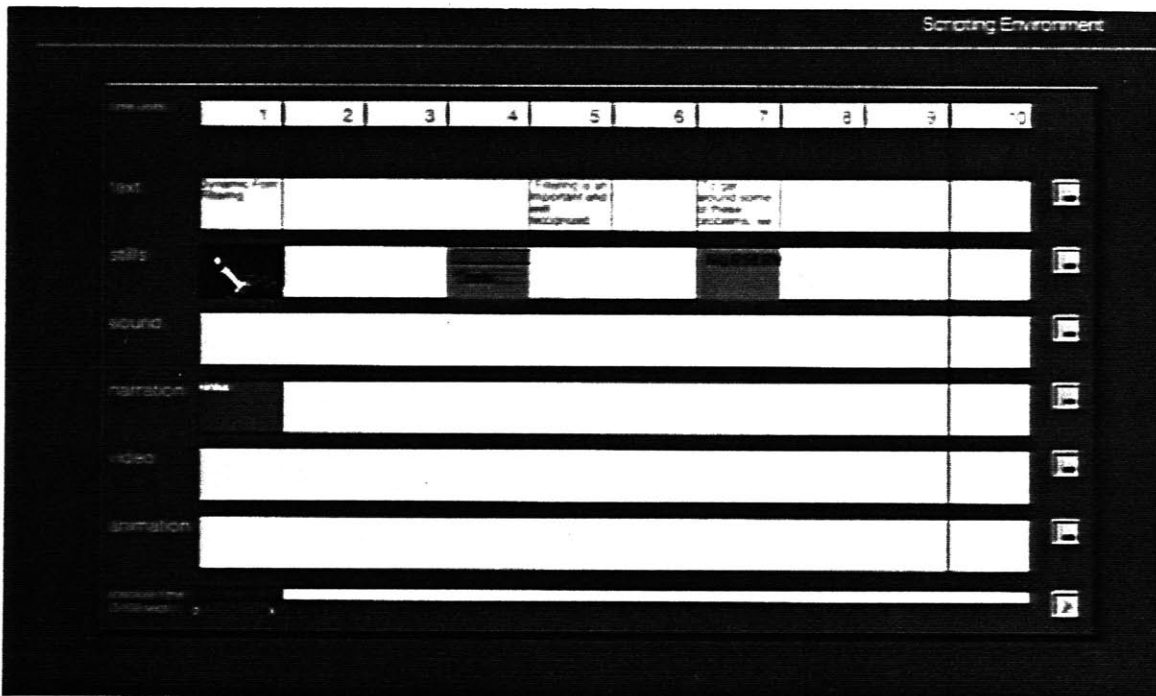


Figure 2.5. The state of the screen after the script-designer composes multimedia objects on the electronic storyboard.

She then sets the durations of each object to coincide with the chosen narration. Finally, she uses the playback environment to lay out these objects as they would appear in a final presentation. She changes the font and point size of the title to one larger and bolder than that of the paper excerpts. She reformats the text blocks so that they form columns, appearing adjacent to the bitmaps they refer to.

The following is an illustration of the playback area as it would look after her edits:

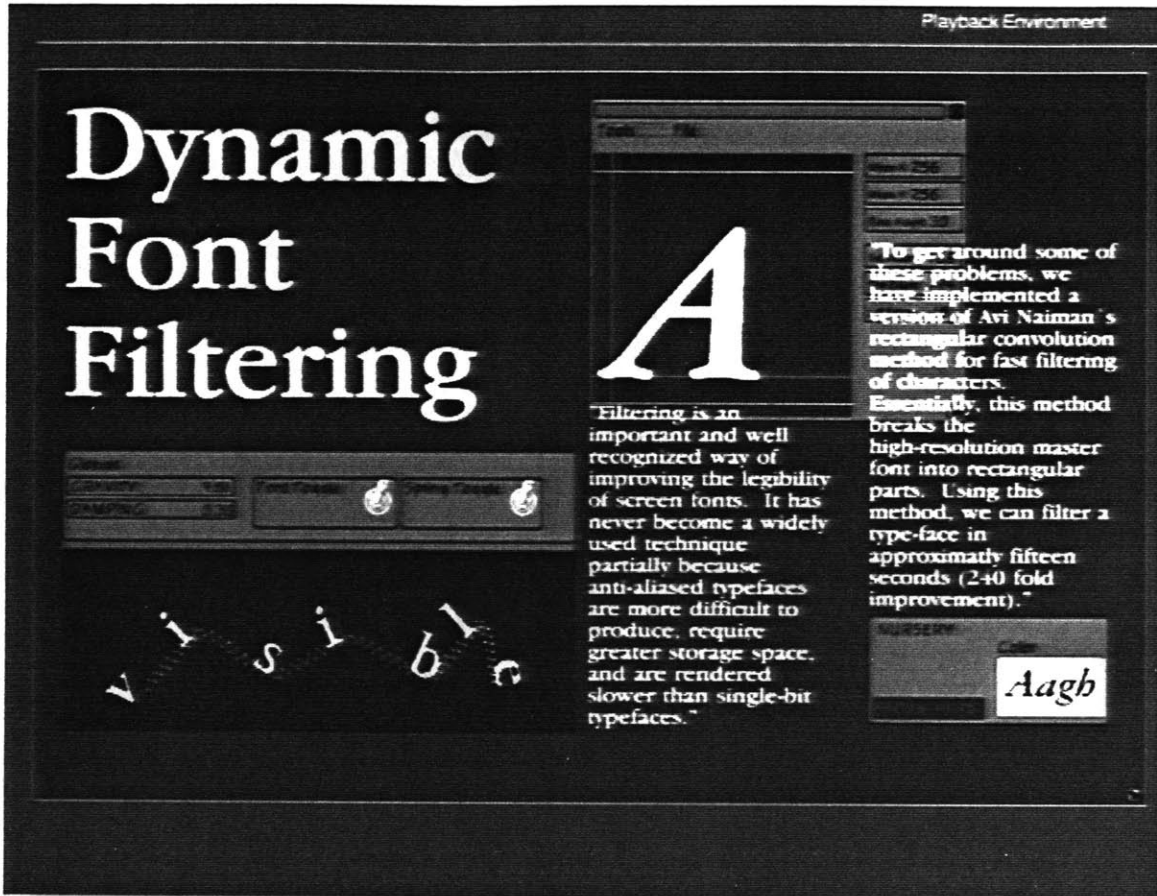


Figure 2.6. Playback area after script-designer lays out and edits the media-attributes of each object.

Once satisfied with the spatiotemporal format and attribute editing, she needs to save out a record of all her actions. By selecting a "save script" option in the storyboard environment, she creates a file on disk of formatting information which can be reloaded at any time. (See Appendix C for a listing of a sample script.)

A2. Design Priority Specification Module

Next, our user needs to specify certain design constraints and priorities about the scripted material, so that the system has a means by which it can intelligently modify the script. Script modification consists of either pruning out or including selected multimedia objects in the presentation, based on a temporal allocation scheme and search heuristics. (To specify a constraint or priority implies that the script-designer may override any actions the system would normally take based on its heuristics.) Priorities can be spatial, temporal, or content-related in nature. Typically, the design priority specification module is used for ensuring both graphical and semantic continuity in the presentation. (See Chapter 4 -- *Constraint Management* for further discussion of the constraint-specification interface.)

To continue with our example, let's imagine that our user decides that the chosen bitmaps do not communicate effectively without their associated paper excerpts. In order to ensure that the system does not decide to prune or include either a text or bitmap object without its complement, the user can *group* the objects, thereby specifying a compositional priority, ensuring both graphic and semantic continuity. See Figure 2.7.

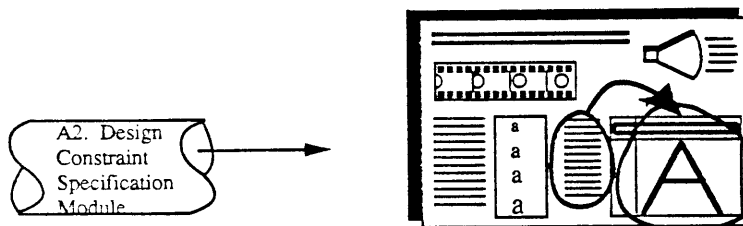


Figure 2.7. Using the Design Constraint Specification module, the script-designer can specify compositional priorities that will override any constraints imposed by the hypermedia user at run time. This example illustrates specifying an adjacency priority.

A3. Detail Ranking Module

To further aide the system in its script modification decisions, the script-designer is asked to analyze the conceptual structure of the script, and delineate boundaries of "detail".

(What constitutes "detail" is determined by the script-designer; detail delineations most often fall on conceptual boundaries, but could fall on media-type boundaries as well.)

The system uses the resulting hierarchical model of detail for its pruning heuristics, the notion being that pruning "more-detailed" information out will still leave the system with a conceptually continuous script.

The script-designer assigns each object a *rank*, organizing the multimedia objects into a "detail hierarchy", where the nodes at the leaves of the tree represent the finest granularity of detail. To play the "most detailed" version of a script, the system does a depth-first traversal of the detail hierarchy. The "least detailed" version would entail a one-level breadth-first search. See Chapter 3 for further discussion of detail hierarchies.

