RL-TR-96-202
Final Technical Report
December 1996

HYPERMEDIA INTERFACE INTERIM PROTOTYPE

Georgia Tech Research Institute

Ms Susan Liebeskind, Dr Jay D. Bolter, Mr Phillip Hutto,
and Mr Kirk Pennywitt

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-96-202 has been reviewed and is approved for publication.

APPROVED:

SCOTT F. ADAMS
Project Engineer

FOR THE COMMANDER:

JOSEPH CAMERA
Technical Director
Intelligence & Reconnaissance Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL/IRRE, 32 Hangar Road, Rome, NY 13441-4114. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1996	FINAL Oct 93-May 96

4. TITLE AND SUBTITLE

HYPERMEDIA INTERFACE INTERIM PROTOTYPE

5. FUNDING NUMBERS

C - F30602-93-C-0227
PE - 63726F
RR - 2810
TA - BA
WU - 32

6. AUTHOR(S)

Ms Susan Liebeskind, Dr. Jay D.Bolter, Mr. Phillip Hutto, and Mr. Kirk Pennywitt

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Georgia Tech Research Institute
Georgia Institute of Technology, Cmptr Science & Info Tech Lab
Atlanta Ga 30332

8. PERFORMING ORGANIZATION REPORT NUMBER

A-9575

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Laboratory/IRRE
32 Hangar Road
Rome NY 13441-4114

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-96-202

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Mr. Scott F. Adams/IRRE/(315) 330-7788

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report is the culmination of the first stage of development of HyperTech, an interface to multimedia data (text, graphics, imagery, video and audio). Hypermedia capitalizes upon the association between data, permitting data to be linked by context. Hypermedia can be used to produce large, richly connected and cross-referenced bodies of information. The software has multiple methods of viewing data, automated linking based upon text content, and input/output in Hypertext Markup Language (HTML) format. The environment was built upon a relational database. The payoff is an alternative interface which can supplement existing retrieval methods. The work focused on improving access to data on intelligence workstations. Development of the hypermedia environment continues. This Final Technical Report provides an overview of the effort, a listing of other documents resulting from the effort, detailed discussions of the approach, and suggestions for future enhancements.

14. SUBJECT TERMS

Hypermedia, Hypertext, Database, Multimedia

15. NUMBER OF PAGES

28

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Hypermedia Interface Interim Prototype

Final Technical Report

TABLE OF CONTENTS

1. Introduction

Modern intelligence analysis and information management tasks require the capability to administer and manipulate large volumes of text, imagery, graphics and video data. A hypertext system is often the most effective solution for storing and presenting complex and interrelated information. To provide such a solution, the HyperTech application was developed by the Georgia Institute of Technology and the Georgia Tech Research Institute (GTRI) under contract to the US Air Force Rome Laboratory/IRRE in Rome, NY.

HyperTech is a set of "toolbox" routines which may be called by any program to perform hypertext navigation and viewing operations. The HyperTech application described in this document is an integrated hypertext environment which makes full use of the HyperTech engine. HyperTech demonstrates the value of hypertext visualization and hypertext navigation to increase the efficiency of data retrieval, and improve the ease of use and training of information systems. HyperTech is thus intended to present the user with a consistent and easily intelligible view of data elements in one or many documents.

There are three ways of viewing the data elements of documents in HyperTech:
the Text View
the Structure View
the Treemap View

The Text View presents a true WYSIWYG ("what you see is what you get") display of the text and graphics of the elements of a hypertext, as the user would see them in a GUI (graphical user interface) window-based word processor. The Structure View provides a conceptual map of the data; it shows the hierarchical structure of the elements of a hypertext. The Treemap View provides a spatial representation of the size and type of elements of a hypertext.

HyperTech has several ways to support navigation through the hypertexts in the database. The user may navigate along existing paths, or guided tours through the hypertexts, moving forward or backward along the tour at will. HyperTech provides facilities for selecting from the available paths in the system, for navigating along the paths, and for user-controlled creation, modification and deletion of a set of paths.

