AN INTELLIGENT DOCUMENT SYSTEM ON THE X-WINDOW SYSTEM

by

Channa Manjula Weeratunge, B.S.

THESIS

Presented to the Faculty of

The University of Houston-Clear Lake

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

THE UNIVERSITY OF HOUSTON-CLEAR LAKE
August 1994

AN INTELLIGENT DOCUMENT SYSTEM ON THE X-WINDOW SYSTEM

by

Channa Manjula Weeratunge

APPROVED BY THESIS COMMITTEE

Terry Feagin, Ph. D., Chair

Sharon Perkins, Ph. D., Committee Member

Kwok-Bun Yue, Ph. D., Committee Member

Theron D. Garcia, Ed. D., Associate Dean

R. C. Hopkins, Ph. D., Dean

Acknowledgments

I would like to thank my thesis chair, Dr. Terry Feagin, for his continuous support, feedback and encouragement in developing of this thesis. Dr. Feagin gave me the freedom of designing and implementing the tool, while providing his expert knowledge on the X-Window System.

I would like to thank my thesis committee members, Dr. Sharon Perkins and Dr. Kwok-Bun Yue, for their reviews, feedback and support.

Last but not least I would like to thank my parents for their unending encouragement and support of my endeavors.

ABSTRACT

AN INTELLIGENT DOCUMENT SYSTEM ON THE X-WINDOW SYSTEM

Channa Manjula Weeratunge, M.S.
The University of Houston-Clear Lake 1994

Thesis Chair : Dr. Terry Feagin

The X-Window System is an established graphical user interface standard for workstations that provides a uniform interface for different applications. It is heavily used in research, education and business environments. Many useful application programs have been developed for it. However, there are few, if any, tools in X-Windows capable of creating and organizing multimedia information into non-linear hypertext-like documents. The lack of such a tool is apparent when information needs to be organized or retrieved without much difficulty. To fill in this gap, HyperX, a tool similar to Apple's HyperCard was developed in the Unix/X-window System. Beside providing the necessary button linking features, HyperX also provides graphics, text, sound, movie and image capabilities for multimedia presentation. Using objects instead of bitmap in implementing graphical objects, HyperX allows efficient graphical operations such as re-sizing and editing.

Table of Contents

List of Figures

1.0 Introduction

People need to be able to create, store, modify and retrieve information for a variety of reasons. Examples include planning a trip, teaching students the latest in research or making a decision that can affect a nation. In order to perform these activities effectively and efficiently, the pertinent information should be easy to store and easy to access.

Organizing information so that it can be used in the most effective way can be a cumbersome task. It takes time and effort to find the information and then to organize it in a way that is comprehensible to others. To make this task easier, a method called "non-sequential writing" was proposed by Ted Nelson in 1965 [Nielsen, 1990]. The term "hypertext" has been used to refer to this form of information storage.

Hypertext is similar to a book, but much more flexible. A book gives references to certain facts or information that is not directly contained in it. Thus the interested reader needs to do a physical search for the referenced facts or books and then read the necessary information. Hypertext, in contrast, replaces a reference by a "link". A link is a transparent path connecting the current document and the referenced document. A link takes the reader to the correct position in the referenced document. Links can also help to break complex documents into well-organized sections and make it easier to retrieve the information on request. For example, a hypertext document could be a table of contents

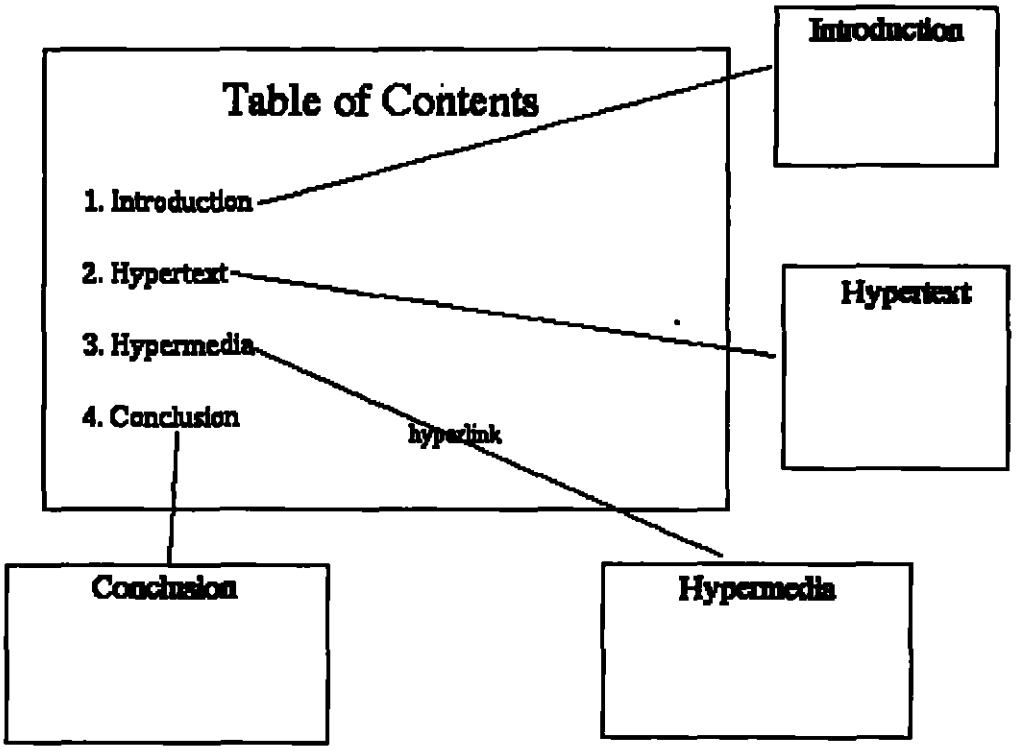
and each title could be linked to the appropriate document. Thus, the reader has the option of selecting the sections of interest and reading it without difficulty (Figure 1).