Suppose the user in our example decides that the title, narration and video clip comprise the least detailed script. The bitmaps, which augment the narration, can be considered a second level of detail; the paper excerpts can be considered a third level. See Figure 2.8 for an illustration of the detail ranking process, and the resulting detail hierarchy.

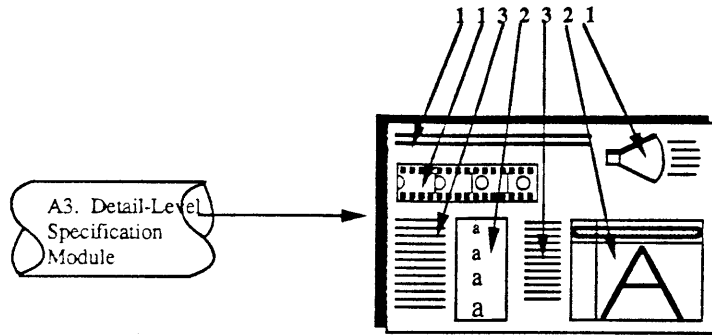


Figure 2.8. a) Detail-Level Specification Module allows for the structuring of conceptual levels of detail into a hierarchy, by ranking individual objects.

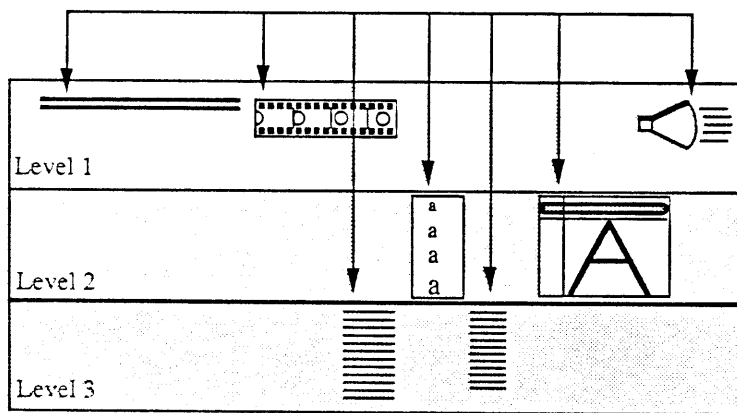


Figure 2.8. b) Detail Hierarchy of above ranking scheme.

Enacting the "Save Script" command again will incorporate all the design constraints and priorities, as well as the detail-ranking information into the script data file. Our user has now created a script for driving a multimedia presentation, structurally sound for being used in an adaptive hypermedia system, as it contains all formatting, design prioritization and detail ranking data.

2.3. Hypermedia Component

Figure 2.9 illustrates that the hypermedia component of the system is used both on a design level (B1), and an end-user level (B2).

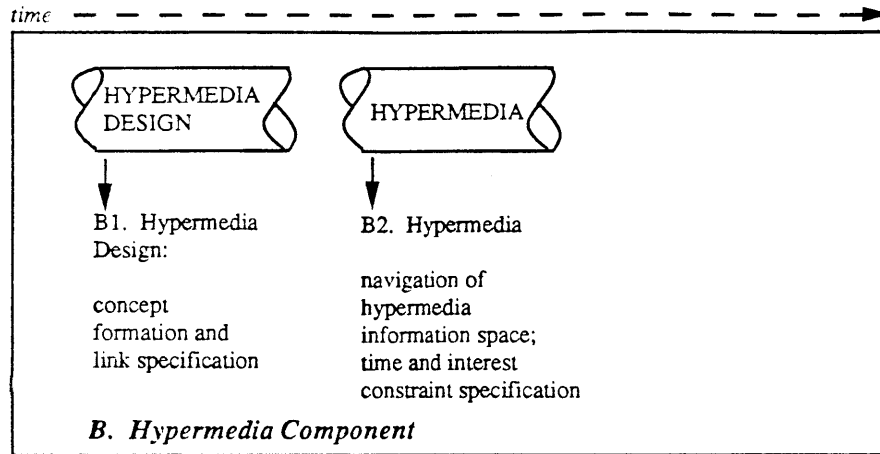


Figure 2.9. Functional Pipeline of the Hypermedia Component.

B1. Hypermedia Design Module

The next task for our user to perform is that of linking the newly created script to the appropriate node of our hypermedia system. This, of course, assumes that the conceptual map (node and link structure) of the hypermedia system has been defined.

In this system, the structure of the example hypermedia system is predetermined, as well as the assignment of scripts to nodes. It was implemented this way for simplicity's sake, but it is well understood that in a production-level system, the user would also take on the role of abstracting concepts from raw multimedia objects and modelling their interrelationships. Thus, the *hypermedia design module* would ideally provide concept-abstraction tools, as well as tools for asserting connections between objects; in particular, the type-instance relation between a concept (node) and its associated media objects (scripts). See Chapter 3 -- *Adaptation Strategies*, for a discussion on hierarchies and concept formation.

For now, we can assume that our example user can interactively link the newly created script with the data structure representing a conceptual node within the hypermedia information space. When someone clicks on this node during a session with the hypermedia system, this script will be played back in some form dictated by how much time and interest is dedicated to this instance of playback.

B2. Hypermedia Module

Finally, our user must make sure that the combination of the presentation format, detail ranking and design prioritization will work in the context of the adaptive hypermedia system. This testing entails playing the role of *hypermedia-user*.

Using a set of interactive constraint specification tools provided by the system, the hypermedia-user can dynamically indicate available time with the system, as well as amount of interest in various concepts. Varying these weights affects the system's display of its multimedia presentations. See Chapter 4 -- *Constraint Management*, for a detailed discussion of temporal allocation strategies.

A "timer" interface is provided so that the user can indicate available time with the system. The timer only runs when a presentation is being played -- it keeps track of the total available temporal resources. A graphical map of the hypermedia space is also provided, called a "subject map", with "interest sliders" associated with each node, and thus, each

concept. The user can indicate levels of interest by adjusting the interest sliders over a normalized range. See Figure 2.10 for an example of a graphical map of a hypermedia information space and the interest sliders associated with each node or concept represented in the map.

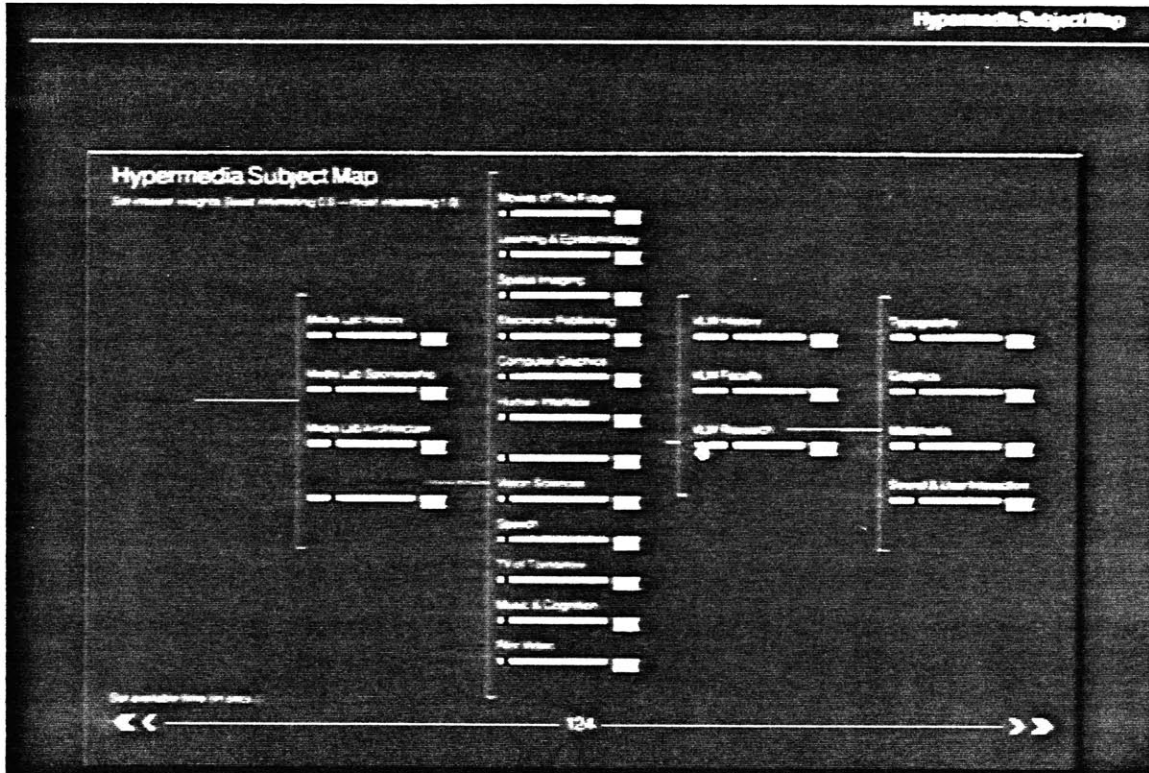


Figure 2.10. "Subject map" for a hypermedia tour through MIT's Media Lab.

In order to test that the system is correctly modifying the dynamic font-filtering script, our user must navigate the hypermedia space to get to the node corresponding to VLW research in typography, and activate that node. The modified presentation produced by the system is formed by pruning the detail hierarchy associated with our script, based on the available time and associated interest weight. To test the adaptive nature of the system, our user can dynamically adjust the time and interest constraints *during* the presentation. Since the system is always polling for changes in the time and interest con-

straints, a presentation in progress can be altered according to the newly imposed constraints. (See Chapter 4 -- *Constraint Management*, for further discussion on time and interest constraint specification.)