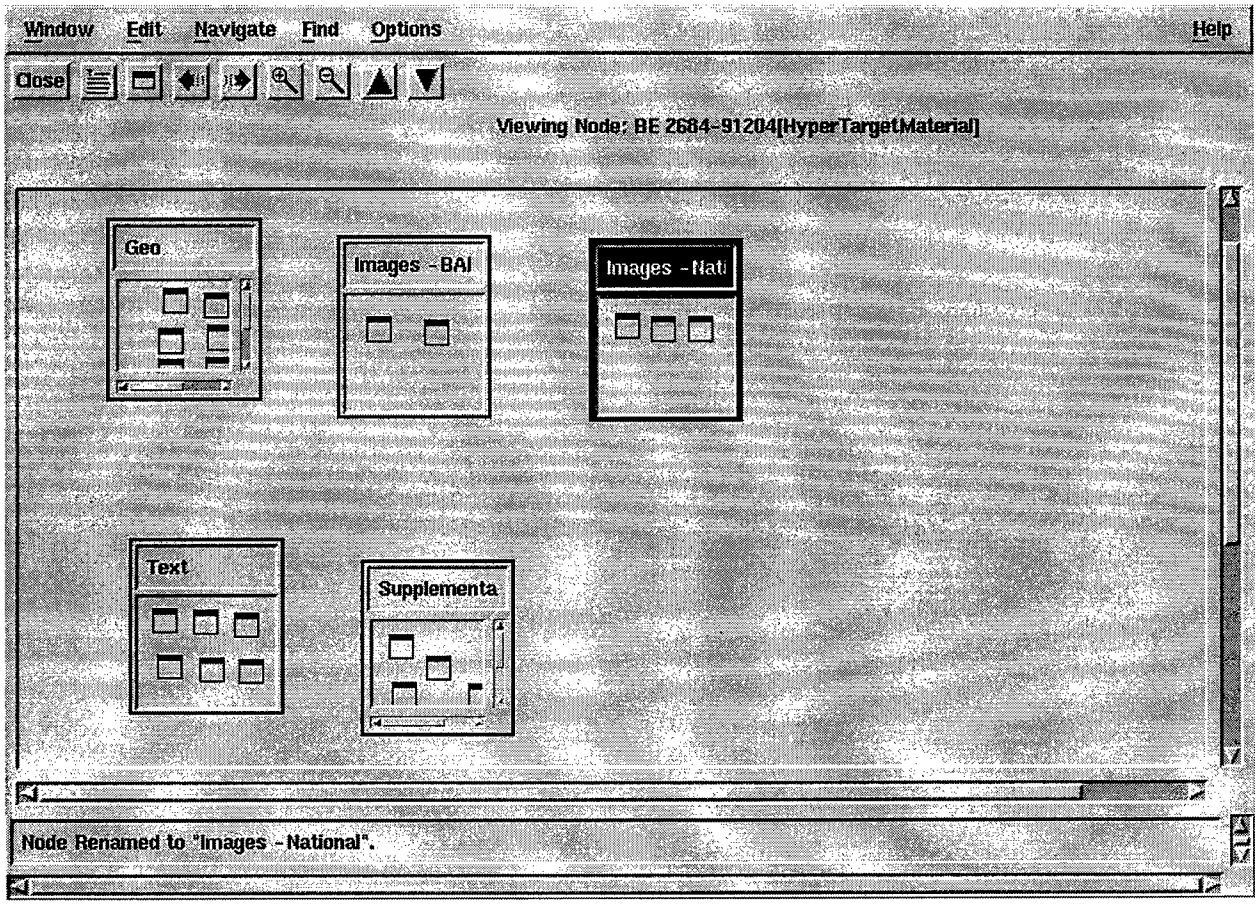
HyperTech can also *automatically* create paths through hypertexts in the database, so that related sections are easily retrieved and indexed. For example, a year's worth of newspaper articles in HyperTech format could be quickly scanned for all articles related to nuclear weapons. The process of automatic path creation may be triggered in the background while the user continues to browse the hypertexts.

This final technical report discusses lessons learned in designing and implementing HyperTech. We provide an analysis of those elements of HyperTech which, in retrospect, could be redesigned or re-implemented, or which were impacted by circumstances beyond our control. Lastly we provide some recommendations for enhancements to and future directions for the HyperTech program.

The Software Design Document, HyperTech Users Guide, and other deliverables mentioned in section 1.1 contain the bulk of documentation for this effort.

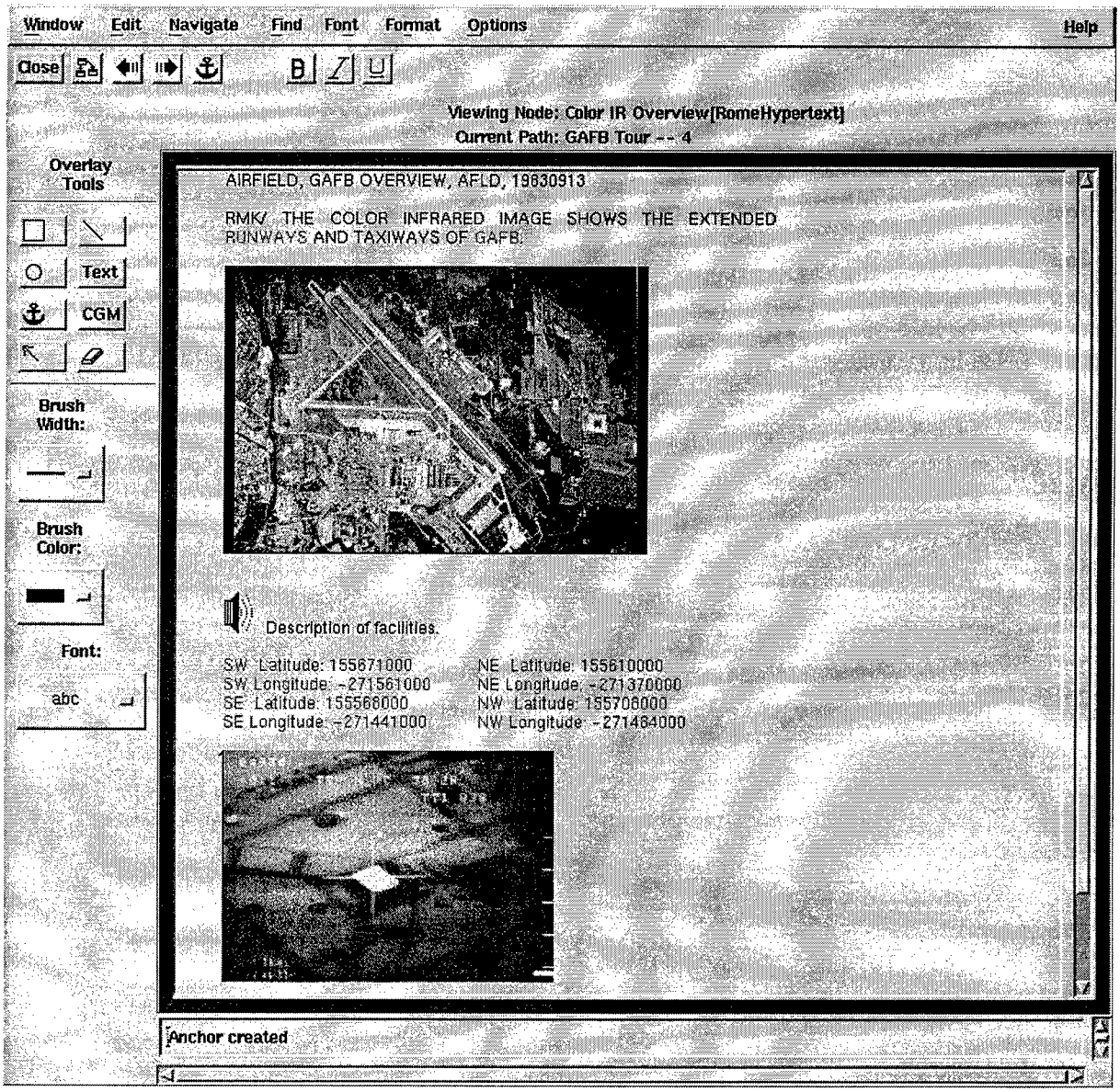
The following pages contain some screen dumps of the HyperTech software to provide an overview of its capabilities.

