

exploring the **mnemonic** user interface

Christian Lagerkvist, MoS HCI — christian.lagerkvist@gmail.com

»It is immensely ingenious, immensely complicated, and extremely effective but somehow at the same time crude, wasteful and inelegant; and one feels that there must be a better way of doing things«

— Christopher Strachey, 1962

Prologue

The manner in which files are visually organized, all according to the popular desktop metaphor, concur with conditions applicable twenty years ago. Over time, these conditions, technical as well as user oriented ones, have radically changed. The desktop metaphor has not.

This article is an offspring of personal reflections over too much time being spent traversing file structures and organizing windows in the user interfaces of today's modern operating systems.

Background

The 1980's is usually denoted the personal computer's very childhood. This is the time when we realized that anyone has use for a computer, and computers targeting ordinary folks seriously hit the production lines. This is also the time when Apple Inc. refined the desktop metaphor into roughly what still is in use today.

In those days, hard drives were a luxury, and in effect we stored our files on floppy disks. Given the moderate storage capacity of floppy disks and hard drives at the time, the evolving desktop metaphor really was adequate for the job.

Managing fifty files divided among five folders is a relatively easy task. This however is not the case of a modern file system, hosting ten thousand times as many files divided into an army of subfolders.

Traversing Folders

Navigating a file structure incorporating tens of thousands of files has become an everyday task. We do this without giving it much thought, as it is by all means natural to first locate the object we wish to manipulate. Let us explore what navigating a file structure is about, and isolate some of the problems involved:

Naturally, when locating a file, the location of the file has to be known. Not a geographical location, but an analogous one expressed in terms of which way to take in a series of branches – a file path.

These branches are made (i.e. the folders are opened) one at a time. For every branch, a decision has to be made as to what branch to take next. If the location of the file is unknown, or almost known, this decision can only be made after having inspected the new available branches, after which the most likely one is chosen. Thus the key to navigating a specific desktop file system fast lies in knowing the exact physical, on-screen location and looks of every folder - as well as their relative positioning (recognition and predictability) along the path to the destination. As the image formed by the letters in a file or folder name eventually becomes the cue, the actual meaning of the word formed by the letters becomes superfluous.

Unix devotees often claim typing the file path is faster than using the mouse. Mastering effective navigation in the command line case lies in memorizing file paths by reconceptualizing them into a personal mental model of the file system. Contrary to navigating a file system via a desktop metaphor (recognition and predictability), navigation is entirely based on recall.

Regardless, it all boils up to this: A user has to associate with, and access, a specific folder for every traversal of the file path elements— even if the user knows the file path by heart!

It should be pointed out that there is a kind of file structure navigation often mistaken for the recognition/predictability version of the desktop metaphor, but with a subtle difference: views of objects are not preserved, that is, the interface forgets the layout of the window content for a folder, and instead arranges objects dynamically with respect to the size of the window. Since this »feature« breaks spatiality, thus taking away the possibility of navigating by predictability, the user is forced to browse file and folder names until reminded what branch to take. Regardless if the user knows the file path on her five fingers.

The intensive thought process of locating doesn't even begin with traversing file structures; it begins with the user having to determine

the more convenient way of locating the file or folder: the »recent items« menu, an already opened window, a shortcut/alias, manual browsing or, if the user has the option, to type the full path of the designated object.

Also, we have learned to associate each desktop object with an icon, revealing what kind of object it represents (file or folder). When locating a file, every single object we double-click on the way is a folder—and every folder has the exact same icon (at least in most of our operating system gui's). So having a special icon for folders boils down to one thing: increasing the size of the target.

The discussion above is a result of file structure visualization only. Moreover, there are a number of other, administrative questions to consider when manipulation files. For instance, deciding whether a specific window should be closed, if double-clicking a folder should open up its contents in a new window or replace the parent one, or if a source file is on another drive than the destination meaning the file will be copied instead of moved. Additionally we have to maximize, minimize, resize, dock, move and close windows. No wonder the main task is phased out from our 7 ± 2 memory chunks, and we find ourselves in loss-of-activation states.

Navigation takes concentration. A lot.

*»Men are disturbed not by things, but
by the view which they take of them«*

— Epictetus ~ 100 BC

The Mnemonic User Interface

Regardless whether navigation takes place using the desktop metaphor or the command line, we still have to focus on passing information between our own mental model and the computer's representation of the file structure.

What if the representation and the mental model were one and the same? In other words, what we see on the computer screen would be nothing less than our own mental model of the file structure. We would not be forced to recall, but rather recognize the location of an object, since the visualization on the screen would aid us just as well—a mnemonic user interface.

