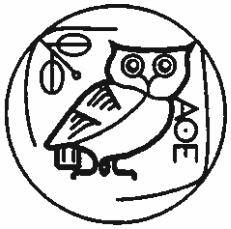




# Bookshelf--A Network Online Reference Tool

# Bookshelf Proper

by Linda B. Merims

## 1. Major Modules: Synopsis

1. Decide upon a format for a catalog card. (Initial/ultimate)

2. Decide upon a format for a shelflist (catalog card +-?) + (display parms). (Initial/ultimate)

3. A program that will read a shelflist and display it on the screen. (Initial/ultimate)

4. An interface that lets you
   - 4.1. Roll over a book and see its title
   - 4.2. Select a book by execing the correct browser
   - 4.3. Close the book and return it to the shelf (as cast-iron a guaranteed exit as the protocol allows).

5. Utility to make a bookshelf.

6. Utility to display and edit the shelflist.

7. Ability to copy books between shelves via direct pick-up.

8. Access the Backroom (should fall out from 4.2 above).

## 2. Bookshelf Proper

Bookshelf proper is the simplest element in the Bookshelf schema.

A Bookshelf holds from 0 to about 15 volumes.

It serves the need as the front-line reference shelf for the individual user--what you put your most-frequently referenced texts upon, and the portals to the Backrooms you most wish to frequent: your own, Athena's, the course's, what have you.

It is the piece I want to get a scratch version of working first.

The Unix command:

bookshelf *directoryname x-geometry*

causes bookshelf to search *directoryname* for a directory called .bookshelf. It defaults to the current directory. *Directory* maybe local, or on any accessible network host.

The library call:

Bookshelf(directoryname,geometry)

has the same effect in an X application.

Once Bookshelf has located the .bookshelf directory, it looks for a file in it called *shelflist*. Shelflist is a small database that defines what Bookshelf is supposed to display. It describes how the volumes are ordered on the shelf, and whether the shelf is labeled. The other thing it contains are the catalog cards of the volumes: the data explaining what files and programs make a volume.

See the Catalog Card definition.

So--shelflist has two things: a collection of catalog cards, and such additional information as is reasonable to specify how they are arranged on the shelf:

- order
- shelf label
- book orientation

Because shelflist is so small, it should not be built out of a sophisticated, slow database. I don't know what Unix tool to use, but ndbm seems to be the one for cottage-sized applications. (Ron has recently suggested a simple text file read with Lex.)

The ***displayer*** module of Bookshelf reads the shelflist file and produces the correct picture of the shelf on the screen.

Displayer ought to be something that could be dropped into the higher aggregate programs (like Backroom) and used to generate bigger pictures.

What is displayer drawing? It's drawing windows doctored to look like books. What the book looks like (fat/thin, cover color, decoration) is taken from catalog card data. We should represent the name of the spine if possible. Otherwise, like the new iconified xterm, we should have a squiggle representation of its title. This distinctive visual aspect of each book is important. After time, and on larger shelves like Backroom, people will re-find volumes by location and appearance, as one finds record albums. Should the book move, appearance will be all the much more important.

The displayer will also recognize Backrooms and portray them properly as doors.

The Bookshelf can be iconified. It turns into a pretty B such as one sees in analysies of font structure books.

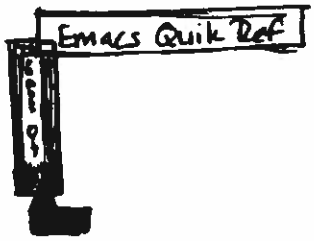
It is perfectly OK to have an empty shelf.

## 3. Interaction

Once the shelf is on the screen, nothing happens until the mouse enters the shelf. Then...

The mouse changes appearance, indicating you're "in" the shelf. The mouse icon has a sharp point, perhaps:

When the point passes into a book, that book's title (from the shelflist) is displayed:




You don't press anything to get this, it just shows up.

There is virtually no delay between the time the mouse enters a "book" and its title shows. Anything that lets the titles lag such that one is seeing the previous title when you've moved on--as Emacs sometimes lags--is avery serious flaw in the nterface. If we relly do this over the network, we may have to cache this stuff. There should be a very rapid, fluid feel to the appearance of items, a 1:1 correspondence between what you're doing and what you see.