Structure View



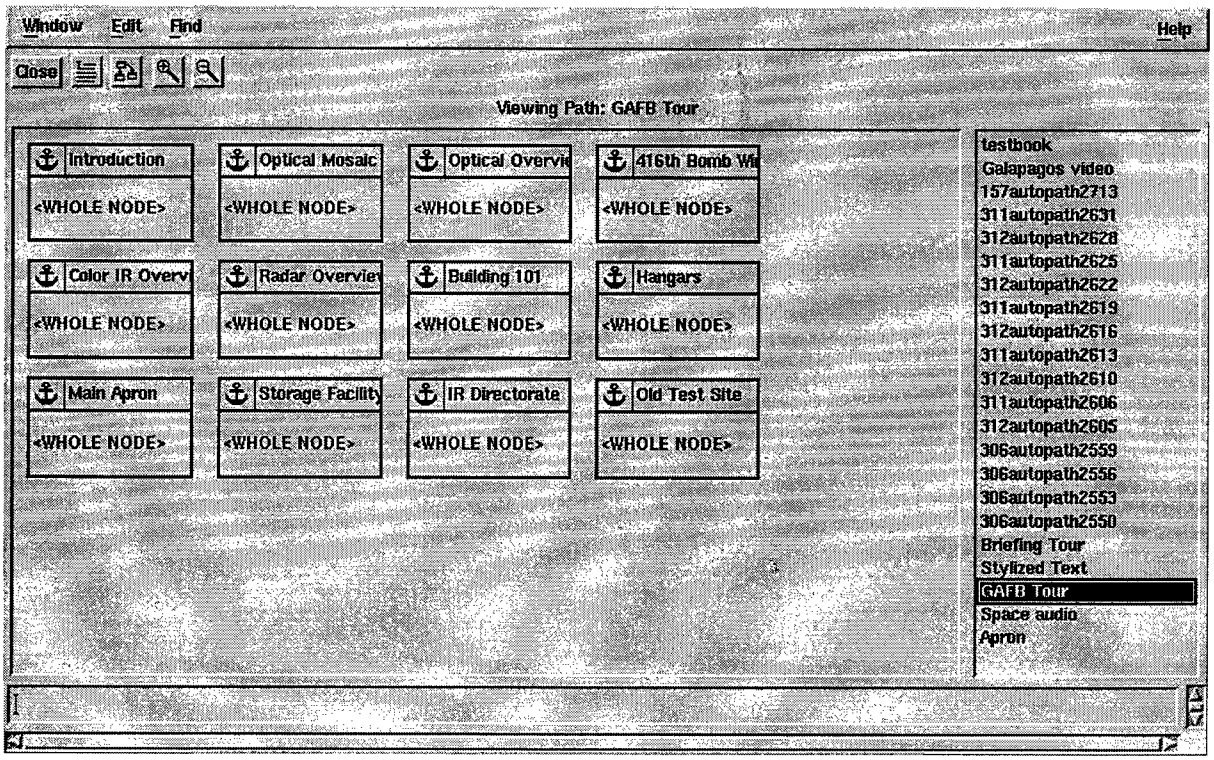
The larger labeled boxes in the Structure View window represent different nodes in the hypertext. Each node may contain text, images, audio, and/or video. The smaller boxes represent children nodes of the parent nodes. The arrow buttons can be used to move about the node hierarchical structure. The structure view allows information to be graphically and spatially organized in an interactive manner.

