Rochester Institute of Technology

A thesis Submitted to the faculty of
the College of Fine and Applied Arts in
Candidacy for the Degree of
MASTER OF FINE ARTS

"NAVIGATIONAL TOOL FOR THE GRAPHIC DESIGN ARCHIVE"
by
Ken Hayward

May
Nineteen Hundred Eighty Nine

Approvals

Advisor: Robert Keough
Date: 12-5-89

Associate Advisor: James Ver Hague
Date: 12-5-89

Associate Advisor: James Gray
Date: 11-22-89

Special Assistant to the Dean for
Graduate Affairs: Philip Bornarth

Dean, College of Fine and
Applied Arts: Robert H. Johnston
Date: 12/8/89

I, Ken Hayward hereby grant permission to the Wallace Memorial Library of RIT, to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.
Date: 12/5/89

Table of Contents

Acknowlegments

A sincere thank you must go to the thesis committee of Jim Gray, Bob Keough and Jim Ver Hague. Without their insight, guidance and patience, this project would not have been possible.

Special thanks to the famous Nancy Ryan for many words of encouragement and for being a pretty cool kid.

Proposal

The purpose of this thesis is to explore the possibilities of using animation to enable users to create a mental map of their location in a vast and complex sea of information. With computers' growing ability to give us large pools of information comes a need for us to use visual aids to help us feel comfortable while interfacing with these systems. This thesis will use animated sequences to help orient users as to where they are, where they have been (their path) and where they are able to go as they navigate through a system. These animations will be created on the Macintosh II computer and will be used as a behavior element associated with the Hypercard navigation buttons. The completed thesis will be in the form of Hypercard stacks containing a complex information tree and will feature animated sequences created with either Video Works II or a similar Macintosh software package.

Technical Notes

Hypercard[1] is the software package that was used to achieve the interactive qualities that are so important to this project. Hypercard uses it's own programming language called Hypertalk. The most useful feature that Hypercard has, for this application, is the ability to create "buttons" These buttons are not the traditional buttons that you would see on a control panel, rather they are "soft" buttons. Soft buttons are designated areas on a Macintosh screen that when acted upon (by clicking with a mouse for example) carry out commands.

Video Works II[2] is a software package that allows the creation of animation that is much more complex and powerful than the animation that can be created with Hypercard. Hypercard animation is very similar to the animation that children create by drawing little pictures in the corner of a book's pages and quickly flipping through the pages. Video Works II software on the other hand, was designed specifically to create animation. It allows you to use transparency effects to overlay images to achieve a dimensional quality. This overlay capability was a key to the visuals in my thesis.

Hypercard has a feature which allows you to access Videoworks II files and show them on the Macintosh screen while Hypercard is running. This feature is called the Videoworks II Driver[3]. The driver is run by two simple lines of Hypertalk code that need to know the name of the Videoworks II animation file to be accessed, the range of frame numbers in the file to be played and the location on the screen where they are to be played. In English, the code might say, "Go to animation file "X", get frames one through five and play them in the middle of the screen" (See script 1). This feature was valuable to my project be-

cause it enabled me to overcome Hypercard's shortcoming of black and white only graphics. It allowed me to use more sophisticated animation techniques than those which are possible with Hypercard.

The Videoworks II Driver takes a while to access the desired frames from memory because it must search through all of the Videoworks II files stored in memory to find the right one. This delay can be annoying to the user. I had different Videoworks II files for rotations and graphics when I first began. Later I discovered that it is possible to pre-load one Videoworks II file at a time into Hypercard's memory for instant access. I strung every Videoworks II file that I had together to make one huge file and pre-loaded it into memory. This eliminated the delay in retrieving frames and made the interaction run more smoothly.

In order to prevent confusion, it is important to make a distinction between graphic buttons and Hypercard buttons. Graphic buttons are simply Video Works II pictures shown on the screen that designate a set of boundaries in which a mouse click will have some result. The graphic buttons do not have the power to do this by themselves. Hypercard buttons do have the power to carry out orders when a mouse is clicked within their boundaries. Although the user can not see them, one or more invisible Hypercard buttons are placed behind every graphic button. It is the placement of a graphic over the Hypercard button that appears to give the graphic buttons power.

The Thesis

After deciding that the topic of my thesis would be the development of an interactive navigational tool, the first task that I addressed was writing a list of exactly what I would and would not do for my project. The proposal that I had written was broad enough that it would have been easy to go off on tangents, so it was very important that I had a precise list of things that this project would include. Some things that I would have liked to work on were eliminated because of time constraints. For instance, icons would be very helpful for quick recognition of buttons, but it would have been a thesis in itself to design icons for the 72 different buttons in the dialogue. By making myself a more pointed statement, I was able to focus on a strict list of things to do without worrying about being sidetracked.

The Purpose of This Tool:

The function of this navigational tool is to provide a front end for the Graphic Design Archive. The Graphic Design Archive is in the form of a Hypercard stack currently containing upwards of 900 data cards. A user who wants to see the cards containing examples of paperback book covers does not want to sort through each of the 900 cards to find them. It would be much easier to go to a nice little pile of cards that have the attributes that you are looking for. The Graphic Design Archive has been broken down into a taxonomy that categorizes the cards into groups. This tool is intended to present the taxonomy in a manner that is more visually concrete and therefore more intuitive for users.

This tool allows you to enter the Graphic Design Archive cards at a specific point. Once inside, the user is free to move about the cards using the card's own means of navigation. The cards are all equipped with a button that has the word "navigation" on it. When pushed this button will take them back to the front end navigational tool.

This navigational tool is designed to be a generic tool in that it is not

built specifically for the Graphic Design Archive. It was built as a means for navigation through any large database and the Graphic Design Archive was simply the database that was applied to it. In keeping with being generic, information about the Graphic Design Archive was not prioritized in any way. If it were, categories of the most importance or categories with the greatest amount of information in them would have been placed in positions on the visual representational model with the most visual impact or given a proportionally larger space allotment than less important categories. I did not feel qualified to make judgements about the information in the Graphic Design Archive so information was alloted space in a random fashion.

Early Ideas:

At the time when I conceived of the idea for this project, I had a few vague ideas of concepts that I wanted to use. To keep track of these ideas, I made the equivalent of a sketch pad in Hypercard. This sketch book consisted of many rough Hypercard animation prototypes that illustrated the ideas that I was developing. These concepts behind these little demos became the basis of the whole thesis project.

Laying the Groundwork-
Visual Metaphor vs. No Visual Metaphor:

Before I actually began to work on this project I decided to plan exactly what course that I wanted to take in terms of the look and feel of the interactive dialogue. One of the most sensitive decisions that had to be made was whether to use a visual metaphor or not. Visual metaphors are graphic treatments used in interactive media that give the user information about how the system works. A good illustration of the use of a visual metaphor can be found on Macintosh computers

with the use of file folder icons. Immediately upon recognition of the icon even the first time user will summon all that they know about real file folders and apply it to the icon. They know that when they open a file folder in the real world they see all of the files that are inside of it. Intuitively they know that if they open a file in the Macintosh world, they will have access to all of the files inside of it. A large part of the battle of having people understand how to interface with a system can be won if, by simply recognizing a file folder icon, a user can assume "This must work like a filing system"

A visual metaphor, when used well can be a valuable framework around which to organize the presentation of visual material. Problems can, however, arise when the metaphor is taken literally. Using the file metaphor for an example again, one can see that, while the metaphor works well in revealing simple things like the ability to open a file and see what is inside of it, it becomes a constraint when the system needs to do things that are inconsistent with the metaphor. Say Macintosh wants to allow a person to open files that are buried three layers deep without having to open the two layers above it first. The whole metaphor is based on what we know about real file folders and we know that with real file folders you cannot open a file that is three layers deep without first opening the files above it first. Allowing these files to do things that real files can not do violates and in some cases invalidates the metaphor for the user. At that point the metaphor becomes an anchor around the dialogue designers neck because it is not flexible enough to support new options.

