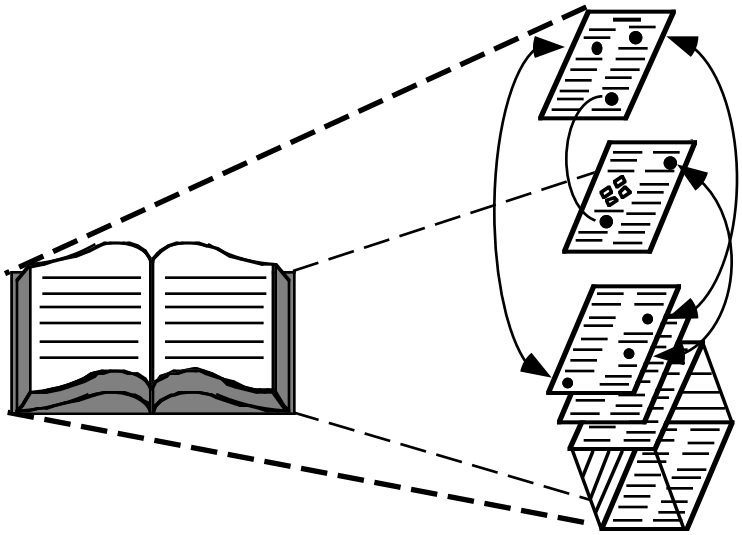
CS 1706 Spring 92
Project Specifications
HyperText Limited

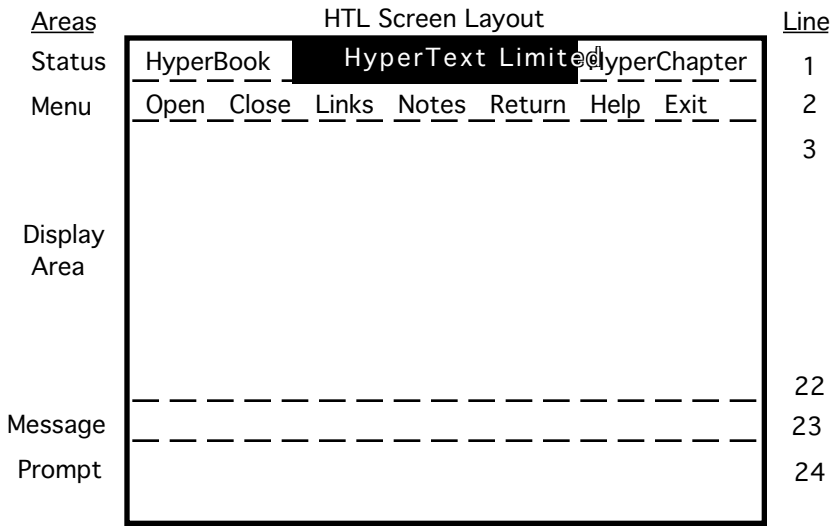


HyperText Limited, (HTL), is a system for restricted nonlinear reading. Normal text in books reads from left → right / top ↓ bottom on the pages. Hypertext provides readers with mechanisms for traversing the nonlinear organization, (graph structure), of a *HyperBook*. While complete Hypertext systems also provide hyper-authoring facilities for composing these HyperBooks, HTL will be limited to reading only. Hyper-writers using HTL will be required to use an existing editor to create their HyperBook which will be composed of a HyperDocument file describing the nonlinear organization of the book to HTL and a set of HyperChapter files. The HyperDocument file will contain the entire set of unique HyperLinks and their associated HyperChapters. The formats for the HyperDocument file and the HyperChapter files are given later. In addition to switching between the HyperChapter files, a pop-up "post-it like" note facility will be provided to enable an author to include brief messages to a reader as appropriate, (see notes command).

A session with HTL will consist of a user first opening a HyperBook, composed previously by a Hyper-Author. This will cause a default initial HyperChapter to be opened for reading. The user will scroll/page through the HyperChapter file while they are reading. At any point in the chapter the user may choose to switch to one of the hyper-linked chapters elsewhere in the book, usually at the suggestion of the author. The reader will be allowed to switch to a different HyperBook only after closing the one they are currently reading. An on-line minimal help system will be available to the user to explain the features of HTL.