Text View



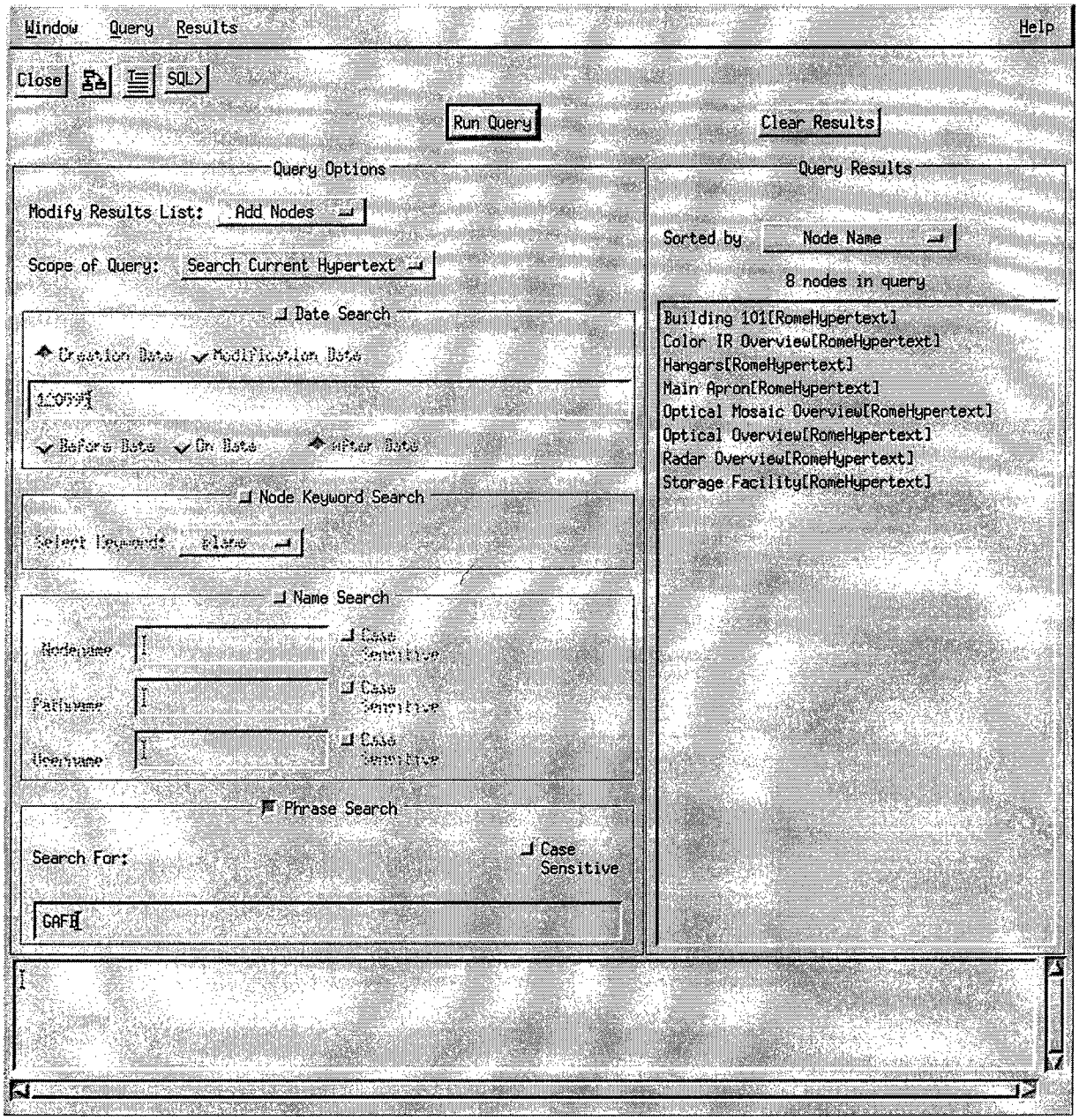
A Text View window displays the data contained within a selected node. It may contain graphics, audio (the loudspeaker icon), video, as well as stylized text. A red border around the text view indicates that a node is an anchor on a path. Red text indicates that text is an anchor on a path. Graphics, or portions of graphics, may also be anchors. The links between anchors provide the real benefit of the HyperTech software - relating information by association or by context to provide a large, cross-referenced body of information.

Path Browser



The Path Browser displays boxes which represent anchors that form paths. The title of each box is the name of the node containing the anchor. Inside the box is a text indication of what the anchor is (a piece of text, a whole node, or a graphic). Anchors may be dragged from text and structure views into the Path Browser to add an anchor to a path. Boxes in the structure view may be dragged to new positions to rearrange the path.

Database Query

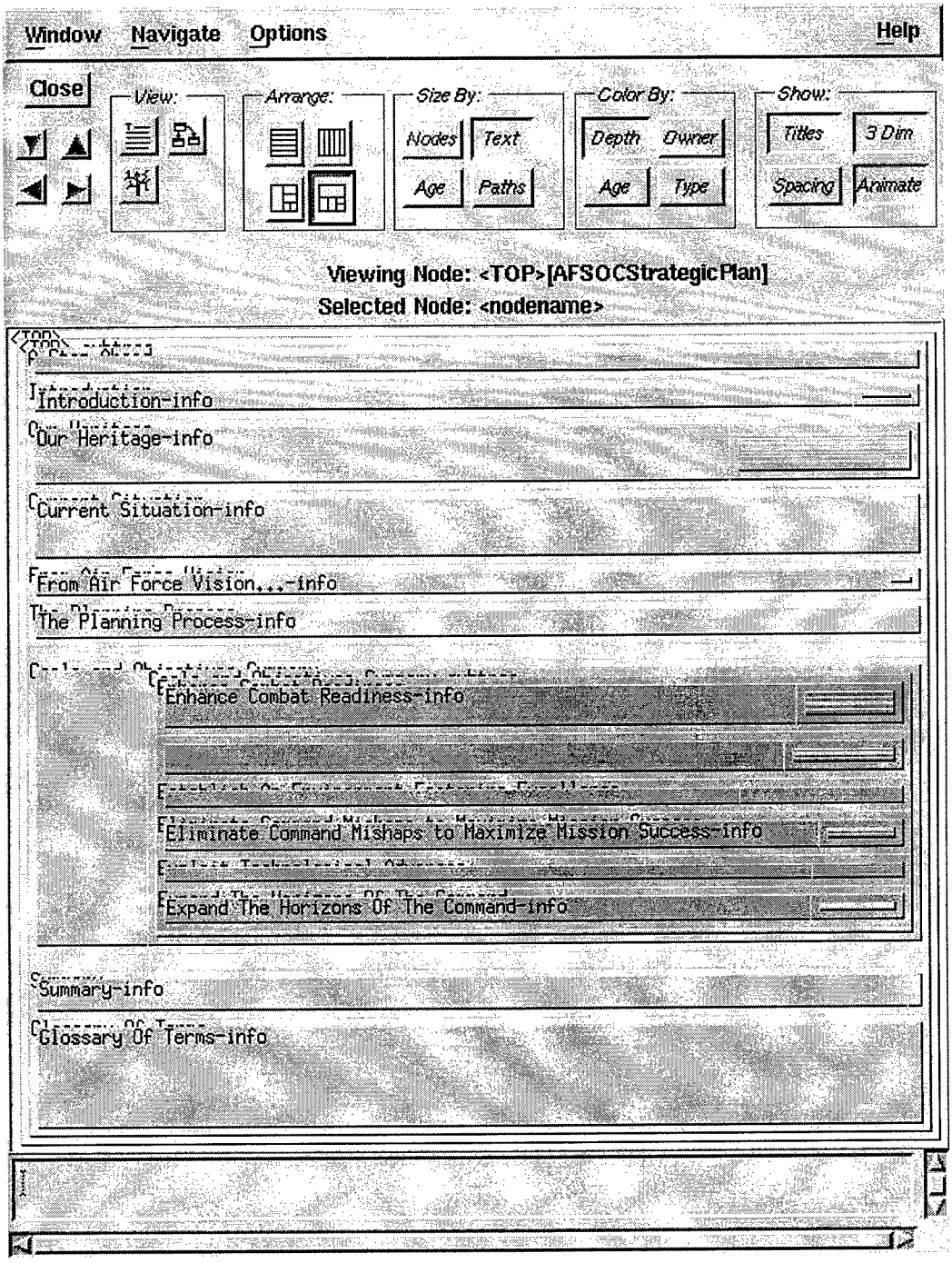


The following types of searches may be specified in the Database Query window:

Name searches - to find nodes with a particular name.
Phrase searches - to find nodes which contain a specific text phrase.
Date searches - to find nodes which were created or modified before, on, or after a given date.
Keyword searches - to find nodes which have been labeled with a given node keyword.