In order to create a hypertext document it is necessary to have all the documents in an electronic format that can be processed by the hypertext program.

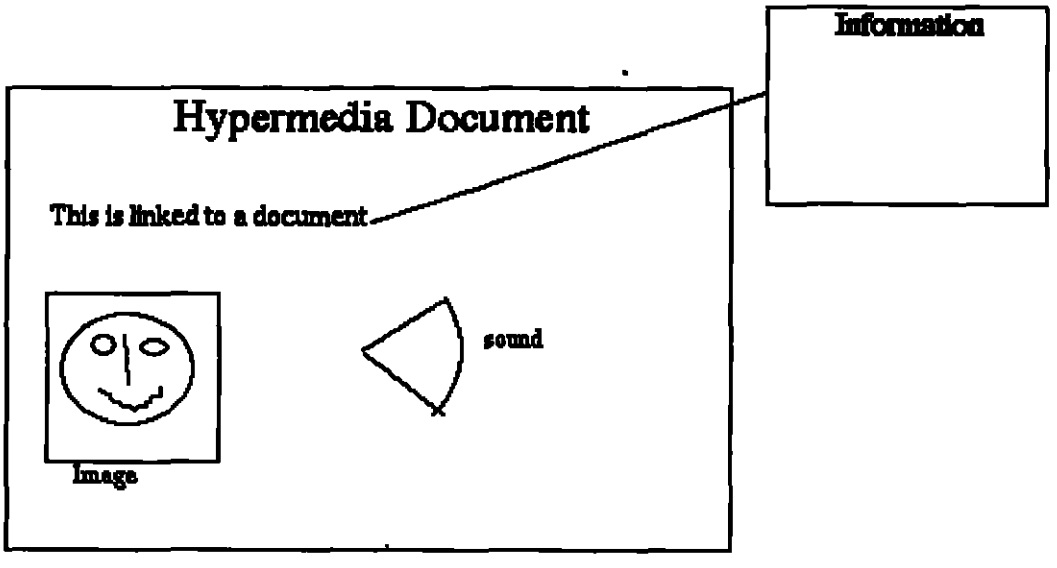
With the advent of multimedia, the term hypertext has been augmented and expanded in scope to include hypermedia [Nielsen, 1990]. Hypermedia includes not only text but graphics, audio, video and images. This brings a new meaning to a "document". Previously, a document had only text and every link brought up another document that also had only text. Hypermedia can include the additional capabilities to play sound, to show a movie or to decorate a document with images (Figure 2).

1.1 Intelligent document

Hypermedia documents are sometimes called intelligent documents. This is due to the fact that hypermedia documents are capable of arranging information in a user-friendly format and gives a user multimedia capabilities to comprehend information.



Hypertext Document
Figure 1



Hypermedia Document
Figure 2

An intelligent document is a set of related textual and graphical data objects which can be viewed, searched and queried interactively by the user in a non-sequential fashion. Individual data objects may contain links or pointers to other documents or data.

Intelligent documents can be very useful. They help to bring the distributed world of information to your finger tips. It also helps to organize information in an attractive fashion to provide many useful applications. For example, an intelligent document can be used as a presentation tool, a computer-aided learning tool, a personal organizer or as an electronic catalog.

Used as a presentation tool, intelligent documents provide many features. Its ability to provide information on request (by links) makes it ideal to organize information in a suitable format for a presentation. Furthermore, audio and video capabilities can be added to provide enhanced presentations.