Discussion

HTL will begin execution with a startup screen giving the usual information about the system and programmer, along with the current product version number. The startup screen must be presented in a clever manner in order to capture a viewer's attention. After 5-7 seconds has elapsed the startup screen will clear and the HTL screen will be displayed, (see HTL screen layout). The command line parameters must be checked to determine if the user has specified an initial HyperBook to open. If a HyperBook name follows the HTL program name on the operating system command line: *htl hyperbook*, this HyperBook, (HyperDocument file name), is input. A file existence check is required and an error message displayed if the file is not present in the current directory. The HyperLink list structure is then formed, at which time the user is instructed to wait. Once the list formation is complete, the default HyperChapter is opened and setup for reading. The HyperLinks for a particular chapter, which precedes the text in the file, is scanned and stored for later selection by the user. There is no limit placed on the length of a HyperChapter file, thus a dynamic data structure should be implemented, (i.e. the double linked-list ADT from MFV). HyperAuthors should note that a HyperLink is bi-directional, i.e. two-way. This implies that if readers wish they will be able to return to the immediate chapter from which they just linked. Thus HyperLinks automatically establish a two-way path between chapters. The remaining discussion will be made with regards to the HTL screen layout and how the system execution affects it.

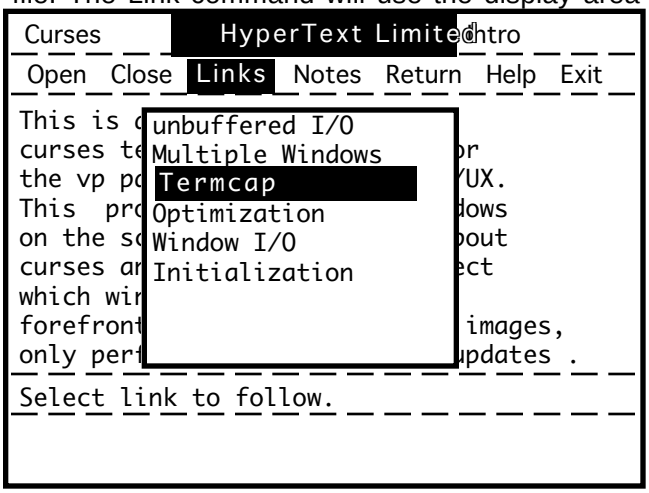


The screen will be divided up into several areas: the status area, menu area, display area, message area, and the prompt area. The contents of these areas may change, but they will be maintained on the screen at all times, after the startup screen has cleared and until the exit command is chosen. The status line will contain 3 fields: 1. the name of the current opened HyperBook, left justified; 2. the HTL system name, centered and inversely displayed; 3. the name of the current HyperChapter, right justified. The menu area displays the HTL operations, which are selected in one of two ways. The menu commands may simply be chosen by depressing the capitalized letter of the command name. Or alternatively the user may hit the tab key to activate an inverse video selection bar, (inverse color for Turbo users), highlighting the Open command. The bar can be moved horizontally using the ← → cursor keys to highlight the desired operation and hitting the space bar or return key to invoke it. The bar should wrap around at the ends of the menu line. In addition, if the backspace, (delete), key is hit then menu selection is exited with no command being chosen. After initial system startup, with no command line HyperBook specified or immediately after a Close operation, the only active commands are Open, Help and Exit. An attempt to activate an illegal command should cause a bell to sound and an appropriate error message to be displayed on the message line. When a HyperBook is open, the Open command shall be inactive as a user must first close the current book before attempting to open a new HyperBook. The display area will be used to present the HyperChapter file contents for reading, a HyperLink menu when the Links command is selected, (see Links command), a pop-up notes menu and pop-up note information, (see notes command). The message line shall hold brief messages, warnings and errors for the various operations. When an HTL action will take more than a couple of seconds to execute, a specific message should be displayed, followed by “Please Wait...”. The prompt area is used to request user input. All input/output from the prompt line, unless explicitly noted, will be buffered, i.e. READ & READLN. All other input, (command selection and scrolling actions), will be unbuffered.