A Note on the Development of this System

The example discussed throughout this chapter is based on the author's experiences of the system's development cycle. While constantly having the mindset of system developer -- creating and maintaining the entire pipeline of functionally separate modules, the author doubles as the script-designer, the hypermedia-designer, and the hypermedia-user. The circularity of the development cycle and the dissolution of the boundaries between functionally separate users provided unexpected insight into design decisions affecting the overall system design. It became clear that, in the context of this system, the role of programmer and designer interchanges at many different levels, and that users of the described system might want to take on multiple roles to better affect their tasks.

2.4 System-Level Control

The system takes care of resolving all the imposed constraints at any point in the playback of the presentation. At any given time, the system must figure out how to distribute the available time over the nodes of the system (based on the assigned interest weights), prune the detail hierarchies based on the allocated temporal resources, and resolve these constraints with those imposed by the information-designer. See Chapter 4 for a discussion of temporal allocation strategies.

Chapter 3. Adaptation Strategies

The previous chapter has provided an overview of the functionally distinct modules of the system, and their roles in the creation and testing of adaptive multimedia scripts. This chapter will discuss the various strategies considered for introducing adaptivity to hypermedia systems, including the one chosen for this system.

3.1 Script Versioning

When designing for adaptive information displays in hypermedia systems, one must consider both the structures for representing multimedia information, as well as the methods for performing the script modifications, based on those structures.

One way of designing for adaptive information display is to have the script-designer explicitly create many versions of the same script. The system then determines which version most closely fits the needs of the hypermedia-user at any particular instant.

While offering well-designed solutions at varying levels of detail, this strategy obviously limits the accuracy with which the system can meet users' requirements; a best-fit match is made based on a fixed set of solutions. It is also a drain of human-resources to have

the information-designer make multiple versions of the same script. This is the weakest strategy considered.

3.2 Dynamic Script Generation

Alternatively, the system could dynamically generate the content and layout of the material on the fly, making decisions based on the user's time and interest constraints. This sort of strategy is useful as a means of design automation, but has limitations due to the difficulty of capturing the subtleties of design knowledge as rules. And as Paul Rand[23], one of America's best known designers, says,

*“Graphic design
which evokes the symmetria
of Vitruvius, the dynamic
symmetry of Hambidge, the
assymetry of Mondrian;
which is a good gestalt,
generated by intuition or by
computer, by invention
or by a system of coordinates
is not good design
if it does not communicate.”*

Until we have a much more sophisticated means of capturing and modelling even relatively simple design knowledge, computers will not be designers. Therefore, this strategy is a bit out of reach, although steps to capture and model design knowledge are well underway (Feiner [9], Lieberman [14], Purcell [21]).

3.3 Detail Hierarchies

The information-structuring strategy used in this thesis is to have the script-designer generate a *detail-hierarchy* for each script created. To reiterate, a detail-hierarchy is a tree of multimedia objects, arranged according to “level-of-detail” with respect to a particular concept. (The ranking of each object is performed by the script-designer at script creation time.) The system prunes these trees to generate different presentations, with depth in the tree corresponding to level-of-detail. See Figure 3.1.

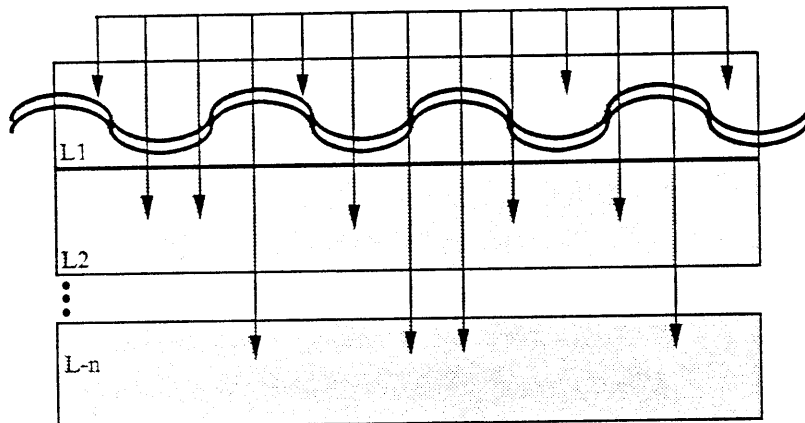


Figure 3.1. a) Level-1 Traversal of Detail Hierarchy (Least Detailed)

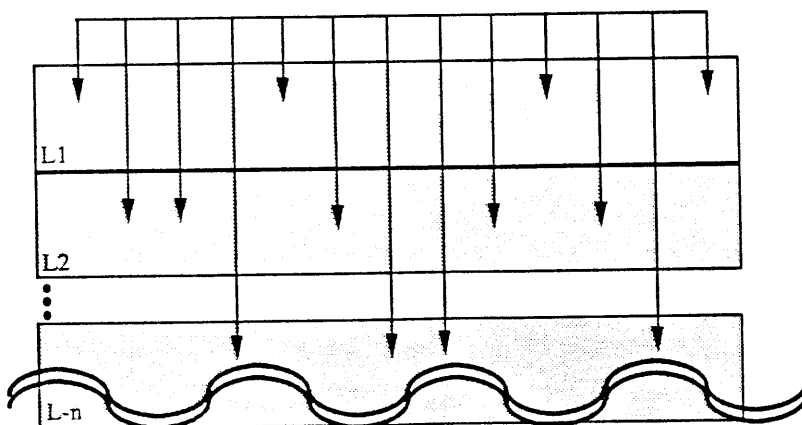



Figure 3.1. b) Level-n Traversal of Detail Hierarchy (Most Detailed)

where  = level of pruning.

Referring back to our example in the *System Overview* chapter, recall that our user has used the detail ranking module to weight each multimedia object used in the dynamic font filtering script, thereby creating a detail hierarchy. The title, video clip and narration byte are ranked at level-1, the images are ranked at level-2 and the paper excerpts are ranked at level-3. If, in the course of running the hypermedia system, the constraints dictate that the least detailed script need be played, the system does a one-level deep depth-first traversal of the detail hierarchy. This would result in the user hearing the narration, while seeing a title and a video clip. If a more detailed version is requested, the depth of the traversal becomes proportionately deeper. The most detailed version of the script would entail a complete depth-first traversal, resulting in the user hearing the narration, and seeing a title, video clip of font filtering in action, two stills of frames of filtered, antialiased fonts, and two paper excerpts.

3.4 Hierarchies and Concept Formation

The use of detail-hierarchies as a means for structuring and filtering related content is supported by literature regarding concept formation. Vygotsky, a Russian scholar and philosopher of the early 20th century, wrote quite extensively about concept formation. He contrasted the notions of an “advanced-concept” and a “complex”. According to Vygotsky [26], a *complex* consists of a number of objects connected by factual bonds, where each bond is functionally equal. An *advanced concept*, he states, emerges only when these objects are analyzed and traits are abstracted. Without the hierarchy formed

from abstraction, each of these bonds is indistinguishable from any of the others, he says.

Herrstrom and Massey [11] also support the use of hierarchies in concept representation. Similar to the philosophy of Vygotsky, their research shows that hierarchical representations of information spaces give shape and structure to concepts which users will find more accessible and predictable. A linked web of information (a complex) is too loose for most hypermedia users, they claim. It invites unguided exploration, which frequently results in user disorientation -- again, the user has no conceptually unifying thread to follow.

The following diagram (Figure 3.2) sets the structures utilized in this system in the context of Vygotsky's concept-formation structures.

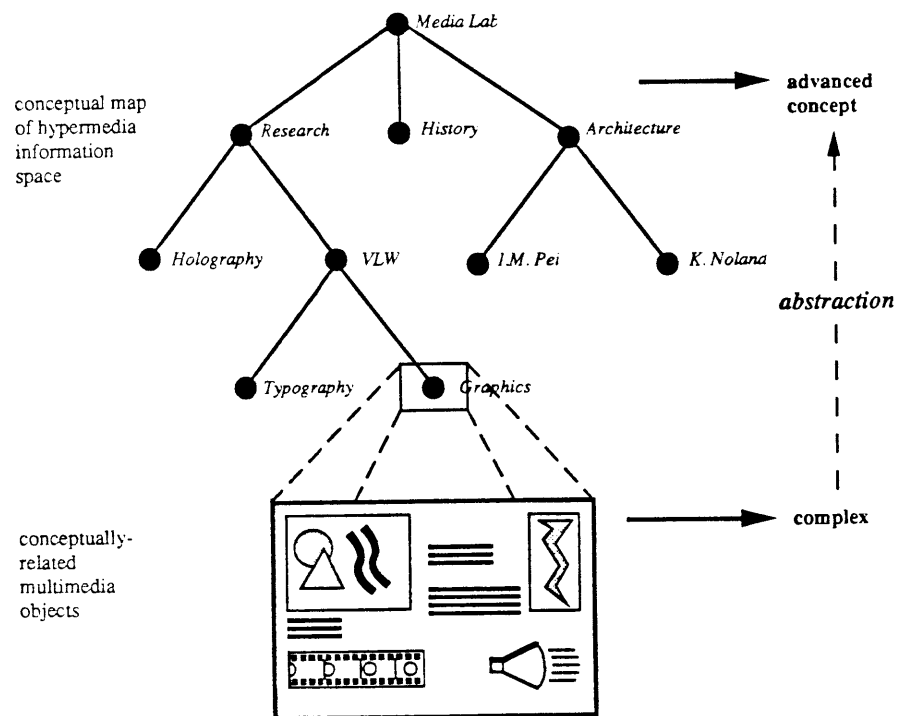


Figure 3.2. Vygotsky's principles as they relate to *Personal Hypermedia*.