The user may specify multiple criteria within the same search. In addition, the user may enter a Structured Query Language command to perform searches that cannot be easily represented with the point-and-click operations available. Nodes resulting from queries may be automatically formed into a path for users to follow.

Treemap View



The Treemap View provides a spatial look at the hypertext documents within the database. Each box represents a node in the hypertext. The size of a node determines the size of its rectangle. Nodes represented by small rectangles contain less information than those represented by larger rectangles. A parent node with many children and grandchildren will appear as a large rectangle enclosing many smaller rectangles. Nodes can be represented based upon the amount of text contained in a node, number of children, node age, or number of paths which use a node.

1.1 Reference Documents

Documents pertinent to, or generated as a result of this project are listed below, along with a brief statement of their function.

- *Architectures for Volatile Hypertext*, Bernstein, Bolter et al., Hypertext 1991, New York: ACM, pp. 243-261. Article describing hypertext applicability to information management, written by one of the project's principal investigators.

- *System/Segment Design Document for the Hypermedia Interface Interim Prototype,* GTRI Project Number A-9575, Georgia Institute of Technology, March 31, 1994. Initial system design document for the program. Subsequent design changes necessitated major revisions to this document.

- *Software Test Plan for the Hypermedia Interface Interim Prototype*, GTRI Project Number A-9575, Georgia Institute of Technology, May 16, 1994. Document describing testing procedures to be used for developed software.

- *HyperTech Users Guide Version 2.0*, GTRI Project Number A-9575, Georgia Institute of Technology, March 30, 1996. Users Manual accompanying Version 2.0 of the HyperTech software, delivered 2 May 1996.

- *Software Design Document for the Hypermedia Interface Interim Prototype*, GTRI Project Number A-9575, Georgia Institute of Technology, est. August 1996. Final version of the V2.0 Hypertech System Design, documenting the software design as delivered in May 1996.

2. Detailed Discussion

2.1 Multi-user Support

Plans for HyperTech did not initially include support for concurrent access to a HyperTech database. Our first database design was geared towards a single user with complete control over a specific database, although we envisioned adding multi-user capabilities at some point in the future.

Per the original requirements, HyperTech 1.0 was delivered as a single-user system. Yet, as the scope of the application grew, it became increasingly clear that multiple users needed to be able to read data from a common database. Locking out HyperTech users who simply wished to browse a hypertext stored in the same database as a hypertext being modified seemed unnecessarily restrictive. With the addition of the Automatic Path Generation batch facility, which required that two users be able to access the same database simultaneously (the human user as well as the Auto-Path batch process), we had to review the database design to develop a solution supporting concurrent users.

The solution developed supports multiple readers and a maximum of a single writer in a specified database. We allow the user to commit or revert changes in a session at his or her discretion, just as with the single-user design. At the same time, we maintain data consistency and do not permit readers to access uncommitted data in use by an active writer.

This solution was complicated by Sybase's concurrency control mechanism, a locking scheme which ensures consistency of data being accessed by multiple users. While logically, HyperTech could restrict access to a certain hypertext at a time, the physical mapping of HyperTech objects onto records in Sybase database tables makes that restriction impossible to enforce due to Sybase's implementation of locks.

Specifically, the main objects in HyperTech are hypertexts, whose contents are stored in multiple records across multiple tables in the database. HyperTech assembles and disassembles the hypertext components from these records, and the user is never aware of the composition and decomposition happening on the fly. Unfortunately, Sybase's locking scheme does not provide the record level control we required, i.e., it does not allow us to restrict access to all records related to a given hypertext. Instead Sybase implements a page level lock or a table level lock when modifying data in a table. Operations which intend to modify a single record (such as the act of renaming a node) would, at a minimum, lock all the records on the data page containing that record. Thus the system restricts access to records which might have nothing to do with the hypertext being modified.

Several different work-arounds were attempted before we arrived at the approach implemented in HyperTech 2.0. The first attempt was to install a time-out mechanism, so that users would time-out when trying to access locked data. HyperTech would notify users when the time-out occurred, allowing them to retry or abandon the operation. This approach was not feasible due to general user interface concerns, e.g., a time-out due to server inaccessibility looked the same as a data lock conflict. Additionally, the need to wait for a time-out was very annoying to the user.

The next approach was to force record level locking by placing each record on its own page in the database. A page level lock would then lock only the single record, instead of other unrelated records that were coincidentally stored on the same page. However, we ran into a problem with this solution due to the need for indices. When updating data, if a column is not indexed, then the more sweeping table level lock is applied. Page level locks are only used when indices are in place. When updating indexed records, the indices must be updated, and to update an index, a lock on the page containing the index must be made. Since multiple indices are stored on a page, we still ran into conflicts that locked out entire pages instead of just records.

Since we were unable to "fool" the record-level locking mechanism, we decided to circumvent it entirely through the use of the "dirty-read" facility in Sybase SQL Server v10.1. A dirty read is the act of reading data regardless of any locks in place. But this meant a reader could potentially access uncommitted data in the database, data which a writer might later revert. Thus, we could not ensure data integrity, as a user could access the contents of a node as it was being deleted. Since hypertext objects are stored in multiple records, the deletion operation would take several steps to perform, and the user could end up seeing the text of a node that no longer existed. The requirement of data consistency would be violated by this solution.