In my opinion there is a trade-off with any visual metaphor. Using a metaphor is great because it helps inexperienced users get a quick sense for how the system works and it is a nice way to organize a simple interaction. The trade-off is that if the system is going to grow and get more powerful and complex, the metaphor will invariably break down. At some point the dialogue designers will have to decide whether to not push the dialogue any further because the metaphor will not support it, or to push further and eliminate the metaphor.

For my project I decided to sacrifice the benefit of a visual metaphor for the flexibility of a system where I made all of the rules. I decided to design an object that would visually represent the body of information in the Graphic Design Archive. This object would be a symbol for the database and it would have a set of physical properties that pertained to it only. The trade off for not using a metaphor was that there would be a learning curve that users would need to digest these properties and understand how to work with them. To shorten this learning curve I would have to develop of a visual treatment that would be somewhat intuitive to users. For the graphic treatment, I wanted to create a visual model that would represent the Graphic Design Archive taxonomy and would illustrate the three layers of depth that it had.

Progressive vs Full Disclosure:

Another early concern was whether to make all of the textual information fully disclosed at the outset or disclosed progressively as you move through the Graphic Design Archive taxonomy. Progressive disclosure gives you only the top layer of information at the beginning and dispenses more as the user needs it. This is particularly useful to the inexperienced user because it allows them concentrate on the job at hand without being distracted by peripheral information that is not needed yet. Full disclosure better suits power users who want all of the information available so that it can be processed in a less structured fashion. Power users do not want to go through A and B to get to C, preferring rather to jump directly to C.

The solution to the progressive versus full disclosure debate that I came up with was a compromise of sorts. I wanted to be flexible enough to lead first time users through the taxonomy step by step while allowing the experienced users to have the power to skip layers. Information would be chunked to a certain degree on the representational model because not all of the model would be visable at once. Howev-

er, once an area of the model is chosen there has to be a means for giving the user information, whether it be all at once or a little at a time. I decided to fully disclose the information within an area in a manner that gave some feeling of order to the different layers. This would entail making the top layer the focus of attention for inexperienced users benefit while still having the bottom layer accessable to the experienced users.

Creating the Visual Representational Model:

In creating a visual representation of the Graphic Design Archive, I was faced with two main concerns, screen space and database size. Since I knew that I would be working on a Macintosh computer using Hypercard software, I was aware that there were going to be severe space constraints because the Hypercard window on the screen only measures approximately 5 1/2 x 7 inches. The portion of the Graphic Design Archive taxonomy that I was using had 51 different end points. This told me that I had to be very efficient with the space that was available. I considered using both organic and geometric forms to represent the Graphic Design Archive taxonomy but chose geometric shapes because their modular qualities allow for more efficient use of space. The most space efficient shapes for text placement, I discovered, were squares and rectangles and the least efficient were circles and triangles. I spent a great deal of time sketching various shapes and configurations of shapes and finally settled on a 3-D geodesic sphere. This sphere which is made up of pentagons and hexagon is similar in appearance to a soccer ball. The hexagonal panels on the sphere would each be a separate button and all of the separate buttons would form together as one whole unit that the user could rotate. This unit, the sphere is extendable in that it can be sub-divided to include thousands of panels like the famous dome at Disney's Epcot Center.

Problems Creating the Visual Representational Model:

At this point, the need arose to put together a quick prototype of the 3-D sphere to see if it was a feasable solution. First, I learned to use the software for Pro 3-D[4] and with it, I created a sphere. In constructing a sphere, Pro 3-D automatically uses hexagonal panels. When viewing the sphere straight on, the center hexagonal panel looked perfect and the panels surrounding it were skewed to look convincingly like they were in perspective (See figure 1).

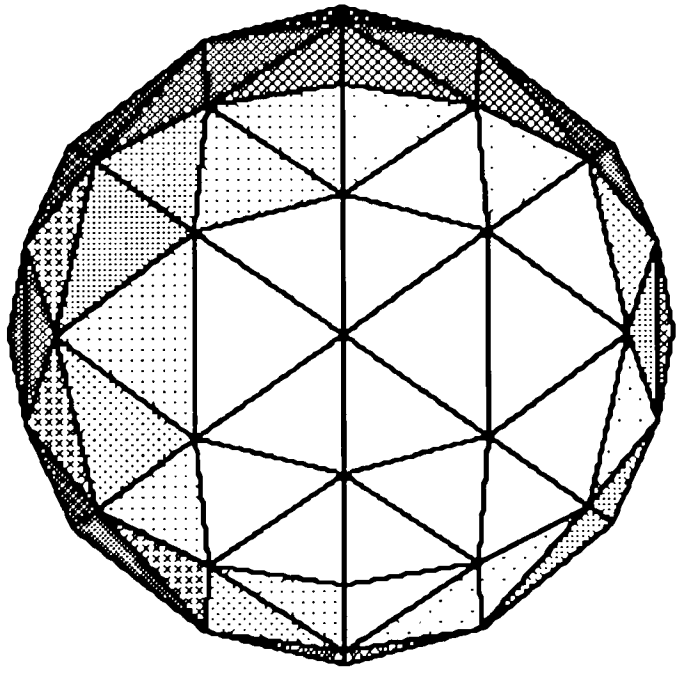

Figure 1

However, when rotated one quarter turn the center panel on the sphere was no longer a hexagon, it was a square. In addition, the surrounding hexagonal panels were badly skewed but not in true perspective, leaving me with unacceptable results (See figure 2).

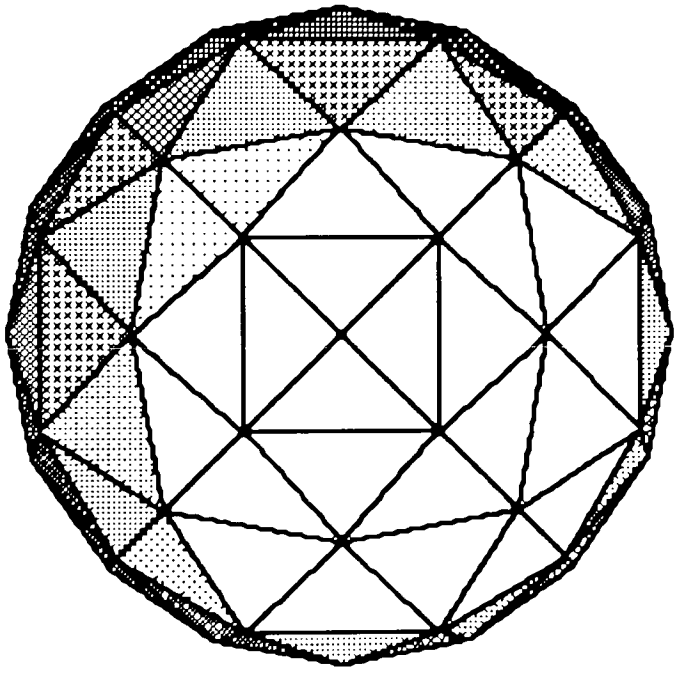

Figure 2

This distortion is due to the problems with displaying a 3-D object in 2-D space. Looking straight on, the sphere appears to be a uniform and symetrical object shown with the proper perspective. When the object is rotated 90 degrees, it becomes apparent that the object is not uniform at all, it is created in such a way as to give the illusion of perspective when viewed only straight on.