Intelligent documents can also be used for developing personal organizers or calendars. Each day of the month can be linked to a document giving a to-do list. Furthermore, audio messages could be played as needed within the document.

As a computer-aided learning tool, intelligent documents show considerable promise. Their ability to incorporate text and audio-visual effects makes the learning process more pleasant than reading a book. Furthermore, since the information can be broken down

into documents and linked to each other, only the minimal amount of necessary information is presented at any one time. This allows the student to grasp information more quickly and also does not force viewing of information that is unrelated or unnecessary. For example, suppose an intelligent document had an image of a car and each of its major parts (engine, tires, etc.) are linked to intelligent documents giving a lengthy description of what that part does. If a student is interested in leaning about a car, he may view the image of the car and click on the part of the car that is of interest.

Intelligent documents can be used as a catalog. This feature might interest advertising agents due to the ease of presenting the goods. For example, if a company manufactures five types of paper clips, it could display the five images of the paper clips in an intelligent document and have each clip linked to an audio file giving its features. Thus the buyer can see the items and also get an explanation about it.

1.2 Contributions of this Thesis

Obviously, intelligent documents make the process of retrieving and viewing information more pleasant. A few tools exist to create and manipulate intelligent documents. HyperCard[1] software on Apple[2] Computers is one such tool. Although the UNIX[3]/X-

[1] HyperCard is a registered trademark of Apple Computer, Inc.
[2] Apple is a registered trademark of Apple Computer Inc.
[3] Unix is a registered trademark of AT&T

Window[4] environment is a popular platform [Nye, 1990], no tool to develop and manipulate intelligent documents has been developed so far.

This thesis concerns the development of an intelligent document system using the UNIX/X-Window System. The document system provides basic capabilities such as text, graphics, movies, images and sound. In addition, it provides a user friendly method of creating and using intelligent documents. The document system has a built-in editor to create intelligent documents and the capability for storing and retrieving the information. It also has the capability to traverse or browse through an intelligent document. The intelligent document system adheres to the style of the graphical user interface (GUI) of an X-Window toolkit.

1.3 Organization of the Thesis

The approach that has been taken to design the intelligent document system is discussed in Chapter 2. In Chapter 3, it is compared with similar intelligent document systems available commercially. The difficulties that were encountered in developing the intelligent document system are discussed in Chapter 4. In Chapter 5, future enhancements are discussed. Appendix A lists all the features and commands of the intelligent document system.

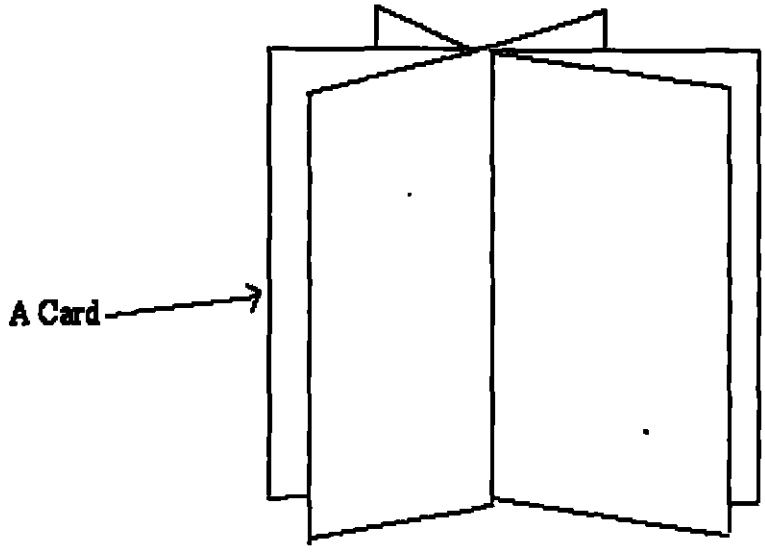
[4] X-Window System is a trademark of the Massachusetts Instate of Technology

2.0 Design of HyperX

The intelligent document system developed is named HyperX. HyperX has the capability of storing and retrieving information. In addition, it can create intelligent documents and display them on the X-Window System. It also provides a user-friendly interface for traversing through an intelligent document.

An intelligent document created by HyperX is called a stack (similar to a stack in HyperCard). This is due to the fact that the document is divided into blocks called cards and combined to form a stack or an intelligent document. Thus, a stack consists of one or more cards (Figure 3). The cards within a stack are arranged sequentially, but the stack can be traversed in any particular order. HyperX constructs intelligent documents as stacks for many reasons. First, since the document is subdivided only the important or necessary information is viewed at a time. Furthermore, documents can be created efficiently by using a special feature of cards called background and foreground, to be explained in the next paragraph. Modifying information is also a simple task since the information is broken down into cards. Finally, organizing information into an user-friendly format in HyperX is a simple task.