The user may browse through a HyperChapter file by moving either a line or page up/down at a time. The ↑ ↓ cursor keys will allow the display area 20 line *window* into the chapter file to be scrolled one line backward/forward. If the user attempts to scroll above/beyond the top/bottom of the file an error message should be inversely displayed and the bell (control-h) sounded. The user may also scroll up/down a 'page' at a time by pressing 'U' or 'u' / 'D' or 'd' respectively, (Turbo users should use the PgUp and PgDn keys). HTL will also allow the top or bottom of the chapter file to be moved to directly, by the user depressing either the ‘T’ or ‘t’ / ‘B’ or ‘b’ keys, respectively. The display area should never contain blank lines for files with ≥20 lines. Consider a file with 100 lines, if currently lines 11 through 30 are displayed and page up is hit the first 20 lines in the file should now be on the screen. Similarly, if lines 71 through 90 are displayed and page down is hit, the last 20 lines (81..100) are then displayed. For files with <20 lines, page up/down would receive an error message and line up/down should not cause any line to scroll off the display. Speed of display should not be the over-riding factor for the implementation of the browse actions.

Command Execution

The Open operation prompts the user for the name of the HyperBook file (*HyperBook.htl*), which causes the same actions to occur, when a HyperBook file is specified from the command line, (see previous discussion). The Close command clears all associated data structures for the current open HyperBook and closes the "htl" file. The Link command will use the display area to put up a vertical menu of the associated HyperLinks for the current Hyper-Chapter, as shown at the right. The Link names will be displayed one per line, with an inversely displayed selection bar initially positioned over the first link. The user will move the bar vertically using the ↑ ↓ cursor keys, hitting the space bar or return key to choose a link for transfer. The selection bar must not be allowed to move onto a blank line. Any attempt by the user to do so must be prevented. The pop-up window shown at the right is not required, but strongly recommended. Alternatively, the display area may simply be cleared and only the links displayed. The user is additionally allowed the option of hitting the backspace, (delete), key in order to cancel the links command. In this instance, the links menu is cleared and the display area restored to its state immediately before the links command was activated. When a link is selected it causes the current HyperChapter to be pushed onto the chapter stack, and the HyperChapter associated with the HyperLink to be input for browsing. The notes command will display a list of the associated HyperNotes for the current HyperChapter. The desired HyperNote that a reader wishes to view will be selected analogously to the HyperLinks. No more than 20 HyperNotes will associated with any one HyperChapter. Once selected, the corresponding HyperNote text is shown in the display area 'on top' of the current HyperChapter contents. The reader may cause the note to 'disappear' by hitting the delete or backspace key, in order to go back to normal HyperChapter browsing. The Return command will cause the current HyperChapter to be removed from the display area and browsing of the previous HyperChapter to be resumed from the point at which it was suspended, (i.e. when it was pushed onto the HyperChapter stack). The Help command will use the display area lines to give a brief overview of the HTL system and short explanations of the menu line commands, along with accessing instructions. Optionally, but not required, a context sensitive help system may be implemented. The Exit command must prompt the user to be sure that they wish to halt the system, close the currently open HyperBook file, issue a completion message, clear the screen and return to the operating system.



File Formats

The format for the HyperDocument, (HyperBook), file is simply a series of pairs of HyperLinks and associated HyperChapters, one per line. The HyperLinks are unique strings in columns 1-40 and the HyperChapters are file names in columns 41-80. All of the HyperLinks in the HyperDocument must be present in this file. A violation of this restriction may result in a runtime error. Note that the same HyperChapter may be associated with more than one HyperLink. The HyperChapter file names must be fully qualified if stored in a separate directory, however it is recommended to HyperAuthors that all of the files for a HyperBook be stored in the current directory for simplicity. The HyperChapter file names must also contain the '.htl' extension required by the system. The first HyperChapter paired with the first HyperLink, specified on line one of the file, will be used for the default chapter to present to the user for initial viewing. It is these HyperLink/HyperChapter pairs that will be inserted into a double-linked list and searched when a reader chooses a HyperLink for viewing.