Thus, passing the mouse over the books, one sees a rapid flutter of titles popping up and disappearing as you pass out of them.

## 4. Executor

To read a volume, you point to it with the mouse and press ANY BUTTON. (So far, there's nothing else defined for the buttons; this may change.)

At this point, the ***executor*** figuresout how to display the volume. In the original bookshelf design there was no question of what program to run at this point. It ran "the" browser on the filname given in the shelflist. The generality of bookshelf has been greatly expanded. Now it only runs "the" browser if it isn't given any alternate program associated with the book in the catalog.

The volume file you're trying to read may be:

- local to your machine
- a file somewhere on the network, specified in
  - NFS
  - RVD
  - RFS? format.
- a name to be resolved by a server

The browser to display that file can also be local, network, or a name resolved by a

nameserver.

If the browser program is remote, the desired action is *not* to transport the file and its browser program to your local workstation, but rather to start it up on its home machine and have it establish an X connection to your workstation.

This is desireable because you can pretty well trust that the browser and its data will run on the machine it lives on (RT w/BE2, for example) but you have no such guarantees it will work on yours. It should be quicker to start things.[1]

As the executor sets about starting the book, it should tell you it's working on the problem, and renew its messages until the book starts. If there is a failure, ("Bill's Tips for Life at MIT," is on workstation cassandra, can't contact cassandra") it should explain what has happened.

One thing the executor will try (should it be here or a background fix-up that occurs whenever you start bookshelf?) if it can't find a volume is to check the catalog card against the central Athena ***Union Catalog***. If the volume's owner has moved it, and it is a cataloged item, the Union Catalog should have the updated information.

We should make a grid of local/remote x file/browser x filesystem cases and see which ones make any sense.

I don't know how this would work, or if it is feasible.

I am not sure who would provide the remote file facility; would bookshelf do it itself, or rely on some system facility (a la J. T. Kohl's suggestion) to do the dirty work?

The executor keeps one ABSOLUTE area of control: an exit button, always visible, perhaps floating on the screen when it can't predict the browser's behavior, perhaps as part of an enforced border. If the user presses this button, the book closes and returns to the shelf. **Always. Without fail**. Since we don't know how obvious the interface to a browser is going to be, and we're being loose about who can let themselves in, the user must have this final control.

When you take a book down, should it disappear from the shelf? Ron thinks it should just turn grey, so that you can take the book down many times, perhaps showing different pages. I think this is OK.

#### *4.0.1. Remote Execution*

What is the executor using to decide what browser to run? Part of the catalog card is a command string to execute. The executor just interprets this on the correct host system, perhaps with an intermediate visit to the nameserver.

Actually this mechanism is entirely unclear. Saltzer says in the absence of a generalized remote procedure call, it may be best to just rely on rsh to do this on other machines.

Then there is an issue of who the other machine is? A Berkeley Unix that understand rsh? What about a VMS machine?

---

[1]Actually, it seems that X and the file systems we're bringing in are working at cross-purposes, here: one tries to do things on the other machine, while the other tries to copy everything to you.

I also would like to be able to take advantage of various program's existing help facilities. For example, we don't have any online manual for RS/1. However, RS/1 does have a fairly extensive internal help facility. Is there anyway to exec RS/1 and call its help facility in one blow?

Maybe this is an area that Don Davis can help us with.

#### *4.0.2. Backroom*

A ***Backroom***--or any other aggregate--is just a program plus a directory combination.

#### *4.0.3. Procedure Call*

So much for the executor. It should also be a drop-in module for other programs.

## 5. Utility Functions

One end of the Bookshelf is special. When in this area either:

- A menu of functions is presented, or
- If you press any button a menu is presented.

(I'm not sure which I like better. I'd tend to the first, but worry some would expect to see "read book" as an option and wouldn't try just pressing a button.)

This menu lets you:

- ***edit*** bookshelf
- ***borrow*** or ***copy*** books between shelves
- create new bookshelves or backrooms
- delete a book
- exit bookshelf?
- iconify the bookshelf?