A card in HyperX can be viewed as a unit of information. A card has two layers of information named the foreground and the background. The foreground is



A HyperX Stack
Figure 3

Name : _____

Address : _____

Phone No.: _____

Background of a Card

Name : Mr. HyperX _____

Address : University of Houston--Clear Lake _____

Phone No.: (713) 283 – 3883 _____

A card with its forground superimposed on its back ground

Background and Foreground of a Card
Figure 4

superimposed on the background to form a card (Figure 4). Another angle of looking at the background and the foreground of a card is to view the background as a template for the foreground data. The background thus stays the same for many cards, while the foreground changes from one card to another. Both layers of a card can hold the same types of information.

HyperX cards can hold three basic types of information; graphics, text and buttons. Buttons are divided into two types. The first is simply called a button while the second type is called a sensitive button. The difference between buttons and sensitive buttons are that buttons have a three dimensional look and give feedback when activated, while sensitive buttons do not have a three dimensional look and do not give feedback when activated.

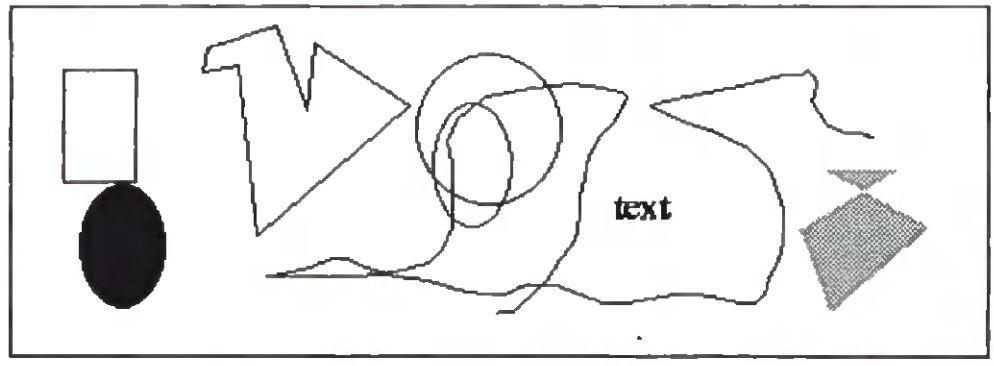
The graphical capabilities in HyperX allow drawings of basic shapes such as rectangles, circles, arcs, lines, polygons. It also has a free hand drawing capability. In addition, graphical text can be typed in different fonts. Though it does not have the capability of importing images from standard graphical formats as GIF , Tiff or Sun raster it can import an X11 bitmap image into the document. HyperX also gives the designer of the document a choice of colors to pick when drawing on a card. In addition, it gives a choice of brush sizes to draw on the card. The graphical editing features provided by the HyperX include a selection of a single object or a group of

objects. After a object or a group of objects are chosen it can be re-sized, moved or modified. It also has a delete/undelete facility and a cut/copy/paste option (Figure 5).

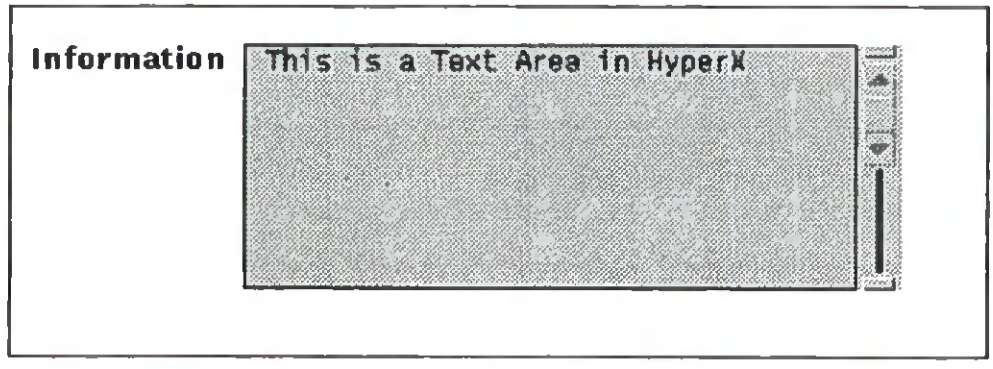
Text capabilities that are included into HyperX are the basic editing features of text, as well as the ability to enter text in a given color. Since the text editing feature is taken directly from a X toolkit, it has some limitations (Figure 6).