So what we want to do is create a visualization of the file structure that suits the human brain's way of representing the information. So what representation does »suit the human brain's way«? This is probably a question with many answers, and I shall settle for presenting a representation which I believe would serve the purpose very well.

250 million people can't be wrong

This is the amount of people that have adopted a technique called »mind mapping«, as a method of learning and memorizing information. Mind mapping, originally developed by Tony Buzan, is a proven method aimed at aiding learning by representing information in a »brain-friendly« way.

The method of mind mapping is beyond the scope of this article, but roughly it is about writing pieces of information, each inside a bubble, on a blank paper, and visualizing their relations with arcs between the bubbles – growing outwards from the key topic which is located in the center.

How then does this concept fit into the problem of visualizing files

at operating system level? Think of the »key topic« (middle bubble) as your root folder and its »pieces of information« as subfolders surrounding it in bubbles. The content of a subfolder (bubble) is, of course, the files it encloses.

Recurse!



Interaction

The interface must incorporate a means of panning and zooming into the desired location of the mind map (see Pad++ a k a Jazz, or Raskin's the Humane interface). Is it at all possible to build an interface that lets you zoom in and pan to the desired view with just a click on the mouse button? The problem is analogous to panning from New York to Wales on a world map by just using a click, a drag and a release. A number of methods can be thought of.

The bubbles should dynamically increase and decrease in size, as the user moves files around the system. It could be argued that such a behaviour would break spatiality. Indeed it would, but bare in mind that the world will only change due to direct user interaction, and thus only the user that should have the power to shape her own file representation. The system should also allow for the easy creating of labels—text signs, borders and colors, in order to provide visual cues for navigation and passive memorization.

Now, let me present three very interesting properties that such a mnemonic user interface would feature:

- 1) No more windows. This means no scrolling or resizing, or moving obtrusive file/folder-windows out of the way. Windows are for applications. The bubbles should of course align so that they would never cover each other.
- 2) No more folders. Since we've translated folders to an analogous structure based on geometry, i.e. the physical location of objects, we have an intuitive (in a justified sense) way of grouping files together: Put them next to each other! Since the folder concept just became obsolete, new »folders« (i.e. bubbles) would be created automatically upon the user placing two files next to each other outside any of the existing bubbles. Move a third file into the bubble and it will grow to encompass all three (as the source bubble shrinks, adjusting its size to the number of files it surrounds).

3) A foundation for more visualization—a shortcut/alias could actually point out (in the direction of) the folder/bubble/view it represents. Visualizing any file dependencies in the system could be done by simply drawing arcs between the files. Even more interesting would be the visualization of network files, firewall, write protected folders...

Introducing new issues as well

Since I haven't had an opportunity to implement and test the visualization, it is hard to predict what obstacles it would present in »real« situations. One is however the action of comparing the content or meta data of two file system objects. Since today's operating system interfaces are based upon the desktop metaphor, they provide a natural way of comparing folders: open the folders and put the windows next to each other. In the highly spatial user interface presented here, there is no natural way of accomplishing this unless the folders are already closely (and thus geometrically) related.

Allowing the user to create virtual views of folder structures is straightforward repulsive. Conclusively, I see no other solution than violating one of the corner stones of mind mapping: the static relation between objects. But. Since a mind map is a two-dimensional vector space, the relation between any two locations can be expressed in terms of a magnitude and a direction. Violating (surrendering) only the magnitude scalar would mean that the direction between bubbles is always

the same, but that a bubble, upon requirement, can temporarily be visualized next to one another. Again, I haven't had the opportunity to try this myself.

Conclusion

There will come a time in the future when people will laugh at today's user interfaces. A fact brought into my mind each time I have to change application just to read a dialog box, or I spot an open/save-dialog which for some reason represents my file system in »compact mode«, or when I conclude that the four corners of the screen still are unutilized in all of today's »modern« operating system gui's. Twenty years of gui development—and I can't even take a quick note of something when I'm on the phone! I just want to click on the desktop and type. Now I'm forced to start some application, type my note, choose »Save«, make up a suitable file name and save it on the desktop. Surely I could have my »quick_notes.txt« on the desktop, the file in which I take all of my notes. Then all I have to do is remove all windows so that I can access the file, double-click it, locate the bottom of the file, type my note, save it, close it and bring back the window I was working in.

Today I'm just looking for a spot where I can type, and that spot is usually the address bar of my internet browser window. Have you ever typed a phone number or a note in the address bar of your browser window?

feedback to the author...

Comments

Where did you find this article?

Enter your e-mail address for receipt