The concept map of the hypermedia space is analogous to Vygotsky's advanced concepts when viewed as an abstracted hierarchy of concepts; the set of related objects at each node of the concept map is analogous to a complex, when viewed as a group of associated objects with functionally equivalent links. The process of abstraction provides relational ties between these multimedia objects and their associated concepts.

3.5 Detail Hierarchy Specification

In order to create detail-hierarchies, the script-designer can either draw conceptual boundaries or media-based boundaries. To create conceptual detail boundaries, the script-designer must have a good understanding of which objects form a complex and which form concepts. Objects making up a complex should be grouped on a single level of detail. They are related by the fact that they pertain to a single concept. For example, within our script on font-filtering research there are images illustrating the results of a font-filtering algorithm. They can be grouped as a complex because they relate to the single concept of a particular font algorithm.

On the other hand, the script-designer could decide that some of the information is secondary, just by the nature of its media type. In the case of our example, the script-designer decided that the textual paper excerpts were auxiliary information -- that they reinforced what was already being stated in the narration.

By having the script-designer do the abstraction to form hierarchies of detail, the system is therefore not responsible for finding information phrasing based on either conceptual relationships or media-types of the objects involved in the script. While one could argue that this level of specification is too detailed for the script-designer, it could be said that concept formation is a too subjective a process to be automated. Even if we were to implement multiple models of concept-formation styles, it is clear that rules could not necessarily capture all of the subtleties of human semantic chunking.

The interface for detail specification involves the direct manipulation of the icons for multimedia objects on the storyboard. When clicked on, the individual icons prompt for a new detail ranking. Setting detail ranking on the storyboard allows for informational views of the objects with respect to temporal sequence and media-type, factors which must be considered when setting the detail ranking. Scripts can generally be broken down over time into “information phrases”. The curve of the detail ranking tends to deepen in the middle of each phrase and return to the starting level at the end of the phrase. Thus, it is important to work with the data as it is represented over time.

A second interface for detail-ranking was implemented which allows a view of the objects’ detail-rankings with respect to one another. A histogram-style display is created, which provides a single-glance view of the levels-of-detail for an entire script. On the other hand, this style of information display does not accurately indicate starttimes, durations, or media types. It was found that both types of information display are useful.

Chapter 4. Constraint Management

As previously discussed, both static and dynamic constraints (imposed by the script-designer and the hypermedia-user) affect the display of the system's multimedia presentations. The constraint management module is responsible for affecting the constraints, and resolving any conflicts which might occur due to conflicting prioritizations.

Currently, the constraint resolution is completed by procedural methods. The use of rule-based expert system technology was explored as a means of constraint resolution, but it was determined that the temporal allocation strategy chosen was best implemented procedurally due to the algorithmic nature of the required rules. (Eventually, when the designer constraint specification becomes more complex, the constraint resolution will have to be resolved by forward or backward chaining methods, as these rules must represent more complex inferencing data.)

The constraint management module operates on three specific classes of assertions: time-based, interest-based, and designer-based. The time and interest weights assigned to each node or concept of the hypermedia system determine the amount of detail presented; the designer-based prioritizations override decisions made by the detail-pruning mechanism.

Thus, the designer-based prioritizations hold higher weight than the priorities set by the detail-pruning mechanism. The following sections discuss how the time and interest values are distributed to each node in the system, as well how the script-designer specifies priorities.

4.1 Temporal Distribution

The amount of time allocated to each presentation at any given moment is a function of the user's available time with the system and the amount of interest at each node. As previously mentioned, the hypermedia-user uses the clock interface to dynamically increase or decrease the amount of total available time.

When the hypermedia information space is structured hierarchically, temporal distribution is straight-forward. See Figure 4.1.

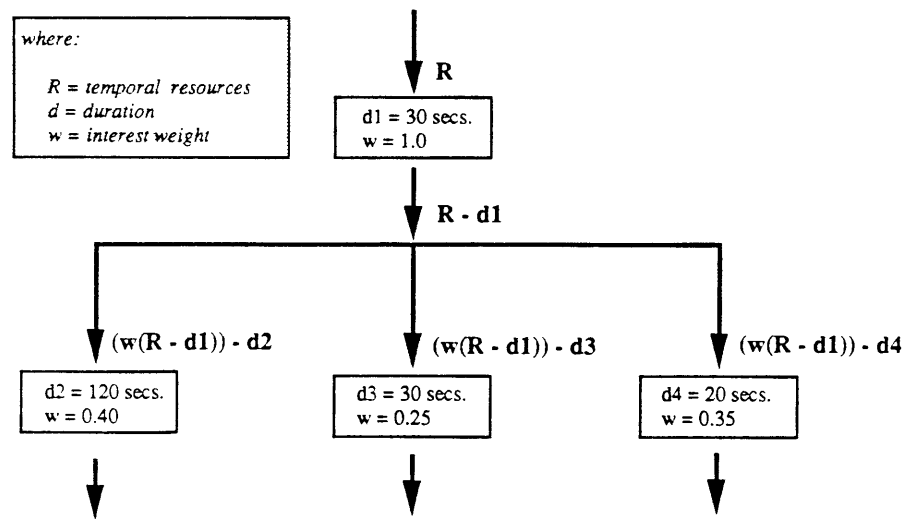


Figure 4.1. Distribution of Temporal Resources over Hypermedia Content Hierarchy

A value, representing total time, is funnelled into the root of the tree. The amount of time required for the presentation at that node is subtracted, and the remaining time is distributed completely over that node's most immediate children. This is done by multiplying that remainder by the interest weight at each child node. This process is repeated down each level of the hierarchy, until the total time is completely parcelled out. When the hypermedia-user dynamically changes the total time or the value of the interest weight at any node, the system automatically redistributes the total available time, based on the current set of interest weights.

4.2 Interest Distribution

In order to ensure that the time is distributed completely at each horizontal tier of child nodes, the interest weights at each horizontal tier are constrained to sum to a constant (1.0). The system provides a set of "equilibrium sliders", where in any given set of sliders, the total value distributed over the sliders is conserved. When any one slider in the set is increased, the other sliders decrease their values by some value proportional to their original value.

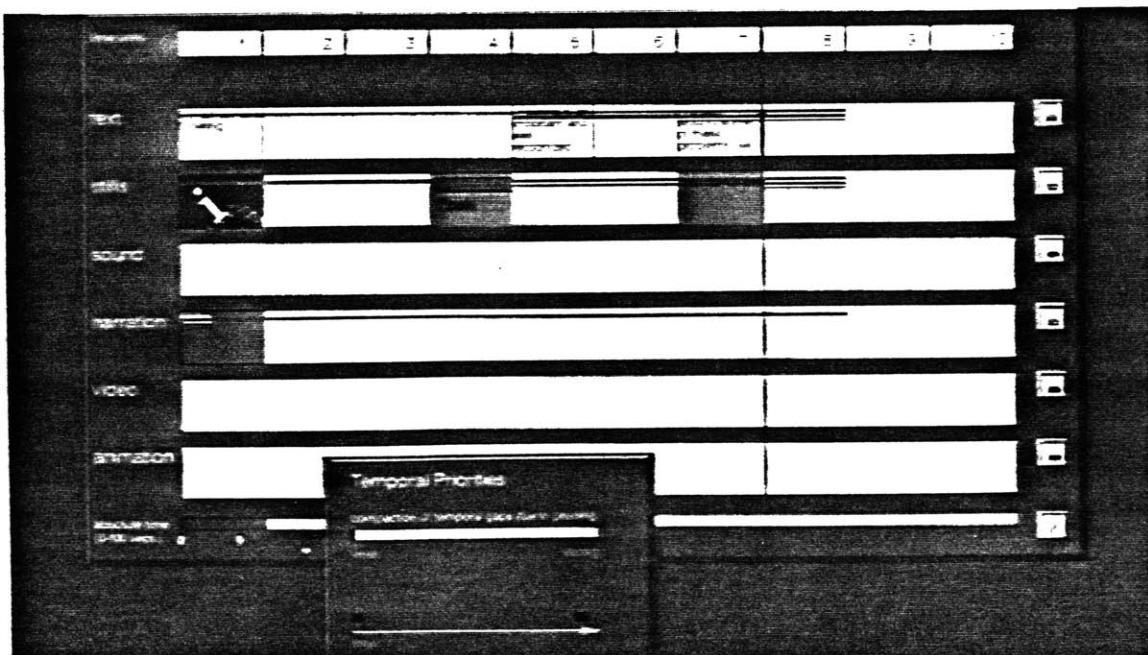
This way, any resources that come down the parent pipe are distributed completely, in proportion to the user's interest. If the interest levels change during a presentation, time is reallocated dynamically.