Buttons and sensitive buttons can be thought of as the most powerful feature of HyperX. Both the buttons and sensitive buttons have the same functionalities. The difference between the two are that, buttons have a three-dimensional look while its size is governed by the size of its label. The label of a button can be a string of characters or an X11 bitmap image. In addition, the button label can have any given color. Sensitive buttons do not have a three dimensional look and it can have any size. Furthermore, sensitive buttons do not have a label. Button and sensitive buttons are activated by using the mouse and clicking with the left mouse button on a button or on a sensitive button (Figure 7).

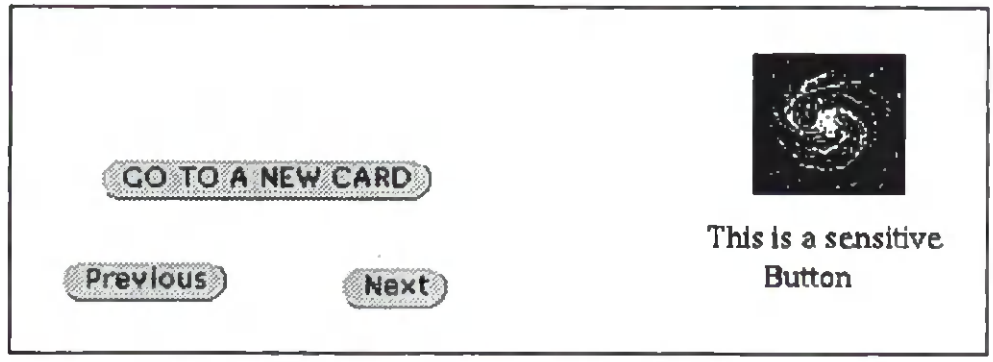
The functionalities provided by buttons and sensitive buttons range from no operations to complex linking of stacks to each other. The operation that is to be performed by a button or a sensitive button can be chosen from a menu. The first few choices duplicate some of the main menu selections. For example, "new card" which



Graphics
Figure 5



Teat Area
Figure 6



Buttons
Figure 7

creates and inserts a card into a stack at the point of creation, the options such as "previous card" and "next card" where the user can traverse the stack sequentially by flipping to the next card or to the previous card respectively. Some complex operations supported are the ability to play sound, movie or display an image. All the complex functionalities are provided through external views. For example, given an audio file that is attached to a button, HyperX play the sound stored in the file when the button is pressed. Buttons can hide data, and display it when a user clicks on a button or sensitive button.

As mentioned earlier, buttons and sensitive buttons have the capability of linking stacks and cards to each other. This feature is the most important in an intelligent document system, since it gives the user the ability of traversing the document in an nonlinear fashion. There can be any number of links (a path to traverse) in a stack. A link is a path to another card in the same stack or to another stack's card. Buttons and sensitive buttons hold the links and when activated, it takes the reader to the linked card in the same stack or in another stack.

3.0 Comparisons /Examples

The HyperCard software provided by Apple Computer, Inc. is very similar to HyperX. HyperCard was developed by Bill Atkinson and is provided as standard software with Apple Computers [Vaughan, 1991].

HyperCard organizes or views its documents as stacks and cards. A card is a basic unit of information and a stack is a named collection of cards. Cards in a stack usually hold the same related theme but it is not necessary.

A card in HyperCard is of fixed size, usually filling the entire screen (5" by 7" rectangle) [Vaughan, 1991]. Thus, its drawing area is limited. On larger Macintosh screens, HyperCard appears in a window. This limitation has been overcome by HyperX. It gives the user the ability to re-size the card to his or her preference by re-sizing the window it appears on. It should also be noted that re-sizing the window does not re-size or scale any data in the window automatically. Thus the user should re-size the window before designing a card.

In HyperCard, a card contains graphics, buttons and fields. These are analogous to the objects that can be found in HyperX. The term graphics is used in both HyperCard and HyperX. A button in HyperCard is analogous to buttons and sensitive buttons in HyperX. A field in HyperCard refers to a text area in HyperX.

3.1 Graphic Capabilities

HyperCard graphics is drawn on to a bitmap of fixed size. That is graphics is stored and drawn onto a two-dimensional array. The array contains a mark to indicate which pixels are activated to creates an image. HyperX stores the images as objects in a data structures and draws them onto the screen. For example, in HyperX a circle of radius r at position x, y is stored as a circle of radius r, at position x, y, while in HyperCard, it sets the positions of the pixel array that corresponds to a distance r from the point (x, y). Since objects are stored separately in memory, HyperX can provide the capability of adding color to an images without consuming lot of memory. It should also be noted in HyperCard, the memory consumed to store images are fixed, whether a single image was drawn or hundreds of images were drawn, but in contrast the grater the number of images stored the greater the memory requirements to store the images in HyperX.