The menu behaves like Xmenu (although it is not a card stack):

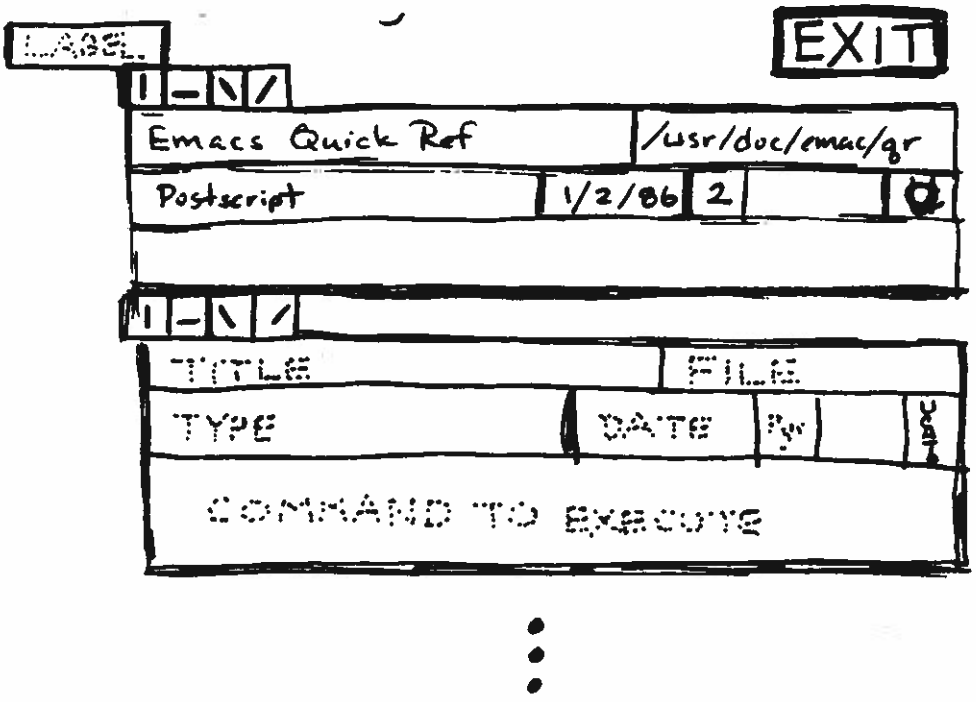
- As you roll the mouse down, the different choices are highlighted.
- If you roll off the menu, it disappears.
- If you either:
  - have held the button down since the beginning, release it on a choice, the choice happens.
  - have released the button, then press + release, it happens.

## 6. The Editor

The ***editor*** is how you add books to the shelf from scratch (without copying them from elsewhere). It is how you re-arrange a shelf, and specify a volume's appearance. You can change fields to your liking: perhaps substituting a text browser you prefer for the one supplied, if you know what you're doing.

The editor is presented as a form, with fields to fill in for each volume, plus the shelf

labeling and book orientation fields. It looks something like:




Most fields of the catalog show. Some may have to be reached with subfields when you enter them; it depends on cluttered things get.

To edit, you move the mouse to the field you want to type in--an editing block cursor will follow the mouse. Again, you don't press anything, it just follows.

### 6.1. Editor Help

It is likely that you won't understand what most of the fields are. You may be able to guess pretty well from other examples, but some--such as the Union Catalog imprimatur--will be obscure.

The editor will have a "point and ask" help facility. This should be a general X toolkit call. If you press the help key, you will receive a block of text explaining what the field is and how to fill it in--for whatever window the mouse is in. This may become more elaborate, but this is enough to start.

(Since Ralph asked me to amplify this idea, I would say that I'm thinking of a XWindowHelp call that could be associated with any window id. When you <do something defined uniformly throughout the event handling mechanism of the toolkit>, the help for that window is displayed. I think there needs to be a "more" facility, perhaps a 2-level tree, built into this. I also think that the structures that define the text data for this help should be interactively editable in the vein of an application generator-- so that if you get the application to compile, and you can get the blank help window on the screen, you can fill it in there, as opposed to Emacs-ing some huge include file somewhere.)