The next attempt at a solution was through the use of another software package, Swivel 3-D[5] This package, which supports color, works in much the same way as Pro 3-D, except that rather than using hexagonal panels, it used verticle and horizontal circles, like lines of latitude and longitude, to construct its sphere. This left me with a sphere made up of paralellograms rather than hexagons (See figure 3)

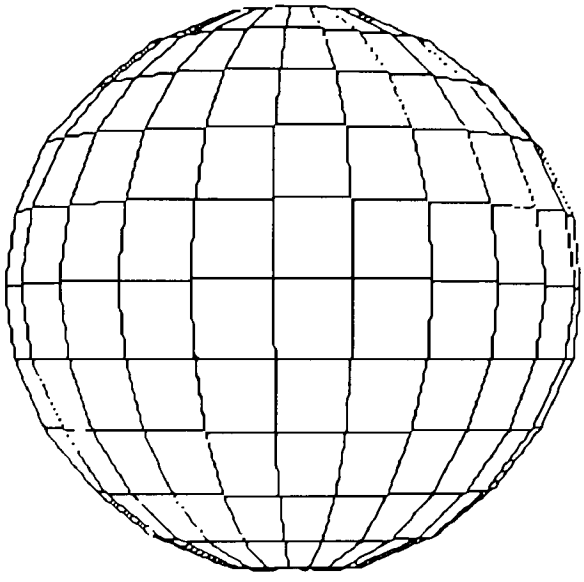

Figure 3

This was an interesting possibility since squares and rectangles are the most space efficient for the placement of type. The problem was that when rotated vertically, the top and the bottom of the sphere (where the north and south poles would be on a globe) were made up of triangles (See figure 4).

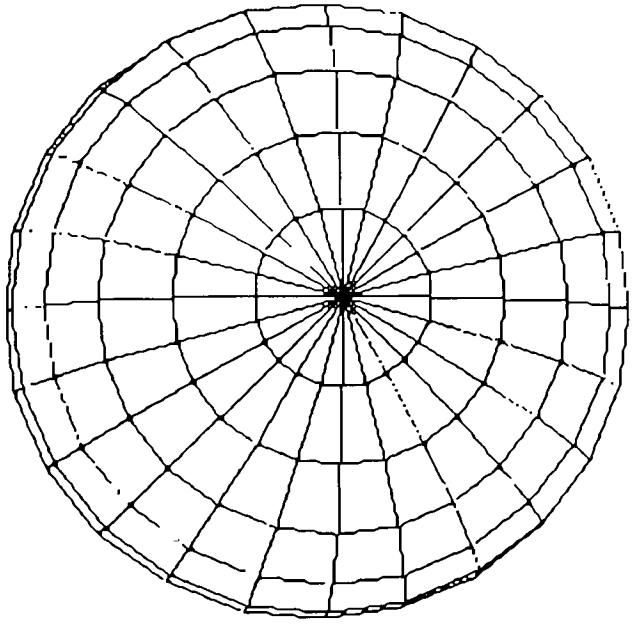

Figure 4

This inconsistency of shape left me with another unacceptable prototype.

There were no more 3-D software packages at my disposal, so I had no alternative other than to develop a new geometrical form. This form was made up of cubes arranged to resemble a 3-D plus sign (See figures 5 & 6).

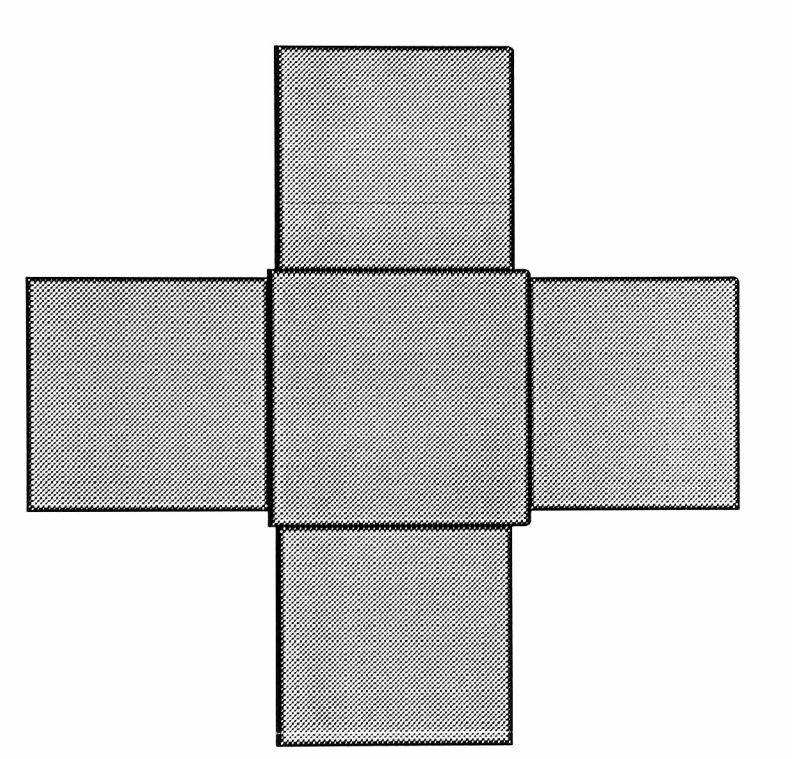

Figure 5

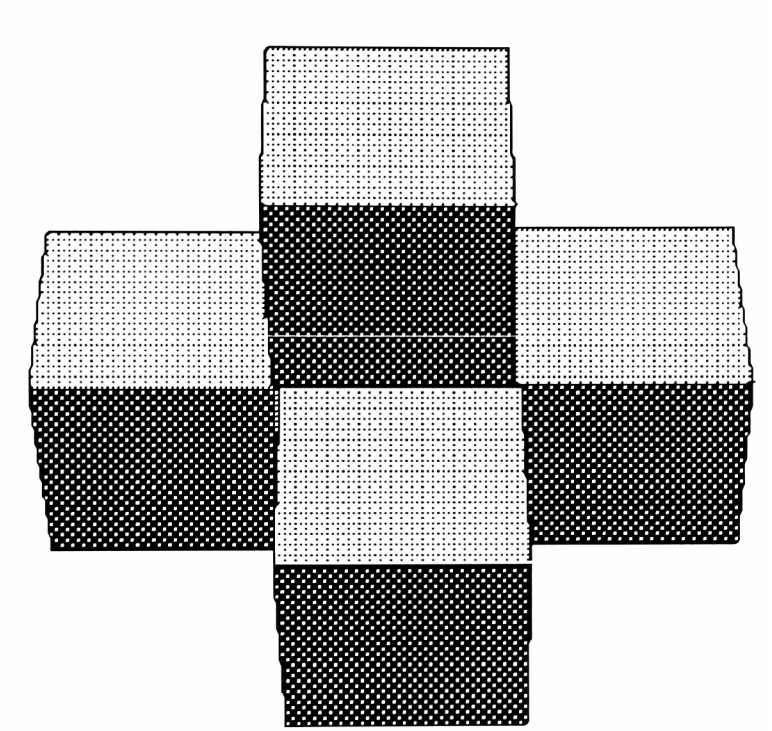

Figure 6

This object had all of the desirable properties that the sphere had. It was modular, geometrical, 3-D, rotatable and extendable. I used Swivel 3-D to construct the form, which was made up of six separate cubes linked together.