4.3 Design Constraint Specification

Script-designers must have ultimate control over information display conditions. If the system is modifying scripts based on a set of heuristics and assertions, the script-designer must have the means to override the system's decisions.

In this system, interfaces were developed so that the script-designer could indicate these priorities by directly manipulating the objects themselves. The available interfaces for prioritization fall into two groups -- temporal and compositional.

The temporal prioritization interface allows the script-designer to adjust the way the system compacts multimedia objects over time when a pruning has occurred. By default, the system completely compacts objects over time to fill temporal gaps created by pruning. This interface allows the script-designer to specify any degree of compaction, from strong (no temporal gaps) to weak (no compaction). See Figure 4.2.



The compositional prioritization interface allows the script-designer to specify groupings of objects such that if one object appears in the presentation, the other must also appear, despite detail rankings. This way, the system's rules for pruning according to detail-level can be overridden, based on contextual or conceptual groupings.

4.4 Script Modification Strategies

Once the time and interest weights have been set, and the designer-based priorities have been specified, the constraint management module swings into action, making script modification decisions based on particular strategies. These strategies will vary, based on the goals of the particular hypermedia system.

Interest Satisfaction Strategy

Some hypermedia systems might not be very concerned about keeping their information displays within stringent temporal boundaries. In educational or tutorial systems, for instance, it is more important for the user to be able to control the amount of detail shown of specific subjects than it is to adhere to a specific time schedule. The available time measurement is used as a relative measurement instead of a literal interpretation of temporal thresholds. It could be used to dynamically indicate that the user wants to spend “more” or “less” time with a particular presentation. Changing the available time changes the amount of resources allocated to a script, resulting in longer or shorter presentations with respect to the original script.

In the case of these systems, the adaptive information display algorithm is based solely on the value of the interest weights. These weights are used as indices into the amount of information shown. If the interest weight at a particular node was 0.70, the system would try to determine what constituted seventy percent of available information in the script. Unless each individual object was weighted as to its richness of information, the process to determine what constitutes a percentage of that information is somewhat arbitrary. Should each object be factored in as a contextually equivalent unit, or should the division take into account the contextual weight or richness of each object? Regardless of the method chosen for determining a percentage, the system disregards literal interpretations of available time.

Temporal Satisfaction Strategy

Other hypermedia environments exist where the most important goal is to stay as close as possible within the user's time constraints -- for example, a system in an airport or transport terminal where strict schedules must be adhered to. Interest weights would be used strictly as a means of breaking down the available time, having no direct indexing effect on the amount of detail accessed. This means that the system would try to create the most detailed script that remained within the allocated temporal resources, which are in turn, directly proportional to the interest weights.

To do this, the system would do a breadth-first traversal of the detail-hierarchy, summing the durations of each object. It must also take into account any temporal overlap and any

overrides on temporal compaction that the script-designer might have specified. As with any breadth-first traversal of a problem space, the time complexity is exponential -- $O(b^{**d})$, where b is the branching factor and d is the depth of the hierarchy. This is the strategy employed in this thesis.

Chapter 5. Scripting Tools

This chapter takes a detailed look at the authoring tools introduced in the *System Overview*. These tools are organized into a pipeline of three environments -- browsing, scripting, and playback. The browsing environment is used to browse and select objects from a multimedia database; the scripting environment is used to sequence and synchronize the objects on a track-based storyboard; the playback environment is used to edit the spatial layout and media-based attributes of each multimedia object. Each environment is discussed in further detail in the following sections.

5.1 Browsing Environment

The browsing environment is essentially a holding place for the script-designer's materials, rather like a palette. Iconized versions of multimedia objects are stacked in containers or "browsers", grouped by media. The script-designer can scroll through these stacks, selecting objects to send to the storyboard. See Figure 5.1.

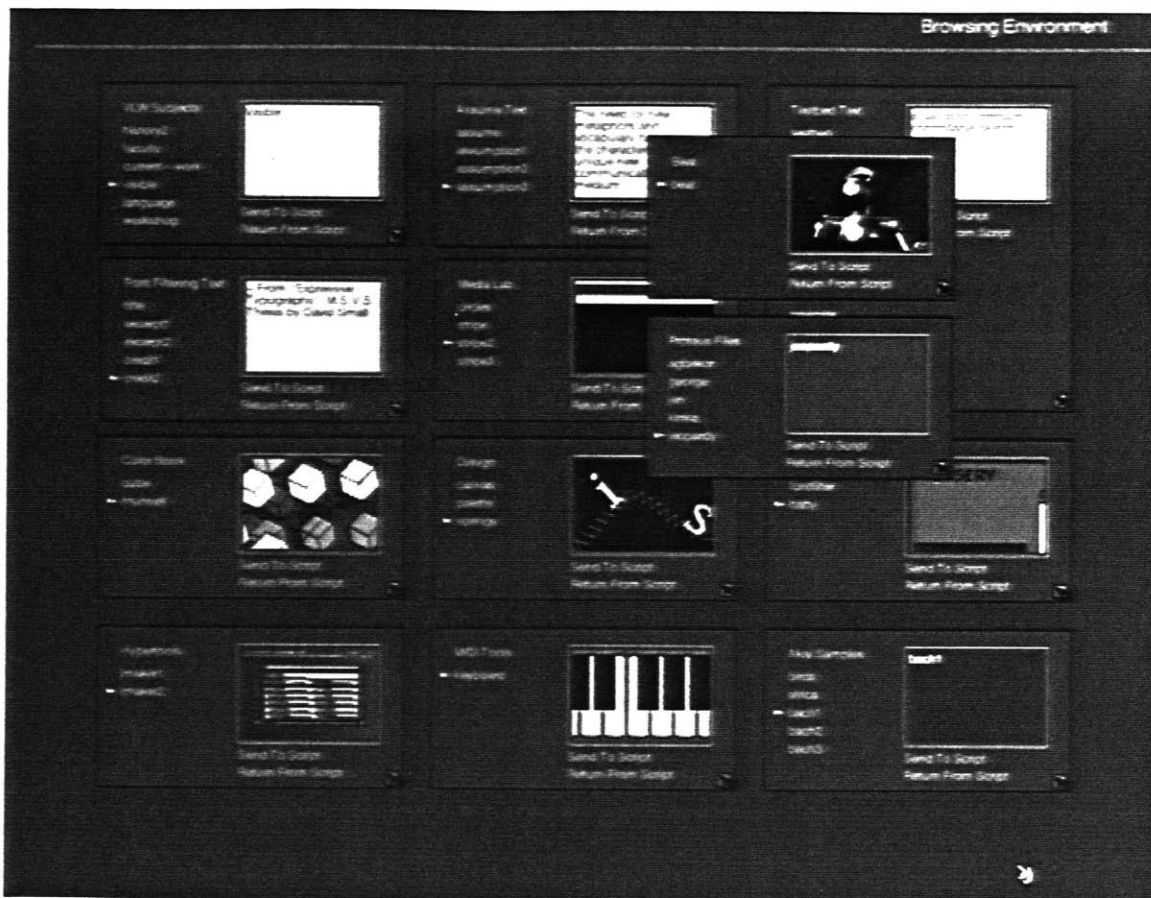


Figure 5.1. Browsing Environment has containers of multimedia objects, grouped semantically and by media.

The multimedia objects available for browsing are pre-logged in a format acceptable for the system. Each object records its media-specific attributes (text objects, for example, record their font, pointsize, color and translucency), as well as their window-specific attributes such as size and position. The icons for these objects are both static and dynamic. For static media, such as text and still images, the icons are simply miniaturizations of the actual object. For dynamic media, such as video, animation, and sound, the icons are “indicators” for that media (video and animation icons use their first frames).

But since these static icons for dynamic media offer no content information, there needs to be a dynamic representation of the object. For video, animation, and sound objects, the script-designer can click on the static icon and see or hear a short clip or sound byte, respectively.

5.2 Scripting Environment

The scripting environment provides a track-based storyboard for sequencing and synchronizing multimedia objects. See Figure 5.2.