The format for the HyperChapter files consists of the number of HyperLinks, stored alone, on line one at the beginning of the file. Followed by the HyperLinks stored in columns 1-40, one per line. HyperAuthors should be warned that these HyperLink names must be an exact match to the HyperLink names specified in the HyperDocument file. The spelling, spacing and case must be duplicated, i.e. HyperLinks are NOT case sensitive or free format. No more than 20 links may be present in any HyperChapter. This will allow all of the HyperLinks for a HyperChapter to be listed in the display when the Links command is activated. Thus requiring no scrolling of the links vertical menu. Following the last HyperLink line, the next line will contain the integer number of HyperNotes for the file, limited to no more than 20. Each HyperNote will contain the unique name of the Hypernote in columns 1-40, followed by the integer number of lines of the note on the same line. The text lines of the HyperNote then follows. A HyperNote may contain no more than 15 lines of text limited to a length of no more than 60 characters. (This will allow a HyperNote to be shown in the display area without blocking the entire underlying HyperChapter text). The text for the HyperChapter follows the text of the last HyperNote in the file.

Text Windowing

The use of a text windowing package is not required for this program, but is strongly suggested. The ability to define multiple windows to refer to the same screen area will greatly aid the task of screen content management. The restoring of the display area's contents after listing the HyperLinks pop-up menu will be simple. Toggling between the display area HyperChapter text and the Help command information will also be quite easy. The text windowing facilities will greatly enhance the professional look-and-feel of your project.

A specification addendum for this project may be forthcoming shortly. The addendum, if needed, will describe further enhancements or changes to the HTL specifications.

PROJECT GRADING and DOCUMENTATION

Successful completion of this software development and implementation project will result in the following External and Internal Specification Documents for submission:

1. A complete **STRUCTURE CHART** of the system's design, including all interface specifications. The structure chart must be of sufficient detail to communicate to others WHAT functions are needed to implement this system and how the various parts are interconnected. The structure chart is complete if the algorithms for each module could be developed from it, along with a Data Dictionary, although no data dictionary is required. Use only 8 1/2 X 11 sheets of paper to record the chart, but DO NOT use larger sized pages. If you use > 1 page, STAPLE THEM TOGETHER. Use the symbols described in class for the components and the interfaces. Legibility counts and points will be deducted for sloppily presented work. Careful and CORRECT notation is also required and points will be deducted for a poorly annotated chart. The final chart will be compared with the initial chart to determine the amount of modification made to the original system design. Penalties will be incurred according to the amount of modification. Note - calls to PASCAL system routines: READ/WRITE need not be annotated on the chart.

2. A **FUNCTION/PROCEDURE LIST** must be developed. The list must include:

   a. name of each PASCAL function/procedure,

   b. a brief (1-2 line) description of its functionality,

   c. a description of its parameters,

   d. a listing of other procedures/functions called by it,

   e. a list of all procedures/functions that call it.

3. An **INTEGRATION PLAN**, that is a plan for HOW and WHEN the various phases of the project will be developed and which modules will be added at each integration point.

a. THREE (3) integration points must be fulfilled. The points must reflect a semi-equal division of the project work (i.e. Specifying only 1 or 2 modules to be completed at the first 2 points, while delegating the remaining, approximately 90% of the project for the last point is totally unacceptable. NO integration point may be planned for later than April 20th!

b. For each integration point, you must name the TASKS that will be completed by the identified date AND the PASCAL procedures/functions you expect to code in order to accomplish each task. That is, you must clearly define WHAT BEHAVIOR we can expect your program to exhibit at that point AND what Pascal procedures/functions you EXPECT to write in order to ensure this behavior. For each procedure/function you identify, you must describe BRIEFLY what role that procedure/function will play in accomplishing your stated objective.

c. The three integration points for the system are: 1. 03/26/92 2. 04/09/92 3. 04/20/92. Each student will be expected to demonstrate to their GTA the behavior of their project at one of the first two integration points. No student will be required to meet both integration points. There will be no labs on these dates, instead the labs will be used for demos. The GTAs will post a list at the start of the week of the students required to meet their integration point. Failure to meet with the GTA for the demo will result in point penalties.