HyperCard and HyperX provides the ability to draw basic shapes such as rectangles, circles, polygons, lines, text, etc. Both are also capable of importing images. HyperCard can import images from drawing programs like MacDraw. It stores imported image locally while HyperX imports X11 bitmap images from a bitmap editor, which is stored as a separate file. When the application is running the images needed for that stack should be available in the specified files. Editing features provided by HyperCard are little different from what are provided by HyperX. Since

an object cannot be retrieved after it is drawn, HyperCard provides a lasso to cut and paste areas of the screen [Beekman, 1991]. In HyperX, however, each individual graphic object can be cut and pasted or has its attributes changed.

3.2 Buttons

Button are the objects in HyperCard and HyperX that have the capability of initiating actions such as making a link to another card or stack, starting visual or sound effects, or displaying hidden information. Thus, buttons provide the essential part of making a HyperCard or HyperX documents intelligent.

In HyperCard, when a button is created, the designer has the option of selecting the style of the button. A HyperCard designer can pick from transparent, opaque, rectangle, shadow, round rect, check box and radio box [Beekman, 1991]. Furthermore, the button label can be set to display a name or an icon.

HyperX provides two types of buttons rather than providing many styles. They are called buttons and sensitive buttons. A button in HyperX has a three dimensional appearance and when activated, has an visual effect of being pushed in. In contrast, a sensitive button is an area selected by the designer of a card. When clicked with

the mouse, it does not give any feedback (no indication That the button has been activated) but performs the option that it is suppose to do. HyperX has the capability of giving a name or an icon to a button's label in a suitable color.

3.3 Text Areas

HyperCard calls the areas where text can be entered and edited fields, while HyperX calls them text areas. HyperCard provide many styles for its fields and has the ability to enter text in a number of different fonts. HyperX is limited in this respect. It has one style of text area and its font is fixed (This inadequacy can be overcame by using graphical text). It should also be noted that HyperX provides the ability to write in different colors.

3.4 User Levels

The structure of HyperCard is different from HyperX. HyperCard has been built on five user levels. The five users level, in order of increasing capabilities, are called browsing, editing, painting, authoring, and scripting [Apple, 1987]. In each of these modes, the user menu to work with change and the capabilities of what the designer

or user can do increase while going from browsing to scripting. HyperX does not have such user levels. It gives all the users full access to all of the functionalities. This could be dangerous if a user deletes or destroy an object by mistake. By providing user levels stacks can be protected, by giving different access levels to different users. This feature was not incorporated in to HyperX due to time restrictions.

To control these user levels and other functionalities, HyperCard relies on a special stack called the home. It has a card to act like a directory for HyperCard stacks, and also a card which is used to select user levels. Since HyperX does not deal with user levels, it does not have a home stack. However, a user may create a home stack if desired.

It should also be noted that HyperCard is very powerful. It has its own script language called HyperTalk[1] [Lu, 1992]. This feature allows the user to costomerize buttons, fields, card background and foreground and stacks themselves.

[1] HyperTalk is a trademark of Apple Computer Inc.

4.0 Difficulties/Limitations

Developing any software application, one comes across many difficulties. Some can be thought of been common to all applications, while others are application specific. For example, designing a data structure is a difficulty faced by all application developers. An application specific difficulty is not as easy to explain since it depends on the application.

Many difficulties were encountered developing HyperX. One of the major difficulties was designing the data and file structures. Another difficulty was implementing algorithms to store and retrieve data from a file.

HyperX was built on the X-Window system. Thus, the X-Window system governs the look and feel (GUI) of HyperX. In addition, the interface to the X-Window system is well defined, therefore the process of designing and coding was greatly simplified. Some objects like text and buttons are stored internally by the X-Window system. This feature also adds to the ease of programming.

Although the X-Window system simplified the task of developing HyperX, it added some limitations to the design and implementation. For example, there is less flexibility in designing the layout. Furthermore, since some items such as text and buttons were used for the

implementation of HyperX, it was difficult or impossible to customerize them according to the specification of the design.

4.1 Data and File Structures

Two types of structures were needed to implement HyperX. One to handle data internally and another to store data in a file. Both of these structures had be compatible such that data transfer from one structure to another could be done efficiently.

HyperX only handles data belonging to a card at any instance of time. Thus, a whole stack does not need to be in memory. Furthermore, since a card holds foreground and background data it is necessary to have a method of associating foreground data with the background data rather than duplicating the background data for each card. Internally, two distinct data structures were used to represent the foreground and background data. The file structure for storing foreground data in the file uses a pointer to locate its background data, which is also stored in the same file.