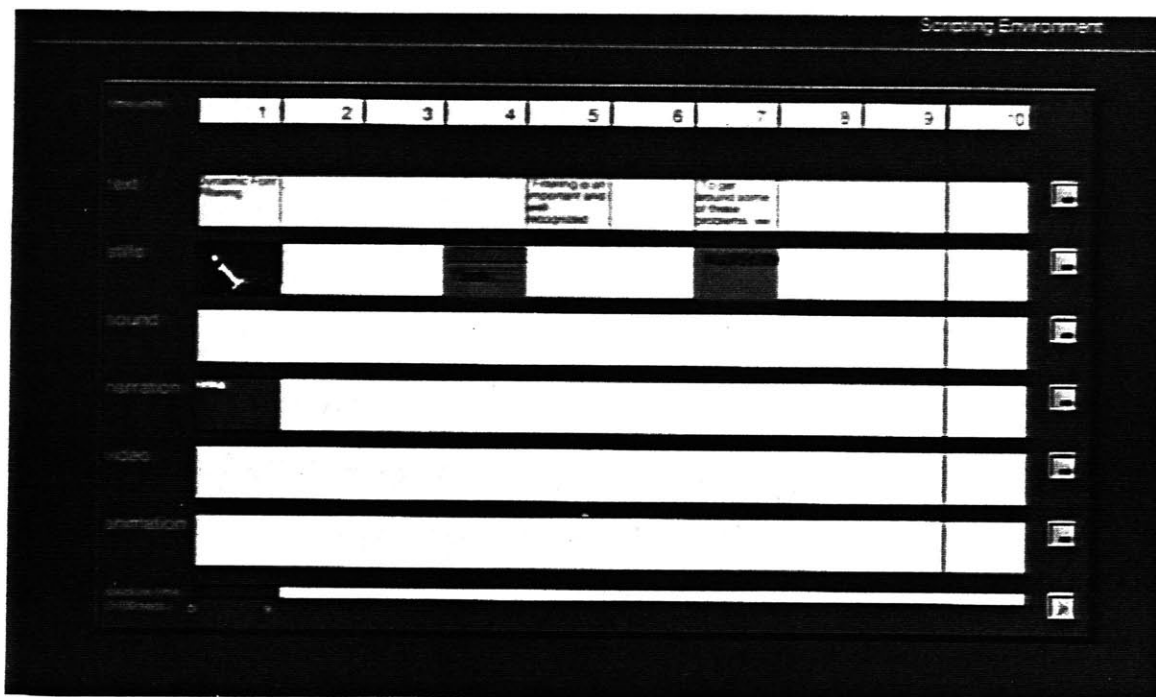


Figure 5.2. Electronic Storyboard for sequencing multimedia objects.

A scrollable timeline runs across the top, with each unit representing some designer-definable unit of time. A single track for each represented media parallels the timeline. Two vertical bars slice through the timeline and media tracks, marking the start and end

play positions. These bars are movable, so that the designer can control the time period being played back. There is a playbutton on the storyboard for invoking the playback.

When the designer chooses an object from the browsing environment, an icon for that object appears on its corresponding media track of the storyboard. The icon is movable and will snap to the nearest time unit represented in the timeline. This icon's position determines its object's start-time. Setting the duration of the object determines its end-time. Each track has a displayable "duration-mode" for inspecting the relative durations of the objects on each track. See Figure 5.3.

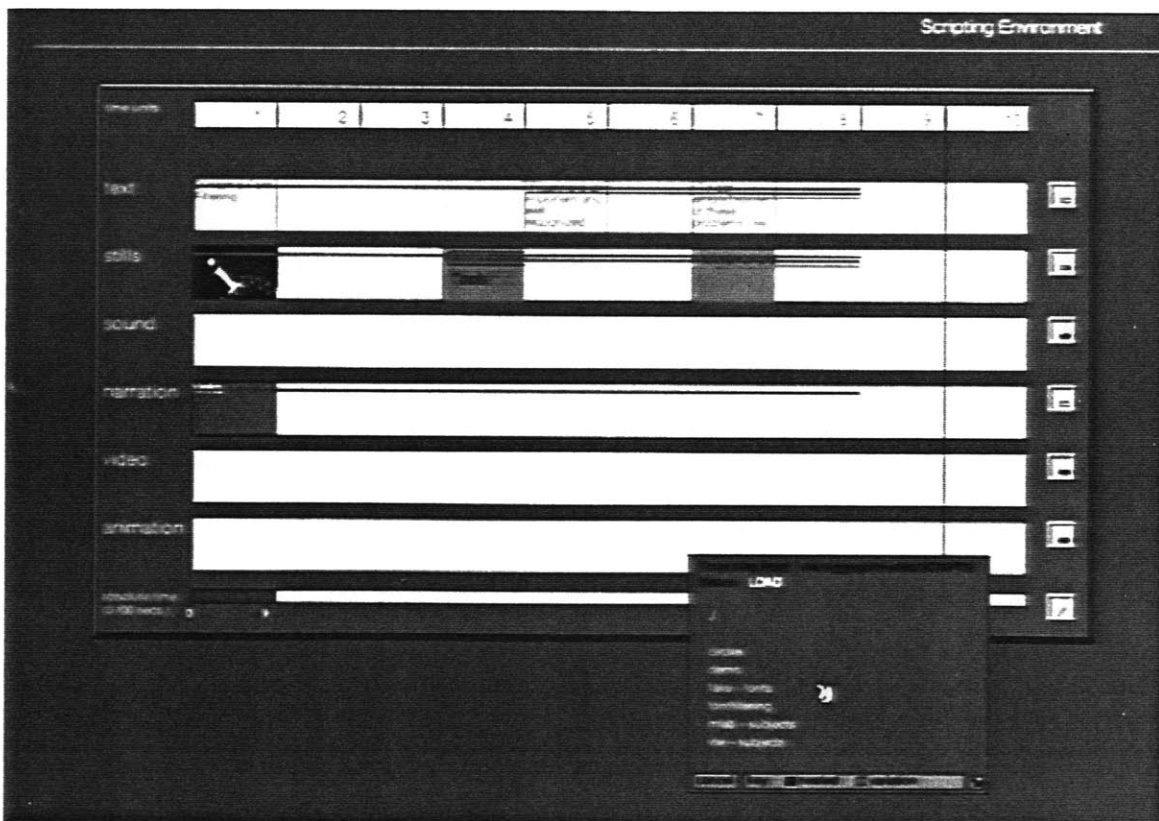


Figure 5.3. Duration mode represented on storyboard by "measuring tapes" extending from each object.

5.3 Playback Environment

The playback environment consists of a playback screen and a number of tools to modify the layout and media-specific attributes of the multimedia objects. See Figure 5.4.

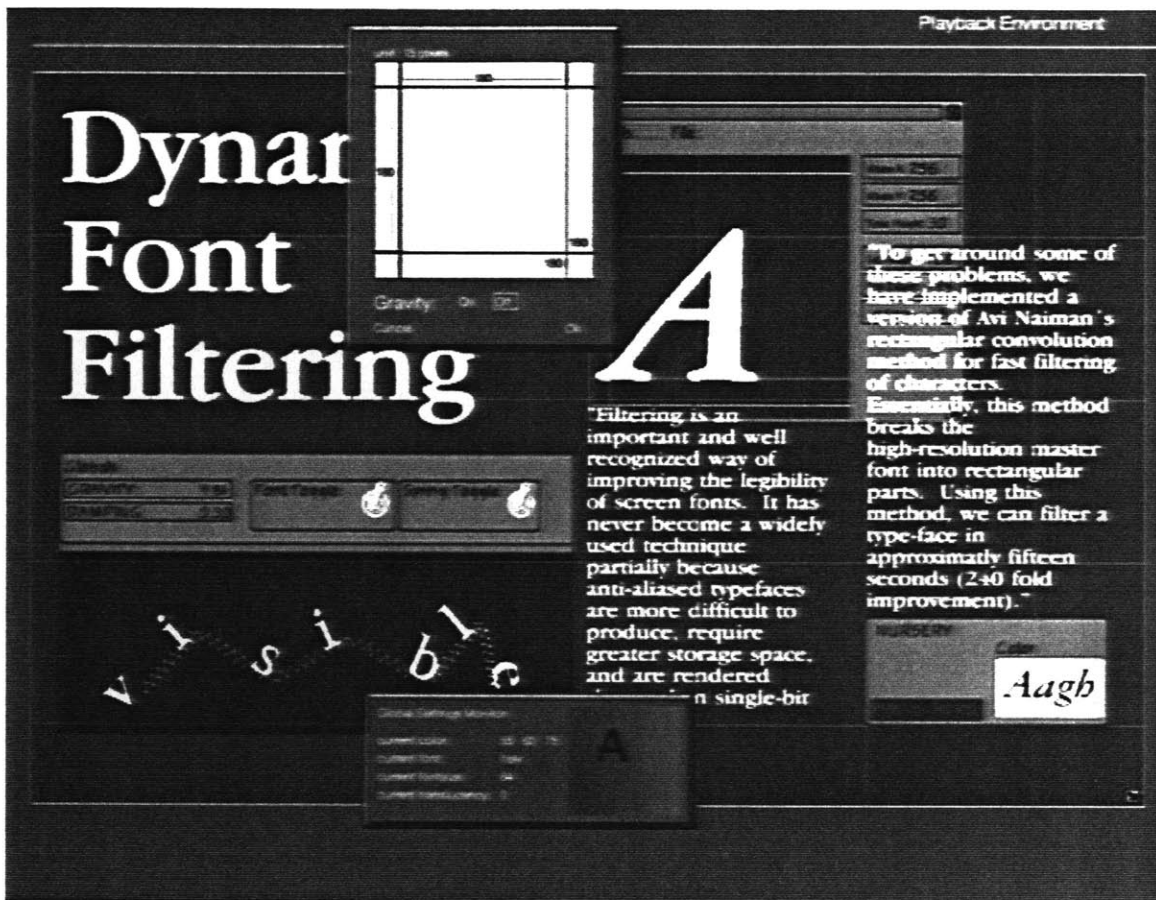


Figure 5.4. Playback Environment shows object layout and media editors.

The playback screen is resizable and can have a solid color background, or a bitmap.

There is a grid which lies on top of the playback window to guide the designer in layout decisions. A grid editor is available to control grid granularity. According to Paul Rand,

“Like the architect’s plan, the grid system provides for an orderly and harmonious distribution of miscellaneous graphic material. It is a system of proportions based on a module, the standard of which is derived from the material itself. It is a discipline imposed by the designer -- a discipline that frees one from the time-consuming burden of making certain decisions (dimensions, proportions) without which fruitful and creative work is extremely difficult. One can move directly to those aspects of the problem in which individual expression, novel ideas, and freedom of choice are essential.”