We came to the conclusion that there was no way to outsmart the locking mechanism, and thus a database redesign was needed which would avoid lock conflicts as much as possible. The final solution was to implement database versioning, whereby each record is tagged with a version number. Readers examine HyperTech objects formed from the latest published version of the data, i.e., those records with the latest version less than or equal to the value in a special "version" table in the database. Writers create versions which are one greater than the value in this table, using a copy-on-write scheme. With copy-on-write, all modifications are handled by making a copy of the last published record, inserting modifications into this copy, and finally incrementing the database version on the copy. Publishing the version for future readers simply increments the value of the record in the special version table.

This solution has proven to be simple, clean, and relatively easy. Actual modifications to the database, in terms of inserting new records, deleting old records, or modifying existing records for a new version are all performed in short transactions, thereby keeping lock time to a minimum. While it would have been preferable to design for a multi-user environment from the very beginning, we are pleased with the solution developed mid-stream, as we think it is both solid and extensible.

2.2 Use of Xew Widget Set

Finding an existing Motif widget that could support the display of multimedia data and provide WYSIWYG editing features proved to be a challenge. Our goal was to use an existing widget, rather than develop one of our own. The widget needed to support styled text, in a variety of fonts and sizes, along with support for embedding graphics, video and audio data. It also needed to be "Motif-aware". The Motif Text Widget provides none of these features, so we initiated a search for public domain widgets that would provide these features.

After examining a number of possible candidate widgets, none of which satisfied all of the above requirements, we settled on the Text Editor Widget in the Eurobridge Widget Set, commonly

referred to as Xew. Xew contains a collection of widgets which were used as the basis for HyperTech's Text View. The only requirement it did not meet was integration with the Motif widget hierarchy. As part of the HyperTech project, we rectified this flaw by adding the necessary code to make the Xew widgets Motif-aware. We then submitted our modifications back to the maintainer of the Xew toolkit, where they have been integrated into the widget set since early 1995.

While the Xew Text Editor Widget suffers from a number of flaws (it does not scale up well to large documents, its cursor detection algorithm is flawed, its support for application-specified text tags is incomplete, and it lacks a search capability), it was our only option short of implementing a Text Editor widget ourselves. Such a task would have been a tremendous undertaking. The development of the Xew Widget set was itself part of a four-year research project, of which the implementation of the Xew Text Widget was a significant portion. Efforts to develop a simple Text Widget would have taken resources away from other aspects of HyperTech development. Accordingly, we feel that the use of the Xew Widget set was our best option at the time. However, we recommend that future follow-ons should again survey the available technology to see if newer and better widgets are available.

2.3 HyperTech Storage Format

HyperTech stores its text data in the native storage format used by the Xew Text Widget. In retrospect, it probably would have been better to store the data in the Hyper Text Markup Language (HTML) format. Direct support for HTML tags would have made it easier to import and export HyperTech data into the World Wide Web, functionality which was later added to the HyperTech design as an Engineering Change Proposal.

There are two reasons why this format was not chosen:

- the relative immaturity of the HTML format at the time of this project's inception (circa early 1994);
- deficiencies in the early versions of the Xew widget set.

When this effort began in 1993, none of us foresaw the tremendous explosion in popularity of the Web. HTML 2.0, the version of HTML most widely supported today, was just being developed. The first version of the popular Netscape Web browser had not yet been released when HyperTech development began. While HTML was becoming widely used, it was not yet the de facto standard found on the Net today.

In addition, early versions of the Xew widget set had very limited support for loading or extracting data outside of its own format, which is a subset of an ISO standard popular in Europe. It would have been difficult or impossible to parse and convert HTML formatted text to and from the special control sequences used by the Xew Text Editor widget for representing text styling and data embedding. The current version of Xew provides more control for application-specified tagged data, so some of these earlier restrictions are now removed.

In summary, given the popularity and availability of HTML formatted data, we would strongly recommend that future versions of HyperTech be re-coded to store text data directly in HTML for better integration with the World Wide Web.

2.4 Enhancement of Navigation Metaphor

Midway through the project, our internal experience with the link and path metaphor for navigating through hypertexts indicated that it was complex and difficult to use. It was necessary to redesign the navigation facilities to clarify and simplify traversal techniques in HyperTech.

The initial approach for HyperTech 1.0 allowed users to create links by specifying link sources and destinations in two separate steps. When both a source and destination had been set, a dialog would appear asking for the name of the link and the name of the path to which this link would belong. Two separate name spaces were provided, one for links and one for paths, leading to some confusion over which name was which. It was easy to lose track of where you were in the link creation process. In addition, the first development effort was not tasked to create any kind of facility for manipulating paths, so we arbitrarily chose to add new links at the end of a path, which was not always the most desirable course of action. Finally, the interface did not adequately identify when the user was manipulating links versus paths.

When tasked to add a Fisheye View to HyperTech 2.0 (a special view to show the contents of a given path) we employed this opportunity to revise the navigation design and chose to implement links as paths connecting exactly two nodes. Effectively, what had been a link in HyperTech 1.0 became the smallest size path. Instead of specifying link sources and link destinations, we added the concept of an anchor, a spot in a hypertext from which navigation may take place or where bookmarks may be placed. Anchors are created on spans of text or graphic. In addition, every node contains a special anchor which represents the entire node, regardless of the amount of text and non-text data it contains. Paths are now composed of a set of anchors traversed in a particular order.

We then enhanced the requirements for the Fisheye View, now known as the Path Browser, and combined it with a PathList Dialog, which listed all the available paths in the system. The result was a full-fledged path editing facility supporting drag-and-drop placement of anchors in paths. With the Path Browser, users may start navigating from any location on any path. Additionally, users have the option of navigating from any path passing through an anchor displayed in the Structure and Text Views.

While there was considerable work involved in retooling the user interface and data model to support this new metaphor, we believe it has been worthwhile in simplifying the navigational support within HyperTech.