Creating The Animated Rotation:

As most people know, to create an animation you need a series of pictures of an object shown at different positions and played quickly to give the illusion of motion. Swivel 3-D helped a great deal in creation the images that I needed to animate the rotation of my object. After the object was created all used a function called " 'tweening". To " 'tween" you first need to designate a start position for your object. Next you have to move or rotate your object and designate a stop position. Then you specify the number of frames you want between the start and the stop frame and the computer will move the object from the start to the stop frame in the specified number of frames.

Swivel 3-D has a built in light source that adds to the feeling of depth. The light source casts a shadow on the object that gives an irregular appearance when the object is rotated. To remedy this problem, I had to take each individual frame out of Swivel 3-D and paste it into a color paint package called Pixel Paint[6] and touch it up.

From Pixel Paint each frame had to be cut and individually pasted into a color animation software package called Video Works II. This package enables you to play back your frames at whatever speed you want. For my object to rotate vertically and horizontally, I needed to create only the frames for a 90 degree rotation to the right and a 90 degree rotation down in Swivel 3-D. With four of these rotations of 90 degrees my object looked as though it made a 360 degree rotation. The upward and left rotations were completed by showing the downward and the right rotation in reverse sequence.

Developing the Navigation Tool

The visual representation of the Graphic Design Archive as we know is a 3-D object that when not rotating, has only one face visible at a time. The object needs to be rotated in order to see what is on the other faces. To begin the interactivity it was necessary to create a means for manipulating the rotation of the object. First a navigational symbol needed to be created that would tell the first time users that the object was rotatable and to act as a tool to allow them to actually rotate it. After many thumbnail sketches I decided on a symbol that had four arrows radiating from a center origin and pointing up, down, left and right, the directions in which the object rotate (See figure 7).

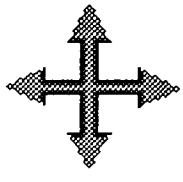

Figure 7

The navigational symbol was created with the Hypercard paint tools and it was placed directly on every card in the Hypercard stack. This enables the user to rotate the representational object at any time.

Now that the navigational symbol was created, it needed to be brought to life and given the power to actually manipulate the representational object. This power was achieved by applying Hypercard buttons to the symbol. I created four invisible Hypercard buttons slightly larger in size than the arrowheads of the navigational symbol and placed them over the arrowheads on the symbol (See figure 8).

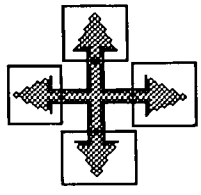

Figure 8

The code for the button on the upward arrow said in essence, "When this button is pushed, show ten Video Works II frames of the object rotating upwards. The other three arrow buttons had appropriate script for their indicated direction of rotation.

Metaphorically, Hypercard is a stack of index cards grouped into different relations. Each card can carry out commands by itself. One type of command that was particularly useful in this project was the "open card" command. This command says "When you come to this card, do this..." Since the taxonomy had five faces worth of information, I created five different cards, one for each face. Using the "open card" command, the code said, "When you come to this card, show the Videoworks II frame that has a picture of this face" (See script 2).

By combining the rotational power of the navigational symbol and the open card command the illusion of a 3-D object in full rotation was achieved. The rotational frames are generic- that is the same ten frames of rotation are shown every time the "rotate right" button is pushed. What makes it look different is that after the ten frames are shown there is a command in the button script that sends you to the next card. There the "open card" commands takes over and shows the appropriate frame for that side of the representational object. When the "rotate right" button is pushed on card one, you see ten frames of rotation quickly followed by the image of face two. If you push the "rotate right" button again you see the same ten frames of rotation followed by the image of face three and so on. Since the visual representational model has four sides (not counting the top and the bottom),

when you push the "rotate right" button on side four you are taken to side one again to complete the 360 degree rotation.

The verticle rotation works in the same way as the horizintal rotation with a few exceptions. The representational model has six faces but the taxonomy has only five faces worth of information. Rather than having five full sides on the model and one blank one, I decided to eliminate the empty side from the verticle rotation. This means that every time that you rotate either up or down from faces one, two, three or four, you are taken to face five. The effect is like that of flipping a coin over. It does not matter whether you rotate 180 degrees upwards or downwards, you still end up on the opposite side. From side five everytime you rotate vertically you return to the face that you came from. This is achieved by using the "push" and the "pop" card commands. To "push" a card is to put it into memory and to "pop" a card is to retrieve it from memory. The code for the "rotate up" button on cards one, two, three and four say, "When this button is clicked push this card (remember it), show ten frames of upward rotation and go to card five (See script 3)" The "rotate up" button on card five says, "When this button is clicked, show ten frames of upward rotation and pop the card (retrieve it from memory)" (See script 4).

The visual representational model created with Swivel 3-D is simply a skeleton with no data attached to it. Swivel 3-D does not support the use of text, so another solution needed to be found for applying textual information to the visual representational model. At this stage the necessary Swivel 3-D frames for full verticle and horizontal rotation were stored as Videoworks II files. What was needed were the dividing lines for the different buttons on each face and the corresponding text describing the button. I used Pixel Paint to create the text and the appropriate dividing lines for each button. These graphics were in essence a Pixel Paint overlay that was copied out of Pixel Paint and pasted into the Videoworks II files of the representational model. The ruling lines and the text from Pixel Paint was layed over a side of the representational model in much the same way as a transparent piece of acetate with press type on it would lay over a photo for a magazine cover paste

up (See figure 9).