It is in this light that the grid system is introduced as a way for the designer to analyze the material to be designed and come up with a grid system appropriate for the design task at hand.

The editing tools in the playback environment allow the script-designer to alter object placement, size, color, font and translucency. A “current-values” monitor reflects the current global selections of font, pointsize, text color and translucency.

5.4 Save/Load Mechanism

The authoring toolset provides mechanisms for saving and loading multimedia scripts. The layout attributes and detail rankings of multimedia objects are saved out as ASCII data; these files can be reloaded either singly or in conjunction with other files to create complex scripts.

5.5 Playback Control Mechanism

The playback control mechanism parses a control list and executes attached functions. The control list is a data structure corresponding to each unit in the playback timeline. This structure records which objects start and which end at that structure's corresponding time unit. Each object stores two pointers to functions -- a start function and an end function. A playback involves the linear traversal of this list, executing first the start functions, then the end functions for each object attached to a particular unit of time. See Figure 5.5.

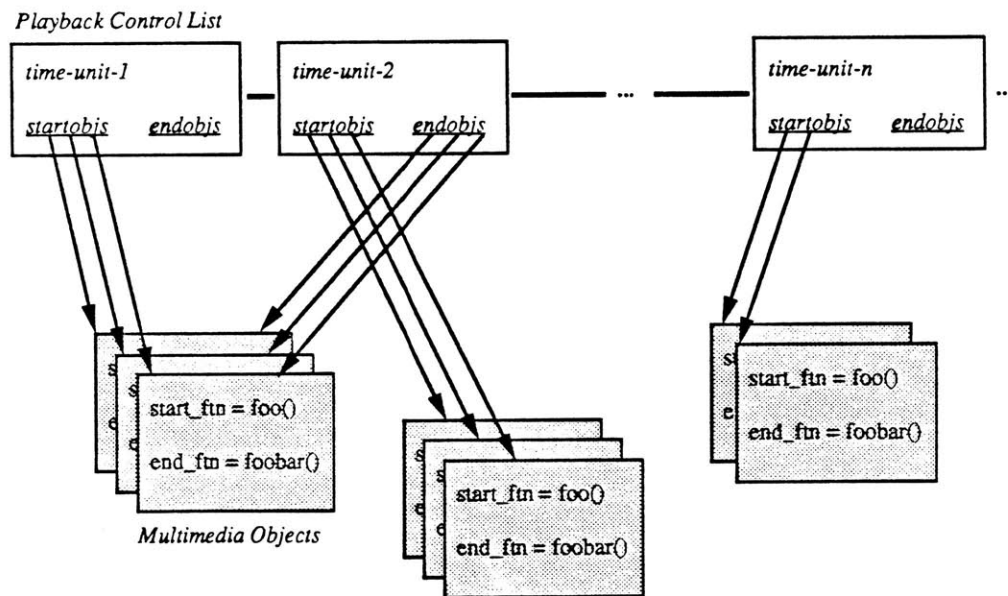


Figure 5.5. Playback Control Mechanism.

The playback control is blind to what these functions do. It simply passes a message to the object to either invoke its start or end function.

Currently, the functions attached to the various multimedia objects do things like open or close windows, send a keyup/keydown message to the AKAI digital sampler, or start a video or animation loop. Object start/end functions can be changed by direct graphical manipulation. A graphical inspector is used to display the attributes and values of the playback control list structures in a form.

6. Conclusions and Future Directions

In conclusion, this thesis has demonstrated the usefulness of adaptive hypermedia systems for accommodating the varying goals of hypermedia users. As more computer users cross paths with these non-linear, associative arrangements of information, there will be a bigger demand for the system to perform with intelligence-based responsiveness. Computers that have enough knowledge about the design of their information displays can dynamically make changes to those displays in a manner consistent with the original form and content of the material. If designed to take advantage of these knowledge structures, adaptive hypermedia systems introduce a symbiosis between those who structure information and computers. Designers of multimedia presentations are given the opportunity to express their design knowledge to the system in the form of abstracted rules and prioritizations through hands on interaction. Meanwhile, the computer builds its knowledge into increasingly complex systems of design rules which it can use for dynamic modification of information displays.

The design rules expressed in this thesis work refer to the spatial and temporal dispositions of multimedia objects. As this system develops, one obvious direction to pursue is

to add rules regarding semantic content. If someone was very interested in a particular subject matter, and the most detailed version of the corresponding script had been played in its entirety, the system should be able to query the multimedia database for the closest semantic neighbors, and format them into a new script dynamically. This would involve extensive preprocessing of all multimedia information, so that all content knowledge was modelled either in the form of existing static links, or represented so that the system could create semantic links on the fly. Morgaine's work [18], an existing VLW thesis, could be used in conjunction with this system to perform content modelling.

Accessing, grabbing, and formatting raw multimedia data for the database used in this system proved labour-intensive and time-consuming. While manual logging techniques cannot be avoided, they can be hidden in other programs which facilitate interactive acquisition of multimedia data. For example, Purcell's Light Table system [22] could be used in conjunction with this system to perform image retrieval in a highly interactive, videographic environment. Light Table provides an interface, connect via LANs to MIT's visual information systems, which allows users to access major collections across MIT's campus. Hypothetically, the link between the two systems could perform the logging operations so that grabbing still, video and animation information from data-rich resources becomes seamless.

Machine learning techniques hold promise for advancing systems such as this one. By using "programming by example" technology with respect to graphic design (Lieberman

[14], Ishizaki [13]), computer systems can record and generalize designers' actions.

These generalizations are modelled symbolically, and can later be applied to future design examples. These techniques could be used for the dynamic generation of multimedia scripts in particular design styles.

The long-range goal of this system is to have libraries of design rules compiled and usable by future designers. The system should be able to handle the complexity of a massive design rule-base with expert system technology, including relaxation techniques for handling conflicts. Additional interactive methods for rule specification need to be developed so that rule specification escapes textual, form-based input, and progresses to more intuitive forms of input such as sketching or gestural input. The structure of adaptive hypermedia systems needs to be extensible so that huge bodies of linked information may be supported and updated. Most importantly, the computer must have the ability to abstract new rules from those specified by designers, so that its role becomes that of a non-passive, learning entity.

To quote a well-coined phrase,

"In the knowledge lies the power..."

Appendix A. Design in the VLW

This appendix provides a glance at what we, in the Visible Language Workshop, view as the future of multimedia and hypermedia design. These perspectives have greatly influenced the shape and direction of this research.

Multimedia Design in the VLW

In the Media Lab's Visible Language Workshop, our focus is not how we can *use* traditional graphic design principles to design multimedia. Rather, the focus is how we can *extend* these principles to accommodate a new medium which demands a new type of design. One of the callings of the VLW is to be prepared to offer new design prototypes, formalisms and vocabularies when we are hit by the flood of electronically accessible information coming our way in the very near future. According to Muriel Cooper, Director of the Visible Language Workshop,

“In an electronic environment, the volume of real-time information will outstrip our ability to process it. The use of graphics as a filter for this complex information, as a means of making it both meaningful and expressive, is the critical challenge of the Visible Language Workshop.”

As we hurtle towards an integrated information environment of wide-area fiber connections, computer users will be dealing in new orders of magnitude when it comes to quantities of text, image, video, sound and voice data. We are literally just around the corner from a multimedia explosion. Most major computer vendors have multimedia and hypermedia platforms that should appear within the next two years, if they haven't already. Multimedia is in schools, museums, medical workstations, businesses, tutorials, documentation, repair manuals... Major vendors are discussing the formation of application-independent hypermedia standards; that is, protocols residing at the system-level so that we can transparently link to and from different applications using common multimedia object formats. As an information society, we are moving closer to a vision set forth by Vannevar Bush in 1945 — an environment where we access, shape and store all available electronic information in a speedy and flexible manner.

At the VLW, we address an issue that far too many overlook; in order to use this information at all, there must be tools to design it well. There is a one-to-one correspondence between good design and effective communication. We look to discover and formalize the new graphics principles introduced by the juxtaposition of static and dynamic media, and the new expressivity offered in a high-resolution, high-bandwidth, computationally-powerful multimedia environment.

Hypermedia Design in the VLW

One of the motivations in VLW research is the fact that in the very near future, computer users will, by necessity, become information designers. The tasks of information designers will include not only temporal and spatial layout, but also the filtering, layering and linking of huge information spaces. By providing high-quality graphic representations of hypermedia information spaces, link structure, traversal maps, and content relations, we provide a conceptually coherent environment for the hypermedia designer. By giving accurate and informative graphic guides, information-web comprehension increases significantly, which directly corresponds to good information design. And since the hypermedia designer and the hypermedia user are frequently one and the same person, we build our tools so that contextually they can apply to both designing and browsing.

Appendix B. Implementation

Development Environment

The coding for this thesis was realized on a Hewlett-Packard Series 9000 model 350 workstation with 16Meg of RAM, and two Renaissance 1280x1024x32 bit framebuffers. This program was developed in C in a Gnu-Emacs environment. The graphics environment was Bob Sabiston's "Badwindows" graphics toolkit (written in C), which supports such advanced interface elements as anti-aliased fonts and translucency. This windowing system was originally developed in the Visible Language Workshop in 1987, and has been continually augmented since then.