### 6.2. Filling in Fields

You type text into most fields. There are simple line editing functions: delete whole line, insert here, delete this character. Some fields are "radio buttons" of mutially exclusive choices, like should the book be flat, upright or tilted. (Tilted is probably out of the question until somebody comes up with a reasonable mechanism for non-xy aligned windows.) Some are check lists, some may be pull down-aside menus, e.g., to specify binder appearance, or give RVD particulars.

Checking the Union Catalog imprimatur means a new book will be offered to the Union Catalog for an accession number. This may cause the editor to be pickier and more demanding about what fields you need to fill out.

Most text fields are varying width, with reasonable upper limits: perhaps 30 for title, perhaps 100 for command.

I show this as a list down the screen, with scrolling. It could be stacked cards (hard to see data simulataneously). Or, with different menu selection modality, the cards could appear below the books/blank slots they reference.

### 6.3. "Stay" Buttons

Alot of this interaction supposes that, as you roll into windows, things appear, and when you roll out of them, things disappear. Often, however, you want to see two things simultaneously. For example, the most natural way to figure out how to fill out a catalog card would be to display that of an already-cataloged book on the screen, figure out what the fields and their allowable data are from what you see, and fill in your own. This assumes you can see two catalog cards at once. Perhaps catalog cards (and potentially many other windows) should have "stay" buttons at the top that, if pressed, mean "keep this window even if I roll out of it."

### 6.4. Exiting Editor

When you exit the editor (press an exit button), it rewrites shelflist and re-generates the bookshelf display.

## 7. Copy Volumes

The ability to copy volumes from one shelf to another--either simple books or the icon representing a backroom--is a major feature of Bookshelf.

Copying allows you to customize your reference environment, taking those things you find most intersting and useful and putting them where they are easy for you to find. You aren't forced to forever follow somebody else's tree: you make your own.

I think there are really two operations; they appear similar but have very different effects. I call them "borrow" vs "own," but this may not get the idea across.

| | |
|---|---|
| Borrow | To copy the catalog entry of a volume from another shelflist into your own shelflist. |
| Own | To copy the catalog entry, text data, and also? the browser from another shelf. |

Usually, you should just need to borrow. Owning a book can take alot of space. After you've brought over an "owned" book, you may discover its browser will not run on your type of workstation and you'll need to edit its catalog entry to specify a substitute.

Copying is done with "direct manipulation." After you click "copy" the system should do something intelligent about making sure your personal bookshelf is "on top." If you're copying between backrooms, this could get hairly. Hopefully, you are adept enugh with the window manager that this is not a problem.

You then point and press a volume to pick it up, and drag it to its new resting place. Other books on a shelf will move over for it.

Behind the scenes, the catalog card of the volume you copied has been duplicated from shelflist to shelflist.

To copy a backroom if it doesn't already appear as an icon, iconify it first, then move it to the new resting place. (I doubt that this is intuitive.)

Evidently, direct object manipulation using X as the communication medium is not possible under X Version 10, but a mechanism exists in V11. If we want to have copying at all in a Version 10 implementation, we must find some other mechanism.

### 7.1. Step-by-Step Help

Again, Ralph asked me to say something here. Many operations are multi-step. For a help to be successful, it can't just static-ly define things (list of commands available), but may also need to show you the two or three things you have to do in a certain order to accomplish a task. In the simplest case, you just open a window with step 1-2-3 described as text in it and make it "stay" until the user completes the task. In the most elaborate case, you may have some kind of animated movie doing this. What facilities could a toolkit provide to make this easier?

## 8. Creating

Creating is equivalent to the:

**`bookshelf create directory geometry`**

command, where directory defaults to current, and geometry to a position just below the existing bookshelf, if any.

An empty bookshelf appears. An empty shelflist(_1) file is created. To fill the bookshelf up, use copy or the editor.

I think it is a good idea to make Backroom a menu option, since I suspect most people will need and want one.

## 9. Shelflist

Some of the material in this discussion of the ***shelflist*** has been superceded by the catalog card concept described later in its own section.

Questions: How many books can be on one shelf? Is a bookshelf fixed length, or infinite? I lean toward fixed length.