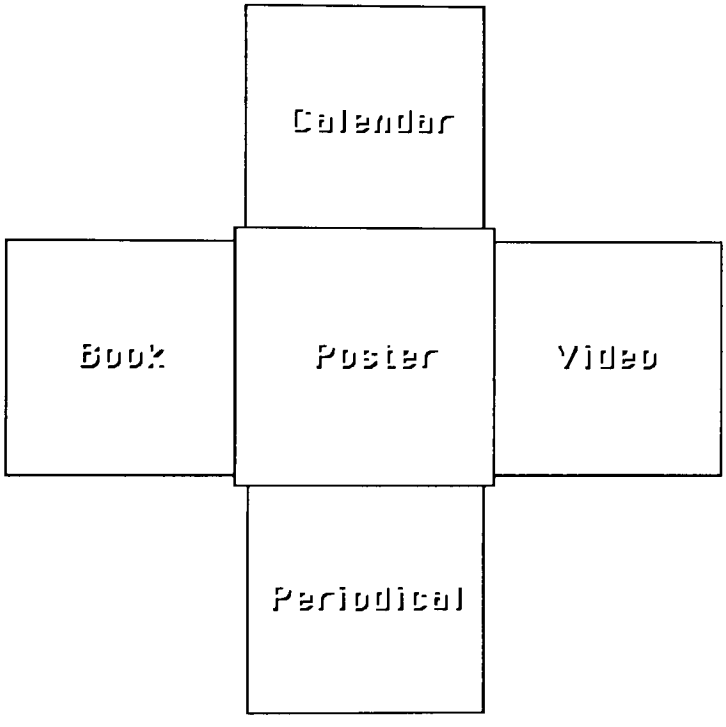

Figure 9

Magnifying Buttons While Browsing:

In an effort to save space on the tiny Hypercard window, I developed an animation strategy that enlarged the area on the representational model that the user was looking at, making the information on it more legible. It is physically impossible to cram all of the information that I had onto the Hypercard window at once and still have it all be a comfortable type size. Instead, I fit all of the information on the screen in an uncomfortable type size and created an animation that enlarged one button at a time like a magnifying glass might (See figure 10).

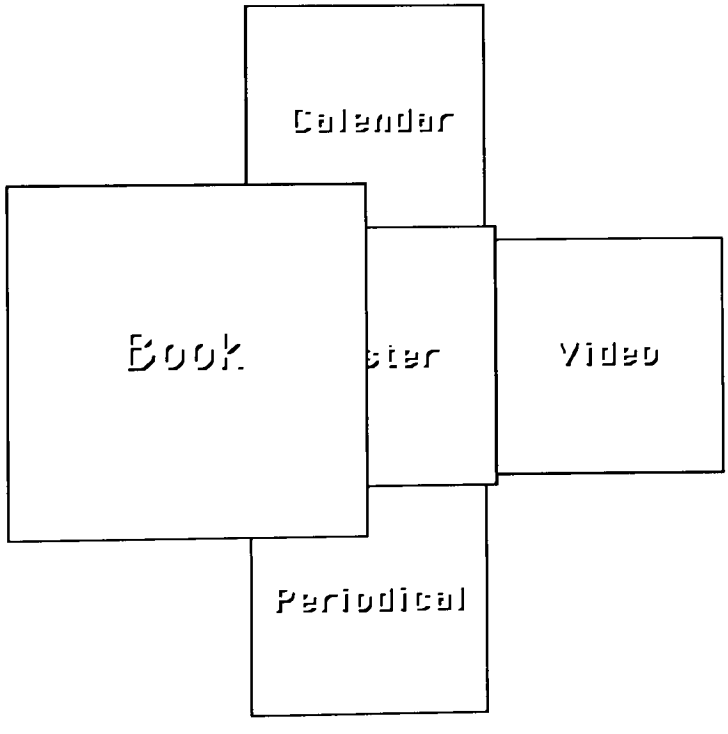

Figure 10

I made the assumption that wherever the cursor moved on the screen, the user's eyes would be moving. Based on that assumtion, I decided to enlarge whatever area on the representational model that the cursor was in.

To do this a transparent Hypercard button was placed over the boundary of each of the top layer button graphics on the representational model. These Hypercard buttons had an "on mouse enter" command. These commands are carried out every time the mouse enters the boundaries of that Hypercard button. The code for these Hypercard buttons said basically, "If the cursor enters my boundary, show the Videoworks II frame with a picture of me enlarged" (See script 5). This meant that I needed to create new Videoworks II frames showing each of the button graphics in the whole taxonomy enlarged. These graphics were created with Pixel Paint. Once the user had enlarged the button graphics while browsing it was necessary to eliminate the old Hypercard button that did the enlarging and replace it with new Hypercard buttons that had the power move through the taxonomy. This was done by modifying the Hypertalk script to say, "If the cursor enters my boundary, hide this Hypercard button and show the layer one Hypercard buttons. Then show the Videoworks II frame with a picture of me enlarged" (See script 6). Hiding and showing Hypercard buttons simply makes the buttons usable or unusable.

A "mouse leave" command was used to stop more than one button from being enlarged at a time. Whenever the cursor exited a button's boundaries a script told the button graphics to shrink to their original state and to have the current layer's Hypercard buttons hidden (See script 7).

The action of enlarging the button that you are looking at enables the user to have the power of browsing through the areas on the representational model without risking making a decision by pushing the button. The pressure of making a choice seems to create a great deal of anxiety, especially with inexperienced users. It seems refreshing to be able to get informaton without the fear of pushing a button.

The fear of interacting with a computer comes in part from people's fear of being in unfamilliar surroundings or worse yet, lost. The real goal of this project was to create an enviroment in which users felt at home and were not apprehensive about moving around and possibly getting lost. A fault of many interactive systems is that they change the users environment or world when a decision like a button push is made. By changing their world, I mean that users are hurled into an atmosphere that has a different graphic look. Often times, important buttons have a new look and/or location which compounds the problem. The time when the user is adjusting to their new surroundings is somewhat distressing because they are forced to take a moment to get their bearings straight and to decide whether they are in the right place. Inexperienced users seem to equate making choices with the unpleasent experience of landing themselves in some unknown surroundings. To avoid this distressing situation, I wanted to keep the user in one environment at all times. The reinforcement of staying within a stable world seems to make people more willing to explore without chancing the consequences of getting lost.

Developing a Graphic Treatment

Finding a suitable graphic treatment for each side of the representational model was a tricky thing to do. In the early stages of planning it was decided to have all of the information on each face accessable at all times. This meant in some cases showing three layers worth of buttons on the screen at once, hopefully in a non-chaotic fashion. To visually order these buttons into a hierarchy I used the graphic design principals of proximity, color and size manipulation. The representational model itself is a medium intensity blue. I wanted the top level button graphics to be the most visually prominent so that they would jump out at the first time users indicating that it was the most important thing to look at. To do this I used a 14 point bold typeface in a high intensity yellow to contrast with the object. To punch up the text even more, I added a dark blue drop shadow. That same blue was used to outline the perimeter of the buttons and to help reinforce what the text went with this button. The text was placed in the center of each button to demand maximum visual attention.

The second layer button graphics needed to be visually strong enough to be easily readable without competing with the layer one graphics. For both the text and the dividing lines I selected a subdued shade of blue lighter in chroma but more subdued in value than the blue of the object. For the text I used a 12 point normal weight typeface with no drop shadow. The text was placed in the extreme corners of it's buttons as far away from the layer one text as possible.

The goal for the layer three graphics was to be legible to experienced users who knew what they were looking for, but visually less prominent than the rest of the graphics. For this I used a very muddy shade of red that was very close in value to the blue of the background. This color was used for both the text and the dividing lines. The text was a nine point light typeface that was easy to miss if you did not know what you were looking for.

These three layers of graphics were then combined like transparen-

cies over a face of the object. The first layer that was placed over the object was the layer three graphics, followed by the layer two and finally the layer one. This layering effect helped to accentuate the visual hierarchy because in some cases, layers of graphics actually layed on top of other layers clearly showing the ordering from top to bottom (See figure 11).