d. At any integration point or when the program is submitted, any non functioning commands, actions, etc. of the system are not expected to bomb the system. A message should be displayed informing the user that this operation has not currently been implemented or is still under development. On the final submission, all non functioning aspects of the system MUST be accompanied by a short explanation describing the suspected problem(s).

e. A user manual generated by the word processor of your choice must be written. The manual should enable the NOVICE HTL user to access all the features of the system. This document should be at least 6-8 pages in length, excluding the title page, table of contents, index, etc.

QUALITY OF CODE

It is expected that the code will be WELL-DOCUMENTED, appropriately indented and VERY READABLE. Individual procedures/functions, excluding their documentation and declarations may be no longer than one page. Each compilation unit should contain only related procedures. Each procedure/function header should look similar to this model:

```
(*******************************************************)
(*PROCEDURE NAME                                       *)
(*                                                     *)
(*DESCRIPTION OF PROCEDURE FUNCTION (4-5 LINES)        *)
(*                                                     *)
(*DESCRIPTION OF ALGORITHM : FUNCTION IMPLEMENTATION   *)
(*                                                     *)
(*CALLED BY: (LIST OF PROCEDURES/FUNCTIONS)            *)
(*CALLS: (LIST OF PROCEDURES/FUNCTIONS)                *)
(*                                                     *)
(*PARAMETERS: NAME AND ROLE IN ALGORITHM OF EACH       *)
(*                                                     *)
(*AUTHOR: name of author                               *)
(*                                                     *)
(*REVISIONS:    DATE, REASON                          *)
(*              AUTHOR FOR EACH (if different)         *)
(*VERSION: x.xx                                        *)
(*******************************************************)
```

Points will be deducted for poorly presented code. This header need not be placed in routines that are used from previous programs (ex. screen control routines, etc.). Each of these documents must be placed in ONE binder or folder. Each separate document must be clearly labeled and separated from the others. The diskette containing the code must be inside and securely fastened. ALL WRITTEN MATERIAL MUST BE EASILY READABLE AND THE PAGES SHOULD BE EASY TO TURN IN THE BINDER. ILLEGIBLE WORK WILL NOT BE GRADED AND YOU WILL RECEIVE A "ZERO" FOR ANY SUCH PRODUCT SO REJECTED.

The following scale will be used for grading:

PRODUCT	PERCENTAGE OF PROJECT GRADE	PERCENTAGE OF FINAL GRADE
Structure Charts:	15	3.75
Procedure/Function List:	5	1.25
Integration Plan:	15	3.75
Code/Execution:	50	12.50
User's Manual:	15	3.75
	----	------
Total:	100%	25.00%

Due date schedule:

There will be no extensions or late submissions for this project! Any project or portion thereof NOT submitted on time will be rejected!

PRODUCT	DATE
Structure Chart:	03 - 17 - 92 (Initial Version)
Structure Chart:	04 - 20 - 92 (Final Version)
Integration Plan:	03 - 17 - 92
Procedure/Function List:	04 - 20 - 92

Code/Execution: ALL code is due by 5pm on 04 - 20 - 92. • D-Day •

User Document: 04 - 20 - 92.

NOTES RE: Submissions:

When you hand in your executable project, you should turn in for your lab instructor the following:

a. Hardcopy and softcopy LISTING and Make files, (required), for all code.
b. A diskette with all your source code AND a single executable image ALL in the root directory, NOT a subdirectory. The executable image must run by typing **htl**. The protection on all files should be set to allow access to any user (i.e. UNIX -rwxrwxrwx). Read/Write access to the data files (hyper???.htl, etc.) will be sufficient. Failure to conform with these standards will result in an immediate loss of 20 points. No "extra" files (other than the .htl files) should be on your diskette.
c. A hardcopy and softcopy of the user's manual. The softcopies (2) should be in the word processor format used to create the manual and also in standard ASCII format.
d. Modified final up-to-date Structure Chart.
e. Second copy of the Integration Plan + the original graded version.