2.5 Data Acquisition

In general we had some difficulty obtaining appropriate data to use for testing HyperTech. Multimodal data that were as similar as possible to the data used by military intelligence analysts was required, e.g., graphics and aerial or satellite imagery and textual analysis of that imagery.

We envisioned intelligence analysts using the system to examine images, make written analyses, and then link those analyses to portions of the images. We therefore needed a database in which texts were related by time or subject to the images so that we could exploit the path building facility of HyperTech. We obtained some imagery and a small amount of text associated with Griffiss Air Force Base and created a small HyperTech database with these files (the "rome" hypertext included with the release of the system). But this hypertext was too small and uncomplicated to provide a rigorous test of linking capability. Discussions with AFSOC (the intended user of the system) did not solve this problem, as we discovered that AFSOC does not generate its own databases. Rather, they primarily use data generated by others.

Since we were unable to obtain large amounts of data from the military community, we turned to publicly available materials. We eventually found data from a repository of weather information kept at the University of Illinois at Urbana-Champaign (UIUC). The "Weather Machine" site at UIUC archives various case studies of important weather events. In these archives we found the "Storm of the Century", a snowstorm that occurred in March of 1993. These case materials seemed to offer a fairly close parallel to the kinds of data in the military intelligence scenario. Included were satellite imagery of regions of the United States both in the visible and infrared portions of the spectrum; various graphics of weather conditions and forecasts; graphics of temperature, pressure, etc.; and a variety of texts. The texts included weather forecasts as well as newsgroup postings by weather professionals discussing the event. These texts could be linked to the various graphics and imagery. The weather graphics could also be linked to the imagery and vice versa. Most of the files carried dates and timestamps that also made it possible to create temporal paths through the data.

With the HyperTech 2.0 software we have included an example of these data and anchoring and path techniques in the "Simple Storm" database, a subset of the full "Storm of the Century" data. We anticipate continuing to use the full "Storm of the Century" database for testing of the Solaris HyperTech implementation.

2.6 Automatic Path Generation

Automatic generation of paths is one of the features that gives HyperTech an advantage over most other working hypertext systems. We have implemented the path generation algorithm described in earlier design documents and proposals — the goal being to link nodes according to textual similarities at the node and paragraph level. Path generation can function automatically in the background, or in a manual mode requiring user intervention. The algorithm used is similar to that of the Wide Area Information Search (WAIS) search engines now common on the Internet. HyperTech's version of the algorithm is based upon the work of Prof. Gerald Salton of Cornell University.

Thanks to good planning and design, the implementation of this algorithm was relatively uneventful. The biggest potential problem was solved by adding the multi-user support discussed in Section 2.1, enabling the path generator to run as a separate user of the Sybase database while the human user is simultaneously logged in. We have tested the implementation on a medium size text (a scholarly monograph) and on the Storm of the Century data. It is important to remember that the algorithm only works on text, not on graphics. Further testing will be necessary to determine what kind of data is most amenable to this path generation method. We anticipate the

system will work best on large amounts of textual data with a consistent and possibly specialized vocabulary.

2.7 User Interface Techniques

We faced some difficulties in designing a user interface that behaved as a standard Motif application while providing the specialized support needed for hypertext editing. Although the Xew Text Editor widget provides most of the needed features, its deficiencies still hamper the interface to some extent. For example, Xew provides very limited mechanisms to differentiate anchors in text. HyperTech also lacks a good facility for indicating when anchors overlap in a span of text; again, due to the lack of facilities associated with the underlying Text Editor Widget. Finally, better support is needed for lengthy operations such as data export by allowing interruption and/or moving these operations to the background.

Although the HyperTech 2.0 interface is sound, any future HyperTech prototypes should continue to refine the user interface towards making the hypertext concepts easier to understand and use.

2.8 Design of User Interface and Data Model

HyperTech was designed as a set of three software layers to facilitate future ports to different database engines and user interfaces. Those layers are:

- The Data Model Access Layer (database independent);
- The Data Model Implementation Layer (database dependent);
- The User Interface Layer.

The Data Model Access layer contains the Data Model Application Programmer's Interface (API). This layer is independent of the user interface used to display HyperTech data. Thus the user interface to HyperTech can be changed without modification to the Data Model API. All public data types are implemented as opaque types, whose implementation is known only to the Data Model Implementation layer.

The Data Model Implementation layer encapsulates the specifics of the relational database engine used to store and manipulate HyperTech data. It contains the implementation of the Data Model Access Layer API, and the details of the opaque data types visible in the API. A port of the Data Model to use a different database engine requires only a port of this implementation layer.

The User Interface layer makes calls to the Data Model API, but contains no direct dependency on any particular database implementation. The User Interface used to display HyperTech data may be changed without impact to the Data Model. Similarly, the Data Model can be modified without affecting the User Interface.

This approach has worked well and was proven effective in an early porting of HyperTech from a linked list database design to the full Sybase implementation. Any future user interface and database engine ports should be similarly successful due to the clear separation of modules and carefully designed interfaces between them.

2.9 Object-Oriented Software Development

This version of HyperTech was implemented in the C language as a conscious decision. Although the use of an object-oriented language such as C++ makes code implementation and reuse easier for the experienced developer, the initial learning curve is considerable. It takes some time to become fluent in the idioms and techniques of object-oriented development. Given HyperTech's schedule and resources, it made better sense to only employ such object-oriented techniques as could be easily implemented in conventional ANSI C.

Now, three years since the program began, object-oriented programming experience has become more common, and porting to an object-oriented language such as C++ or Java become feasible. Java is particularly attractive, not only for its emphasis on Web integration, but also for its reduced complexity and enhanced features compared to C++. Although HyperTech demonstrates that it is possible to develop an object-oriented system without direct support in the programming language, it is certainly more advantageous to have the support structure built-in.