We are using an AKAI digital sampler (Model 950S) for sound input, which is connected to the Bobcat via a MIDIC box (serial to MIDI converter). For video and animation resources, frames were digitized using a Parallax card which resides in the MicroVax II present in our Lab. The digitized frames are then transferred to the HP workstation for use in the multimedia environment as "flipbook" animation or video.

This system also has software hooks to Lucid LISP, via a LISP-to-C converter developed in the Visible Language Workshop. The first implementation of this system was driven from LISP, which provided a modular object-oriented programming environment. Yet it was slow driving the graphics, and the benefits of using the FLAVORS object classes available with Lucid LISP did not outweigh the need to have good graphics performance. Thus, the second implementation is written in C, but has hooks to LISP for implementing object-based intelligence modules (mini design expert systems) in the future.

Technical Problems

Operational hardware for supporting sound, voice and video is non-existent on the HP workstations (though a prototype sound card was tried and found to fail, and Parallax has released a prototype video grabbing card for the HP's, but it is available only for an X environment). The workarounds we were forced to resort to inhibited the data-rich environment necessary for multimedia development. It is incredibly time consuming to gather, filter, record and otherwise manipulate multimedia resources for use in the scripting system. This is not a problem unique to this system -- until we have automated content logging, and high-speed, high-bandwidth acquisition to image and video banks, we will always be up against this data bottleneck.

In order to fully extend our research capabilities into the area of multimedia design, we must have the hardware to support the electronic acquisition and manipulation of large sets of multimedia information. It is especially important that we remedy the large gaps

found in the areas of sound, voice and video. We must also break out standalone systems such as this one into distributed multimedia domains.

Appendix C. Sample Script

```
;;;;;;;;;;;;;
;;
;; Written by:      laura
;; Date:          Thu Aug 02 10:27:25 EDT 1990
;; File:         /vlw/siggraph/multilib/SCRIPTS/fake_fonts
;;
;;;;;;;;;;;;;
```

```
script_duration=9
timing=0_75
compaction=0.000000
area_xpos=30
area_ypos=90
area_width=1200
area_height=810
area_rgb=54_54_54
area_border=1
area_border_rgb=150_150_150
```

```
number_of_track1_children=3
```

```
trackicon=title_trackicon
rel_starttime=1
rel_endtime=9
duration=8
detail_level=0
```

```
x_position=30
y_position=0
width=506
height=414
```

```
fontname=Garamond_book
fontsize=120
red=220
green=220
blue=220
trans=0
```

```
trackicon=excerpt1_trackicon
rel_starttime=5
rel_endtime=9
duration=4
detail_level=0
```

```
x_position=600
y_position=360
width=268
height=381
```

```
fontname=Garamond_book
fontsize=24
red=220
green=220
blue=220
trans=0
```

trackicon=excerpt2_trackicon
rel_starttime=7
rel_endtime=9
duration=2
detail_level=0

x_position=900
y_position=180
width=275
height=455

fontname=Garamond_book
fontsize=24
red=220
green=220
blue=220
trans=0

number_of_track2_children=3

trackicon=springs_trackicon
rel_starttime=1
rel_endtime=9
duration=8
detail_level=0

x_position=30
y_position=420
width=555
height=312

trackicon=fontfilter_trackicon
rel_starttime=4
rel_endtime=9
duration=5
detail_level=0

x_position=600
y_position=30
width=406
height=356

trackicon=baby_trackicon
rel_starttime=7
rel_endtime=9
duration=2
detail_level=0

x_position=900
y_position=600
width=260
height=120

number_of_track3_children=0

number_of_track7_children=1

trackicon=kinka_trackicon

rel_starttime=1
rel_endtime=9
duration=8
detail_level=0

x_position=0
y_position=0
width=360
height=180

number_of_track4_children=0

number_of_track5_children=0

max_detail_level=0

Bibliography

- [1] Barrett, Edward, *Ed.* (1989) *The Society of Text*. MIT Press, Cambridge, MA. pp. xi-xviii.
- [2] Bender, W. & Chesnais, P. (1988) *Network Plus*. *SPSE Electronic Imaging Devices and Systems Symposium*, Jan, 1988.
- [3] Bolt, Richard A. (1979) *Spatial Data-Management*. Massachusetts Institute of Technology. Cambridge, MA.
- [4] Bush, Vannevar. (1945) *As We May Think*. *Atlantic Monthly* 7: 101-108.
- [5] Conklin, Jeff. (1987) *Hypertext: An Introduction and Survey*. *IEEE Computer*.
- [6] Cooper, Muriel. (1989) *Computers and Design*. *Design Quarterly* 142: 4-22.
- [7] Dondis, Donis. (1973) *A Primer of Visual Literacy*. MIT Press, Cambridge, MA.
- [8] Englebart, Doug. (1963) *A Conceptual Framework for the Augmentation of Man's Intellect*. in "Vistas in Information Handling". Spartan Books, London, England.
- [9] Feiner, S., Nagy, S., & Van Dam, A. (1982) *An Experimental System for Creating and Presenting Interactive Graphical Documents*. *ACM Transactions on Graphics*, Vol. 1, No. 1, January 1982, pp. 59-77.
- [10] Feiner, S. & McKeown, K. (1990) *Generating Coordinated Multimedia Explanations*. *Proc CAIA90 (6th IEEE Conf. on AI Applications)*, Santa Barbara, CA. March 5-9, 1990.
- [11] Herrstrom, D. & Massey, D. (1989) *Hypertext in Context*. in "The Society of Text", MIT Press, Cambridge, MA. pp. 45-57.
- [12] Hooper, K. (1988) *Interactive Multimedia Design 1988*. Technical Report #13 from The Multimedia Lab, Apple Computer Inc., Nov. 1988.
- [13] Ishizaki, Suguru. (1989) *Example-Based Graphical Programming*. M.S.V.S. Thesis in Media Arts and Sciences, MIT.
- [14] Lieberman, Henry. (1988) *Design By Example*. Unpublished paper.

- [15] Mackay, W. & Davenport, G. (1989) Virtual Video Editing in Interactive Multimedia Applications. *Comm. ACM*, Vol. 32 No. 7. New York, NY.
- [16] Meyrowitz, N. Intermedia: The Architecture and Construction of an Object-Oriented Hypermedia System and Applications Framework. *Proc OOPSLA '86 Object-Oriented Programming Systems, Languages and Applications Conf.*, Portland, OR.
- [17] Meyrowitz, N. (1989) The Missing Link: Why We're All Doing Hypertext Wrong. in "The Society of Text", MIT Press, Cambridge, MA.
- [18] Morgaine, Sylvain. (1989) MIDS: A System for Describing Image Content for Multimedia Design. M.S.V.S. Thesis in Media Arts and Sciences, MIT.
- [19] Negroponte, N. (1975) Soft Architecture Machines. Massachusetts Institute of Technology, Cambridge, MA.
- [20] Nelson, T.H. (1980) Replacing the Written Word: A Complete Literary System. *IFIP Proceedings*. October 1980, 1013-1023.
- [21] Purcell, Patrick. (1988) Computer Environments for Design and Designers. *Design Studies*. Vol. 9 No. 3 July 1988, 144-149.
- [22] Purcell, P. & Applebaum, D. (1989) Light Table: An Interface To Visual Information Systems. Internal Paper. MIT Media Laboratory. Cambridge, MA.
- [23] Rand, Paul. (1985) Paul Rand: A Designer's Art. Yale University Press.
- [24] Schneiderman, B. (1989) Reflections on Authoring, Editing, and Managing Hypertext. in "The Society of Text", MIT Press, Cambridge, MA.
- [25] Van Dam, A., Yankelovich, N. (1990) Electronic Books: User Controlled Animation In A Hypermedia Framework. *Proc. of GBC ACM SIGGRAPH / SIGPLAN Lecture Notes*.
- [26] Vygotsky, L.S. (1986) Thought and Language. Ed. and Trans. by Alex Kozulin. MIT Press, Cambridge, MA.
- [27] Zellweger, P. (1988) Active Paths Through Multimedia Documents. *Proc of the International Conf. on Electronic Publishing Document Manipulation and Typography*. Cambridge, UK.
- [28] Zellweger, P. (1989) Scripted Documents: A Hypermedia Path Mechanism. *Hypertext Proceedings, 1989*.

Acknowledgements

The Visible Language Workshop has been my home for two years. The work and the people have been formative in my intellectual and spiritual development, providing inspiration and encouragement...

I'd like to extend warmest thanks and respect to Muriel Cooper, Ron MacNeil, Henry Lieberman and Patrick Purcell, for believing that I could, and giving me the chance. From each, I learned.

To Sergio, Nathan, and Nynex, I whole-heartedly thank you for supporting my work and providing constant food for thought. See you on the Multimedia trail.

To Geoff, my family near and far, and all the wonderful folks who've stood patiently by and kept my soul lifted, endless gratitude and affection to you...

To Ming, I raise my glass in a salute of comradeship and toast to our success...

And to my flatmates, Syl, Suguru, David, Bob, Didier, Michelle, Skanky, Grace, Laura, Anner, Marie, Alice and Brian, twouldn'ta been the same without any one of you! Continued success, and may our roads cross soon!