The shelflist contains two layers of information; a layer that controls how the shelf is displayed, mostly controlling where labels are put on the shelf and what's in them; and then the layer of catalog cards defining each volume on the shelf.

Possible fields are:

Title — What to display on the spine of the book, or in the title window that appears when you pass the mouse over it.

Filename — The file with the data in it. The format we support depends upon the file system we use. Plain local files, NFS, RVD, RFS. If the filename is an RVD, we need to appeal to a nameserver to find out to spin it up, or include that information in the catalog card.

Type

- Text
- Text with line printer controls
- Postscript
- Scrapbook
- Binder
- Video
- nroff/troff

You care about type because you may call different viewing programs depending upon the type. The viewer needs to know what to expect. It may be best to allow a hook so the user can specify their own displaying program, perhaps bizarre stuff like Ingres databases. (Note: This is the idea that eventually turned into Executor.)

Number of Pages — So that the browser can give a visual impression of size. So that displayer can show thickness.

Owned or Borrowed — To display on spine? Send appropriate messages. May be determined dynamically from the filename.

Spinebinding — One of several types to control appearance.

Orientation — To control horizontal or vertical appearance.

Program to Call — and a parameter list. Such as backroom.

And? — Keywords? author? owner? synopsis? physical size (n x n inches)?

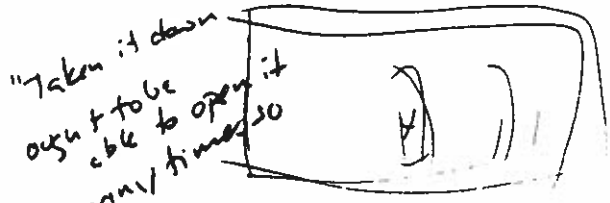
The shelflist could be a text file, an ndbm database, an ingres database? What it is built out of at any level of aggregation should reflect how complicated the problem is at that level. Only the largest should be databases.