2.10 Performance Issues

While the contractual requirements for HyperTech 2.0 functionality were fully met, the performance of the HyperTech application still needs improvement. Our experience with the current implementation has shown that hypertexts with many subtrees, but small numbers of siblings at a given level, perform better than shallow hierarchies with many siblings at a single level. The simple database caching scheme developed early in the project has improved performance tremendously, but tweaking the parameters of the cache should yield even better results. Some simple modifications in the Structure View should speed up the rendering of a wide hierarchy with many siblings. It is also a documented deficiency that the Xew Text Widget does not handle large amounts of text well, but if we modified the underlying implementation of the Text View, we would expect improvement in the text editing facility. Other performance bottlenecks can be detected by use of the *Quantify* performance analysis tool, and resolved through re-coding of the most CPU-intensive modules.

Some of these issues will be addressed while porting HyperTech to the Solaris environment.

2.11 User Feedback

With limited access to potential HyperTech users, we have made educated guesses as to their needs and attempted to address these needs with a powerful, yet easy-to-use interface which facilitates the creation and manipulation of hypertexts. But lacking a significant number of users, and in conjunction with our difficulties in obtaining realistic data, it is difficult to know if we are truly meeting the needs of the user. We hope to cultivate a community of users for HyperTech, whose feedback on the existing application will be invaluable in guiding future directions for the program.

3. Suggestions for Future Work

3.1 Enhanced Multi-user Support Features

The versioning approach used to implement multi-user support in HyperTech opens the door to a number of possible enhancements. With the addition of a simple protocol, we could add automatic notification of newly published versions of HyperTech databases to existing readers. This automatic notification could allow users the option of examining the latest version of HyperTech data when they become available. Automatic deletion of old versions (after the latest reader of the old version exits) could be incorporated into HyperTech. We could also examine the feasibility of reducing the scope of the write lock, from a databases perspective down to a hypertext or even a section of a hypertext.

3.2 User Interface Toolkit Alternatives

In the two years since choosing the Xew widget set as the basis for the multimodal Text View, more support for multimedia editing facilities has become available. In particular, the Tcl/Tk widget set has a number of widgets which support WYSIWYG editing of HTML. Future work on HyperTech should include another thorough search for public domain software that could become the basis for a robust and faster Text View.

3.3 Data Storage in HTML

As discussed in Section 2.3, we would recommend future versions of HyperTech store text data in HTML format. Such a modification would simplify the process of importing and exporting data from the World Wide Web.

3.4 User Interface Enhancements

For browsing purposes, we believe it would be useful to have a view which serves as a combination of the Structure View and the Text View. This view should show both the hierarchy contained within a node, along with the text of selected nodes within that hierarchy. Such information is currently available by launching a Text View from a selected node in the Structure View, but once the Text View is launched, it is decoupled from the Structure View from which it was spawned. The combination view would dynamically update its text display as different nodes are selected in the hierarchical display. This view would address concerns about the proliferation of windows launched within HyperTech.

As mentioned in Section 2.7, we would also recommend examining some aspects of the user interface based on feedback from the user community. A number of concerns in the Text View might be addressed by using a different widget as a basis for the Text View. Again, a reexamination of options for the Text View should be included in any future versions of HyperTech.

3.5 Porting to the Java Language

A port of HyperTech to the Java language would be worthwhile in at least three areas. First, a port to a true object-oriented language would facilitate the application of object-oriented techniques to HyperTech development. We could provide a true inheritance hierarchy for the views, and more easily and rigorously enforce the message passing schemes grafted into our C language implementation. Second, a Java-based HyperTech could work with popular Web browsers, such as Netscape, and provide access to HyperTech and its data in a widely available package. Integrating HyperTech with a Web browser makes HyperTech more accessible to the Web, and the Web more accessible to HyperTech. Finally, such an implementation would be platform-independent, allowing HyperTech browsing and editing on any MS Windows, Macintosh, or Unix system with a network connection to a database server.

3.6 Performance Enhancements

HyperTech experiences some performance problems as the amount of data in the hypertext(s) increases. However, efforts thus far have been primarily spent on enhancing the functionality of HyperTech, and little attention has been paid to deliberately speeding up the application. As a result, we believe there are some reasonably large performance payoffs possible with only a small amount of re-implementation.

In particular, we would recommend experimenting with some parameters of the database cache to reduce the number of hits to the database directly. We can be more selective regarding some of the calls made to the data model, and try to re-implement the algorithms to work with in-memory data, as opposed to constant retrieval from the database. We can also re-code some of the Structure View drawing routines to better handle large hierarchies.

While we intend to improve the speed of HyperTech 2.0 wherever possible during the course of the Solaris port, it would be a worthwhile effort for subsequent versions of HyperTech to exclusively focus effort on performance enhancements, through a thorough analysis of algorithms used and the pattern of database accesses.

4. Conclusion

Our experiences with the design and development of the HyperTech prototype have been encouraging. We feel that the user interface simplifies the understanding of hypertextual data and provides reasonable facilities for editing and traversing their components. The automatic path generation facilities are of tremendous value in reducing the burden of analyzing texts and creating connections between related concepts. The largest present drawback is in the area of performance on large hypertexts, and we intend to address this issue while porting the application to the Solaris platform. Nevertheless we believe that HyperTech currently provides what no other hypertext system does, namely:

- storage of hypertext data in a relational database which is manipulated in an object-oriented manner.
- support for an organizing hierarchy in a hypertext, in addition to traditional non-linear connections;
- import and export facilities to take data to and from HyperTech in a variety of formats;
- support for multi-user access to the same hypertext;
- an automatic path generation facility for automatically building connections within hypertexts.

There is much potential for expansion of the HyperTech application, particularly with respect to integration with the World Wide Web through direct support for HTML, and by re-coding HyperTech as a Java application that could be accessed through a Web browser. We believe that with greater access to more realistic data, such as used by the military community, HyperTech will fully demonstrate the advantages of using hypertext to organize and understand complex hierarchies of information.

☆U.S. GOVERNMENT PRINTING OFFICE: 1997/509-127/47090

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.