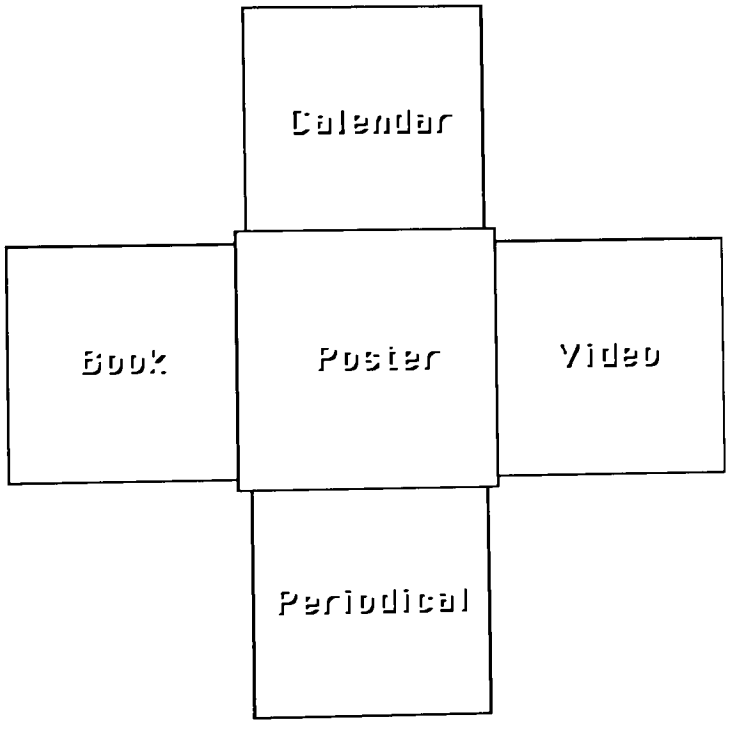

Figure 11

As the user moves along from layer one to layer two the layer one transparency is visually peeled off (See figure 12).

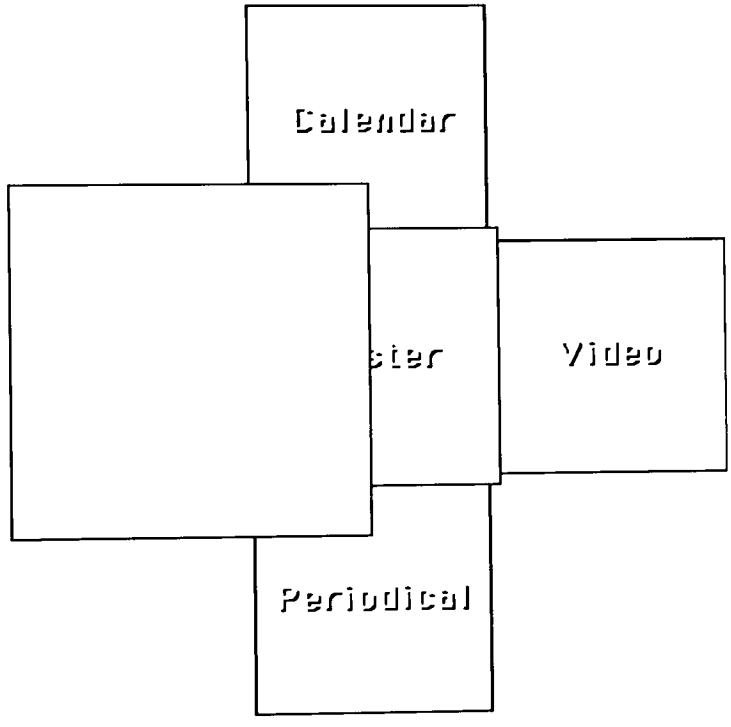

Figure 12

The graphics for layer two then become the most important and need to be adjusted. The intensity of the blue text and lines are given a boost in intensity and the text is changed to a bold typeface.

The same idea carries over when moving from layer two to layer three. In this case the layer two graphics are peeled off and the layer three graphics are punched up in lineweight, typesize and color intensity to be perfectly legible (See figure 13).

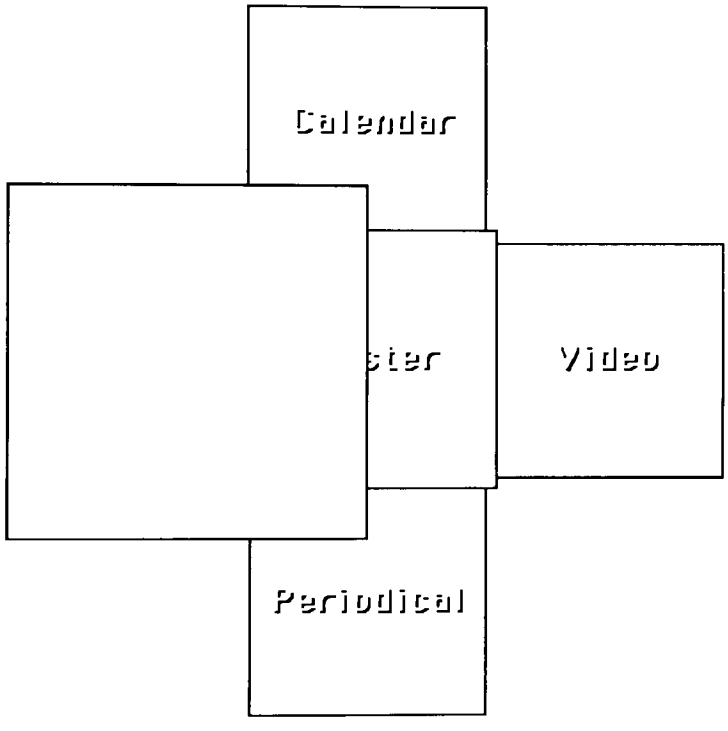

Figure 13

From the layer three buttons, the user is taken directly to the Graphic Design Archive cards (See figure 14).