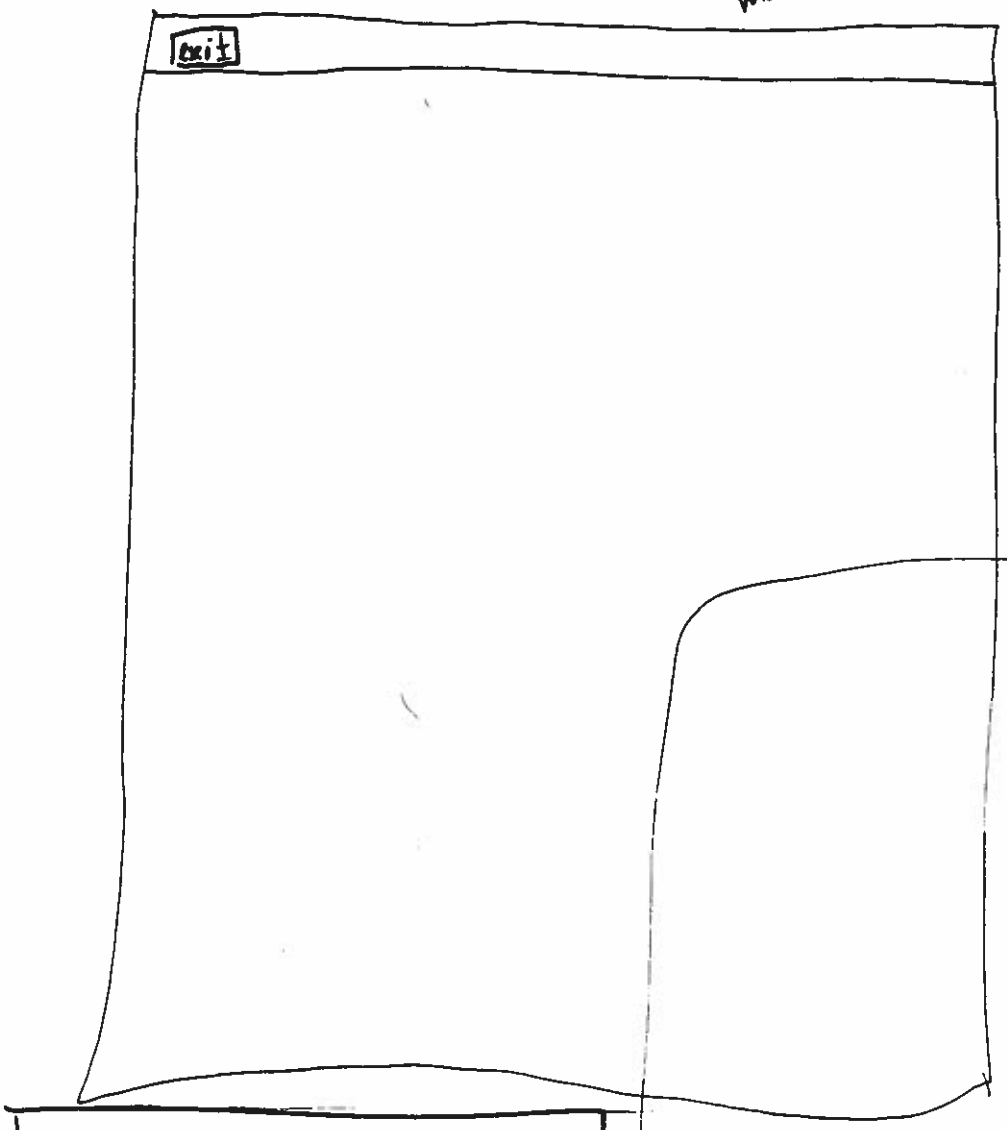
## 10. Sketches

Some sketches showing impressions of various of the pieces outlined in this chapter.

# EXECUTOR

"Standard sandwich"



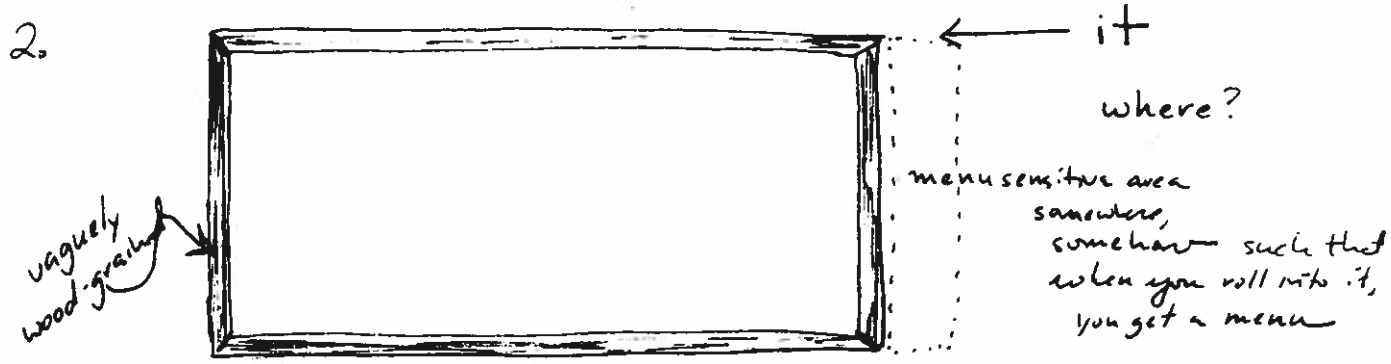

You can open the book many times,
As a solution to looking in more places than one at once.

Come hell or high water, you can get out.

- perhaps subroutines of (problem of containing willful programs)
- perhaps the window manager has a universal facility.

# Command Interface
# Making a Bookshelf

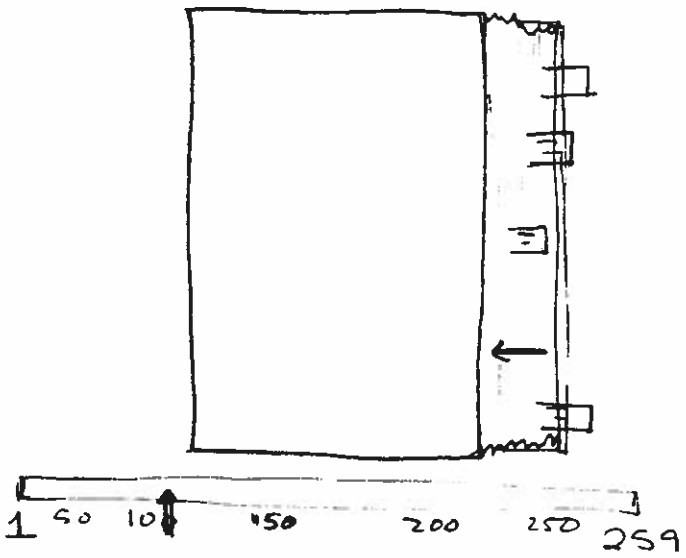
1. bookshelf [make / create / new] ← you