The data types to be stored in the structures vary from one object to another. To store text an array of characters is needed, whereas to store the vertices of a polygon a two dimensional array of integers is necessary. Polygons, texts, and all other objects have some basic

characteristics, such as the x and y coordinates, color, line width and so on. Thus, to incorporate different types to one structure efficiently, a union in C language was used, similar to the one show in Figure 8.

```
struct char_read{
        char name[ MAX_ARRAY];
        int another_block;
        };

struct vertices {
        int x_y[2][MAX_SIZE];
        int another_block;
        }
;
struct main_data{
        int color;
        int function;
        int line_width;
        int x,y;
        int type_of_drawing;
        int another_block;
        }

union data_type {
        struct char_read texts;
        struct  vertices  polygon;
        struct main_data  common_data;
        };
```

Figure 8 Data Structure I

To store a text or a polygon one union structure (Figure 8) is not enough. Thus, the field *another_block* is used to indicate that another block is associated with the current block. For example, to store a polygon, a *struct main_data* is used to hold is color, line width and other

information. Its *another_block* field will be set to indicate that a *struct vertices* is also associated with the current block. The vertices block will hold the information of the vertices. The last block to hold the information of the vertices will not have its *another_block* field set, indicating all the data for the polygon is stored. The data blocks are stored sequentially, therefore an indicator in the current block is enough (rather than a link) information to find out whether the next block is associated with the current block or not.

The union structure alone does not provide the complete details. This is due to the fact that there is no way of knowing what type of structure (*main_data, vertices, char_read*) is used to store data in the union. The union is embedded into another structure which contains a tag field specifying what type of structure the union holds as shown in Figure 9.

```
struct complete_record{
        enum type_stored name;
        union data_type data;
        };
```

Figure 9 Data Structure II

Finally, the structure is embedded into another structure for building link lists (Figure 10).

```
Struct link_list {
    struct complete_record objects;
    struct link_list *next;
     struct link_list *pre;
    };
```

Figure 10 Data Structure III

The above data structure forms a link list, which is used to store foreground and background data internally. For external storage of data, the structure in Figure 9 is added into a different structure (Figure 11). The reason behind this is that within a file, a linked list is implemented differently using file pointers rather than memory locations. In addition, the data stored in the file needs to have a pointer to the background data.

```
Struct file_data{
        int next;
        int pre;
        int background;
        int more ;
        struct complete_record objects[MAX_STORE];
        };
```

Figure 11 Data Structure IV

The structure in Figure 11 can be thought of as a block of data to be read or written to a file. This block stores the background information as well as the foreground information. When storing background information the *next*, *pre* and *background* pointers are not defined. When storing foreground information the *next* pointer points to the next card while the *pre* pointer points the previous card. The *background* points to the background of that card. The *more* field points to another block (Figure 11) if one block is not enough to store all the information.

4.2 Storing and Retrieving Algorithm

HyperX displays one card of a stack at any given instance. If the stack is traversed, a new card is inserted, opened a stack, or created a new stack, the currently displayed card is stored (if the data has been changed) in to a file and the new card is read and displayed (if such a card exists). The process of storing and retrieving cards from a file is time consuming. Thus, the reading and writing of information to and from a file should be handled efficiently.

The write algorithm writes data blocks (Figure 12). The first block of each card has valid pointers pointing to the next and previous cards of the stack. In addition, a pointer to the cards background data blocks exists. Another pointer within the block points to another data block, if one block is not sufficient to store all the data. If all the data cannot be stored in the second block, another block will be used. This process will continue until all the data are stored.

When a new card is stored, the blocks are placed sequentially starting from the end of the file. When a card that already exists is stored it starts storing the blocks from the original position. If space to store the card in the original position is not sufficient it stores additional blocks at the end of the file. If the card requires less blocks than it was allocated previously, a hole is created in the file. A hole is also created if a card is deleted. The write algorithm does not fill the holes, since the process of keeping track of holes makes the algorithm more complicated.

In order to remove the holes a compact algorithm can be used. The pseudocode for the write algorithm is shown below (Figure 12).

Input block - block or data to be written to the file

position - Place to write the block in file

output position - place for writing the next block

Algorithm

```
If position is new          /* a new block is to be stored from the eof maker*/
        go to end of file
        if not last block   /* if more blocks are to be stored in the file */
                set more field of block to file position EOF + sizeof block
        else
                set more field in block to end_of_block
        write block to file at eof
        return position is new;

/* a block read in from the file is written in the same position */
 read a data-block from position /*to find where the next block is in the file */
 if not last block /* if more blocks are to be stored in the file */
        if data-block more field is not end_of_block
                set more field of block and position to more field of data-block
        else
                set more field of block to end_of _block
                set position to new
 else
        set more field of block to end_of_block

 write block to file at data_block position
```

Figure 12 Write algorithm

The read algorithm (Figure 13) is much simpler than the write algorithm. Given the location of the card to be read, it reads the first card and any additional cards into the memory. The read algorithm checks to see if the background pointer of the first card is different from the background pointer of the card in memory. If so, it reads, the background information.