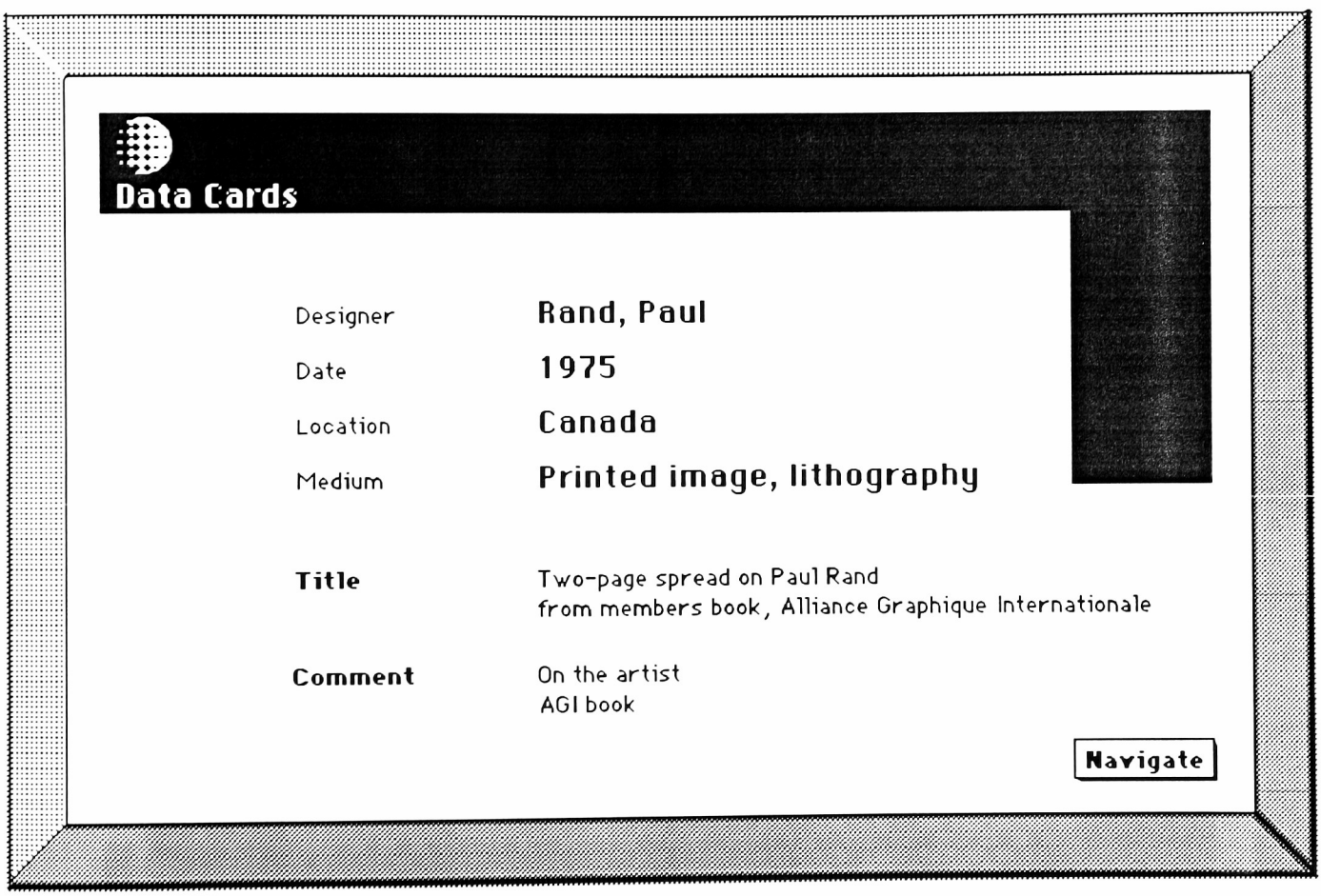

Figure 14

Creating Layered Buttons:

The challenge of keeping the user's environment stable involved some very detailed Hypertalk programming. To do this it was necessesary to construct a layered network of buttons on each card. It was this layering that enabled a first time user to navigate one layer at a time while allowing an experienced user to skip layers. The key factor in determining how many layers would be jumped was the number of times the mouse was clicked. One click advanced the user one layer, two clicks moved them two layers etc.

The first thing that was needed was a means for keeping track of the number of times the mouse was clicked within a period of time. I chose to keep this period of time to one second to differentiate between first time and experienced users. I felt that in this short period of time a user must be deliberate to register multiple clicks. The users who are unsure of themselves would probably make one click. If they accidentally did make two clicks, it would be unlikely that they would come within a one second span.

The device that I developed to keep track of the number of mouse clicks is in essence a counter. It works by making a container and putting one into the container on the first mouse click. For each subsequent mouse click in the next one second, one is added to the container. After the one second has elapsed, the container has the number of mouse clicks (See script 9).

With the number of mouse clicks available, it is possible to construct Hypertalk "if" statements. "If" statements are structured to say "If x is true then do y". The "if" statements that I made to utilize the information that the counter held said things like, "If the number 2 is in the counter then go to layer 2" With "if" statements you must consider an alternative action for the times when you say "If 2 is in the counter..." and there is instead a 3 in the counter. Here you use "else" statements. "Else" says "If that does not work, try this. "Else" strings together "if" statements so that you can say "If 1 is in the counter then go

to layer 1 else if 2 is in the counter then go to layer two" (See script10).

The taxonomy itself is very general at the top and it gets more specific as you move down. The Hypertalk scripts of the buttons are structured in the same way. At the top layer a single click from anywhere within the graphic button takes you to layer two. However, a triple click at any two different locations within the graphic button will take you to totally two different places. The thing that allows this flexibility is the structuring of the Hypercard button scripts from general to specific.

The top layer graphic buttons act to an inexperienced user like one big button that when clicked takes them to layer two. It is in fact many different Hypercard buttons that do identical things when one click has been registered in the counter. This underlying Hypercard button structure moves the inexperienced user along a step at a time, but it is too slow for power users. To give them more flexibility, the many little Hypercard buttons that do identical things when one click has been registered in the counter do different things for the cases where two or three clicks have been registered in the counter. Triple clicking on a Hypercard button can cut out two steps for a power user and take them directly to their destination from the top layer graphics.

Relationships between Hypercard Buttons and Graphics:

The power of this layered Hypercard scripting is closely tied to the visual layering of the screen graphics. The graphics are structured in a visual hierarchy that corresponds with the scripting heirarchy. The top layer graphics jump out for the inexperienced user who can in turn click anywhere in the prominent area and go to the next layer. The layer three graphics are visually submissive but they are legible to those experienced who are looking for them. At the top layer experienced users can see these graphics and triple click on the desired button to jump to their destination.

As a button is pushed and the user is taken to a new layer, the appropriate new screen graphics are retrieved from Videoworks II. There

must also be new Hypercard buttons to correspond with the new graphics. This is done by utilizing the "hide" and "show" commands. When the button is pushed sending you to a new layer, all of the now obsolete buttons on the current layer are hidden and all the buttons for the new layer are shown (See script 8).

Hiding and showing buttons in Hypercard affects the screen graphics from Videoworks II. If a Videoworks II image is on the screen and a button is hidden or shown, an area the same size and location of that button as erased from the Videoworks II image. To work around this problem, it is necessary to do all hiding and showing of buttons before calling the Videoworks II image. There are some cases however, where this solution is not enough.

Backward Navigation Strategy:

So far, all we have dealt with is moving deeper into the taxonomy. In some cases, when a user realizes that they are on the wrong track, there is a need to back out of the taxonomy. As mentioned earlier, the method for escaping any graphic button is to simply leave the boundaries of that button with the cursor. At that time the screen graphics for the button shrink to the original state, the current layer's buttons are hidden and the browsing button is shown. The script that does the exiting is in every Hypercard button. The problem is that even though the top layer acts like it is one big button, it is in fact many small buttons. Each button says, "When the mouse leaves my boundary hide all the buttons at this layer etc."(See script 8). If the cursor moves out of a small Hypercard button but you still wanted to stay within the graphic button, there is a problem. I had to develop something that only shrunk the graphics and hid the buttons if you left the screen button's boundaries. To do this I got the coordinates for the screen graphics and tested to see if the cursor was within them when it left the Hypercard button. The script said "When the mouse leaves the Hypercard button, if it is

within the button graphics coordinates then do nothing but if it is not within the button graphics coordinates then shrink the graphics and hide the buttons" (See script 10).