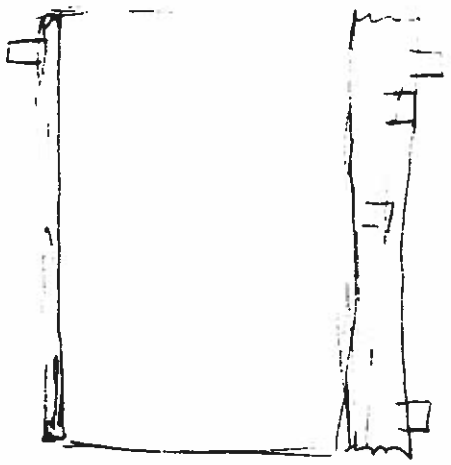
2. 


3. 

# Browser

Thoughts on Browse



Not a simple scroll bar - absence of visual cues
No sense of scale until you've already tried. Is ½ inch 30 pages or 1 page?
Not intuitive — position on this page

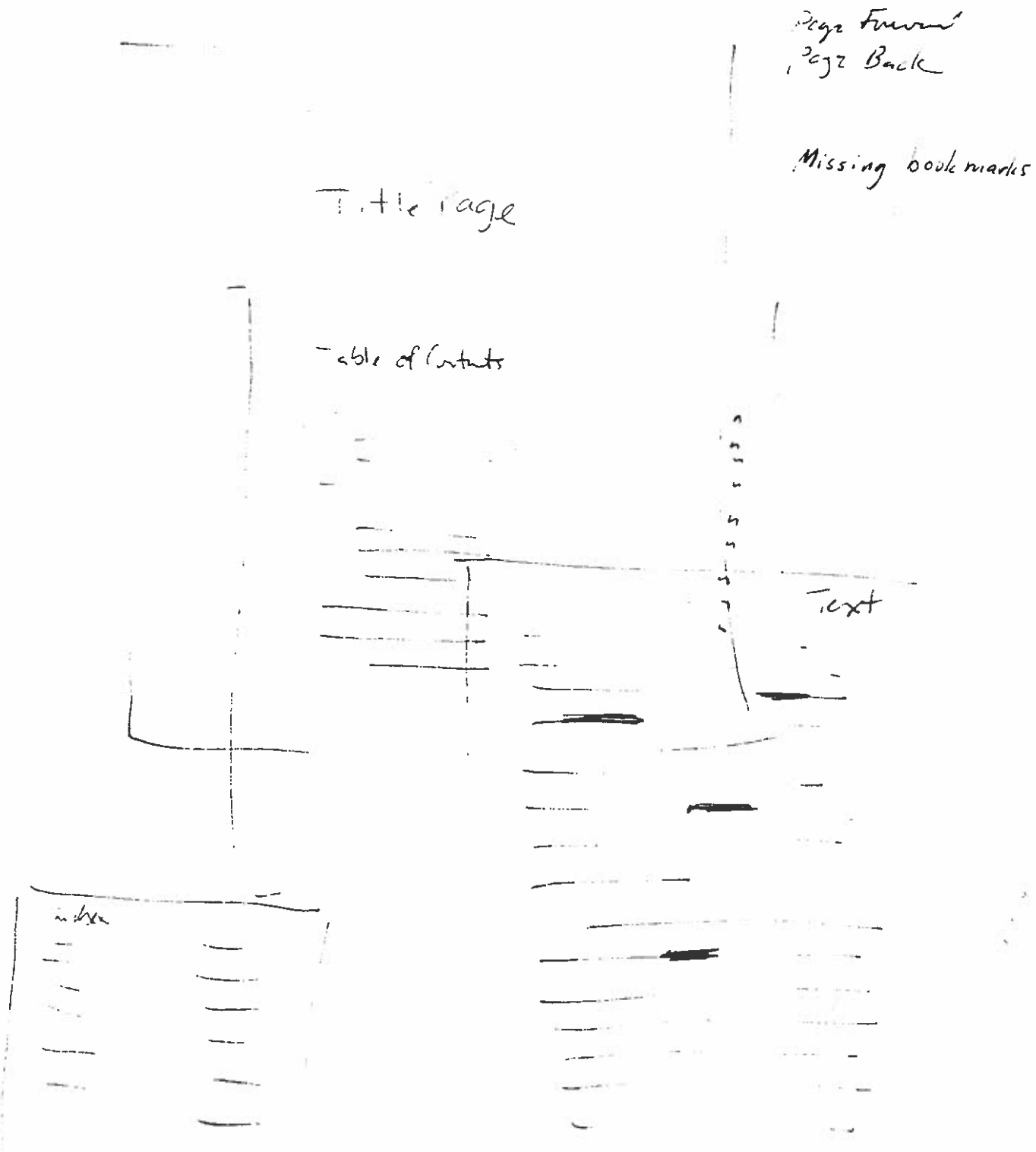
The book is "blind"