Input position - position of data block to be read

Output Block - data block read from the file

Algorithm

```
read the first block from position
if back ground of read block is different from what is in memory
        read all the background data from file
read the rest of the blocks
```

Figure 13 Read Algorithm

5.0 Improvements and Extensions

Improvements to an application are only limited by one's imagination. HyperX could be improved in many different ways. The graphical user interface, the algorithms and the functionality can be improved.

All commands in HyperX are mouse-oriented. To improve this, hot keys could be included into HyperX. This might seem as an added convenience to a keyboard-oriented person.

Though the algorithms used in HyperX perform adequately, they can be improved. For example, in repainting the window, the exposed areas could be repainted instead of repainting the entire window.

Another improvement in functionality is to add a script language to HyperX. Also, graphical capabilities can be improved by incorporating the ability to import images from any format. Text items need to be improved by giving it the ability to have different fonts and styles. Buttons can be improved by giving it many styles and some additional functions to do when activated.

Appendix A

Features and Commands of HyperX

Command	meaning
File Menu	
New Stack	Create a New Stack
Open Stack	Open an Existing Stack
Save a Copy	Copy the Current stack to a New Stack
Compact Stack	Remove Holes in a Stack (file)
Protect Stack	Password Protection for a Stack
Print Stack	Store the Screen in a Graphical format
Quit	Quit HyperX
Browse Menu	
Help	In Line Help on HyperX
First Card	Go to the First Card of a Stack
Previous Card	Go to the Previous card of a Stack
Next Card	Go to the Next Card of a Stack
Last Card	Go to the Last Card of a Stack

Edit Menu

New Card	Insert a New card into the Stack
Delete Card	Delete the Current card
Cut Card	Cut the Current Card
Past Card	Past the Current Card
Background	Display only the background of the Card
Foreground	Display the Foreground of the Card

Tool Menu

Browse	Change Mode to Browse
Button	Change Mode to Button
Text	Change Mode to Text
Sensitive	Change Mode to Sensitive Button
Graphics	Change Mode to Graphics

Object Menu

New Button	Commands to Create Buttons
New Sensitive	Commands to Create Sensitive Buttons
New Text	Commands to Create Text Areas

New Background **Create a new background for a Card**

Feature

Graphic Features

Shapes: *Rectangles, ellipse, Lines, Free hand, Arcs, Polygons, Import of bitmap Images, Text*

Options : *Color, Fonts, Line Width, Drawing Effects, Fill Object*

Editing : *Selecting One or Many Objects, Cut, Copy, Past, Delete, Flip Front or Back, Clear Screen, Unto delete, Modify drawn objects*

Button Features

Options : *Selecting of Operation, label or image, color*

Editing : *modify label, color and Operation, destroy, move*

Sensitive Button Features

Options : *Selecting of Operation*

Editing : *Modify Operation, Destroy, move*

Text Features

Options : *Label , Color, Number of lines and characters, Scrolling of text*

Editing : *Modify Label, Color, Number of Lines and Characters, Find, Edit*

Options of Button and Sensitive Buttons

Operation	Meaning
No Operation	Do Nothing
Next	Go to Next Card
Pre	Go to Previous Card

First	Go to First Card
Specific Card	Go to a Specific Card
Another Stack	Go to an other Stack
Multimedia	Play Sound, Movie or Display Image
New Stack	Create a new Card
More Detail	Reveal hidden Data

* Multimedia capabilities are handle through external viewers:

Movies are played through *mpeg_play* with file extension *.mpg*

Sound is played through *showaudio* with file extension *.au*

Images are viewed through *xv*

References

Apple, Computer, Inc. (1987), HyperCard User's Guide, Apple Computer Inc., Capertino, California, 22-23

Beekman, George (1991), Hypercard 2 in a Hurry, Wardsworth Publishing Company, Belmount, California, 96, 98

Lu, Cary (1992), The Apple Macintosh Book, Microsoft Press, Washington DC, 303, 309

Nielsen, Jakab (1991), Hypertext and Hypermedia, Academic Press Inc., Boston, New York, 5,33

Nye, Adrian (1990), Xlib Programming Manual, O'Reilly & Associates, Sebastopol, California, 1-50

Vaughan, Tay (1991), Using HyperCard from Home to HyperTalk, Que Corporation, Cormel, Indiana, 10, 30