Professional Feedback:

When the thesis project itself was finally finished, I decided to get some expert feedback on its' performance. To get this feedback, I took the project to an interactive team at the Xerox Corporation. The team seemed interested in the demo that I showed, particularly by the concept of representing information as a physical object. They also felt that the rotational qualities of the representational model were strong and that having the buttons enlarge while browsing worked well. They mentioned that they would rather have seen icons than text in some areas. They also said that when looking at a face of the representational object, it was not apparent that there were other adjacent faces. This problem, we agreed would have been remedied if the object had been a sphere because the peripheral faces would have been visible on the horizon. The team was not entirely happy with the treatment of the deepest layer graphic buttons. They felt that they were not readable enough even for the users who knew what they were looking for.

Summary:

This thesis was a successful project for me in that I was able to explore a fascinating topic in great detail. The concepts and ideas used in this project had been in my head for some time and it was a pleasure to finally develop them and put them to the test at my thesis show. The project stayed on schedule which was a good exercise for work in a corporate environment. In the future, I hope to do further research and development with Human Interface at the corporate level.

Script 1

```
ClipVW 116, 31, 396, 311
PlayVW VWlocation, 256,171, vwnoclear ,vwnoupdate, vwrange, 45, 1
```

Script 2

```
on OpenCard
  lock screen
  put " 53, 54, 55, 56, 60, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
  put " 16 ,17 ,18 ,19, 20, " into otherIDHolder
  repeat with n = 1 to the number of items in butIDHolder
    set cursor to busy
    hide card button id item n of butIDHolder
  end repeat
  repeat with n = 1 to the number of items in otherIDHolder
    set cursor to busy
    show card button id item n of otherIDHolder
  end repeat
  unlock screen
  ClipVW 116, 31, 396, 311
  PlayVW VWlocation, 256,171, vwnoclear ,vwnoupdate, vwrange, 45, 1
end OpenCard
```

Script 3

```
on mouseUp
  ClipVW 116, 31, 396, 311
  PlayVW VWlocation, 256,171, vwnoclear, vwrange, 23, 33
  go to card id 6246
  push card
end mouseUp
```

Script 4

```
on mouseUp
  ClipVW 116, 31, 396, 311
  PlayVW VWlocation, 256,171, vwnoclear, vwnoupdate, vwrange, 23, 33
  pop card
end mouseUp
```

Script 5

```
on mouseEnter
  ClipVW 116, 31, 396, 311
  PlayVW VWlocation, 256,171, vwnoclear ,vwnoupdate, vwrange, 67, 1
end mouseEnter
```

Script 6

```
on mouseEnter
  show card button id 99
  show card button id 100
  show card button id 101
  show card button id 102
  show card button id 103
  show card button id 104
  show card button id 105
  show card button id 106
  show card button id 107
  show card button id 108
  show card button id 109
  show card button id 110
  show card button id 111
  show card button id 112
  show card button id 113
  hide card button id 20
  ClipVW 116, 31, 396, 311
  PlayVW VWlocation, 256,171, vwnoclear ,vwnoupdate, vwrange, 67,
end mouseEnter
```

Script 7

```
on mouseLeave
    hide card button id 99
    hide card button id 100
    hide card button id 101
    hide card button id 102
    hide card button id 103
    hide card button id 104
    hide card button id 105
    hide card button id 106
    hide card button id 107
    hide card button id 108
    hide card button id 109
    hide card button id 110
    hide card button id 111
    hide card button id 112
    hide card button id 113
    show card button id 20
    ClipVW 116, 31, 396, 311
    PlayVW VWlocation, 256,171, vwnoclear ,vwnoupdate, vwrange, 45, 1
end mouseLeave
```

Script 8

```
on mouseLeave
  if item 1 of the mouseLoc > 122 and item 1 of the mouseLoc < 256 and-
  item 2 of the mouseLoc > 104 and item 2 of the mouseLoc< 237 then
    exit mouseLeave
  else
    hide card button id 99
    hide card button id 100
    hide card button id 101
    hide card button id 102
    hide card button id 103
    hide card button id 104
    hide card button id 105
    hide card button id 106
    hide card button id 107
    hide card button id 108
    hide card button id 109
    hide card button id 110
    hide card button id 111
    hide card button id 112
    hide card button id 113
    show card button id 20
    ClipVW 116, 31, 396, 311
    PlayVW VWlocation, 256,171, vwnoclear ,vwnoupdate, vwrange, 45, 1
  end if
end mouseLeave
```

Script 9

```
on mouseUp
  put yes into the message box
  put 1 into temp
  put the ticks into start
  repeat until the ticks - start > 15
    if the mouseClick then
      add 1 to temp
    end if
  end repeat
end mouseUp
```

Script 10

```
on mouseUp
  put yes into the message box
  put 1 into temp
  put the ticks into start
  repeat until the ticks - start > 15
    if the mouseClick then
      add 1 to temp
    end if
  end repeat
  if 1 is in temp then
    hide card button id 99
    hide card button id 100
    hide card button id 101
    hide card button id 102
    hide card button id 103
    hide card button id 104
    hide card button id 105
    hide card button id 106
    hide card button id 107
    hide card button id 108
    hide card button id 109
```

Script 10 (continued)

```
  hide card button id 110
  hide card button id 111
  hide card button id 112
  hide card button id 113
  show card button id 114
  show card button id 115
  show card button id 116
  show card button id 117
  show card button id 118
  show card button id 119
  show card button id 120
  show card button id 121
  show card button id 122
  show card button id 123
  show card button id 124
  show card button id 125
  show card button id 126
  show card button id 127
  show card button id 142
  ClipVW 116, 31, 396, 311
  PlayVW VWlocation, 256,171, vwnoclear ,vwnoupdate, vwrange,
else
  if 2 is in temp then
    hide card button id 99
    hide card button id 100
    hide card button id 101
    hide card button id 102
    hide card button id 103
    hide card button id 104
    hide card button id 105
    hide card button id 106
    hide card button id 107
    hide card button id 108
```

Script 10 (continued)

```
      hide card button id 109
      hide card button id 110
      hide card button id 111
      hide card button id 112
      hide card button id 113
      show card button id 128
      show card button id 129
      show card button id 130
      show card button id 131
      show card button id 132
      show card button id 133
      show card button id 134
      show card button id 135
      show card button id 136
      show card button id 137
      show card button id 138
      show card button id 139
      show card button id 140
      show card button id 141
      show card button id 145
      ClipVW 116, 31, 396, 311
      PlayVW VWlocation, 256,171, vwnoclear ,vwnoupdate, vwrange, 78, 1
    else
      push card
      go to card id 18214 of stack "blah"
    end if
  end if
end mouseUp
```

Bibliography

1 Hypercard, Version 1.2, Apple Computer, Inc.
Cupertino, CA

2 Video Works II, Version 2.02, Macro Mind, Inc.
San Fransisco, CA

3 Video Works II Hypercard Driver, Version 1.2, Macro Mind, Inc.
San Fransisco, CA

4 Pro 3-D, Version 1.2, Enabling Technologies
Fresno, CA

5 Swivel 3-D, Version 1.00L,

6 Pixel Paint, Version 1.1, Super Mac Software,
Mountain View, CA