visual effect -

Interaction an animated experience . . .

The VS100 Help Browser - UIS

document man page



Postscript

— Adobe — what are they doing
- need it in source form so that we can adapt it
- need to find out lots more about this.

✓

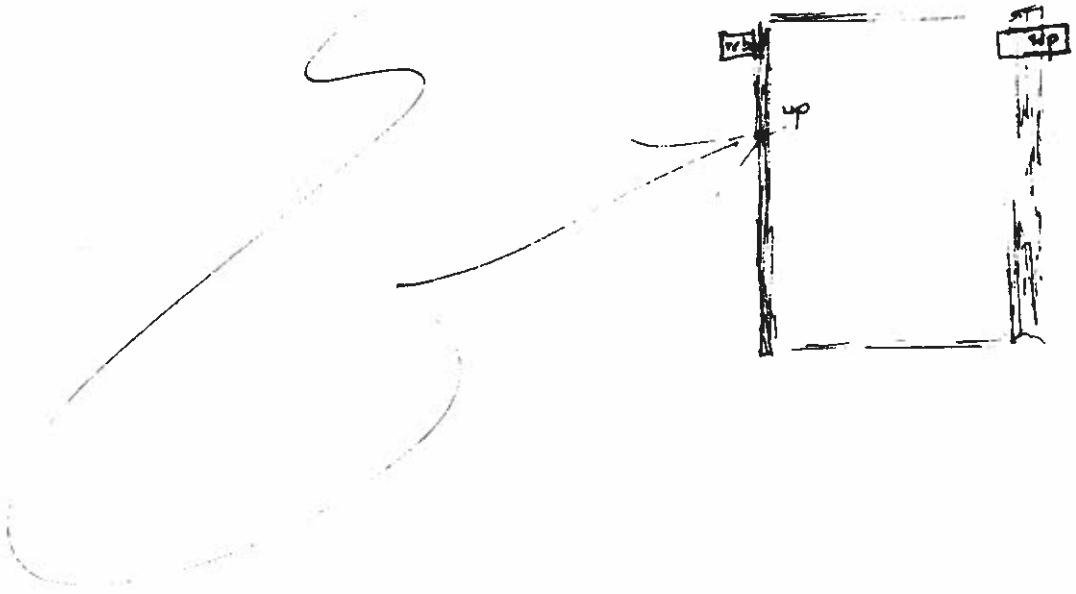
hitmaps —

# Types of Books

# Scrapbook

~A collection of "snipped" bitmaps
from anywhere any kind of window

Up is the opposite of down. You need to up an RUD before you can use it. You can't up it unless somebody else has clowned it. You should always down an RUD you've upped so the next guy can up it too. To up an RUD:


from an idea by Beth Anderson & Frank Colan

# Backroom

Backroom