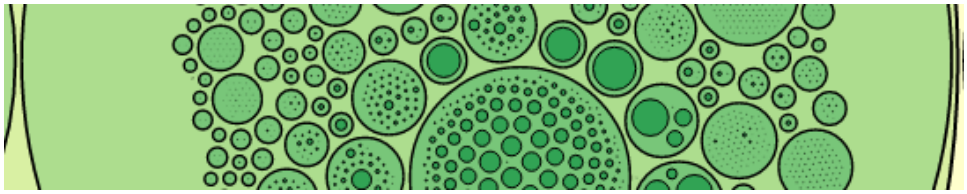
Directory Tree Visualization for Hard Drive Management

Philippe Poirier* Bruno Dumas†

Universit Claude Bernard Lyon-1



ABSTRACT

A directory tree or treemap is a graph in which each node is a folder or a file, and each children is a sub-folder or a file included in a folder. Directory trees are widely used to visualize hierarchies (e.g. Windows file hierarchy) since they are quite easy to understand. In this paper we propose a novel visualization that combines a global view for a quick and easy access to the whole environment, a more specific view to get all the needed informations such as the file extension, and a toolbar to keep an eye on the current path of the examined file. Thanks to this visualization, it is easier to manage hard drives.

1 INTRODUCTION

Tidying up a computer hard drive can be very annoying. When done in the hurry, sensible data may be lost in the process. Many softwares offer directory tree visualization such as WinDirStat[1] or JDiskReport[2]. However, those visualizations are limited. Indeed, either they are too global and it becomes unhandy to retrieve important informations, or they are too specific and there are no landmarks to rely on.

That is why we introduce this visualization which unify a global view with a directory tree and a more specific view with a circular pack layout.
We tried to combine old ideas with new technologies and algorithms for a clean result. We managed to implement degree of interest functions [3] to improve a lot our visualization and thus have faster running time. Our visualization allows a person to easily manage its hard drive with an intuitive interface.
It allows a non expert user to do various managing tasks in single clicks and always keep an eye on what is happening contrary to shell managing.
This work is only a glimpse of the possibilities this visualization can offer.

This paper first reviews related work on hard drives visualizing techniques and then describes the two visualizations we are relying on in detail.
We then show by example their use and interactivity.
We finally discuss the results of our visualization and how our work can be improved to get additional features and to target new people with other uses.

*e-mail: philippe.poirier@etu.univ-lyon1.fr
†e-mail:bruno.dumas@etu.univ-lyon1.fr

[1]https://windirstat.net/
[2]http://www.jgoodies.com/freeware/jdiskreport/
[3]See related work part

2 RELATED WORK

Visualizing tree directory structures is a complicated problem because of the huge amount of information and the limited available space .
For this reason, a lot of people have been searching to solve this problem, and thus many different visualizations have been created. All the visualizations described below have inspired us in a way or another.

2.1 Classical Hierarchy tree

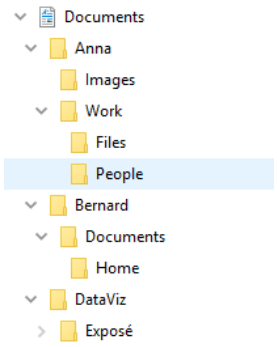


Figure 1: A classical hierarchy tree in Windows.

Probably the most famous one, this visualization is widely used in Operating Systems such as Windows. It is an easy structure to understand and to navigate through. With a single glance, you can locate a folder or a file and see its relative position in comparison with another folder.
However, apart from the position, we have no useful information about the file we are looking at.
Another default, if there are a lot of subfolders in a recursive way, it is easy to lose track of the root folder and so the path.
Hence this second visualization :

2.2 Toolbar

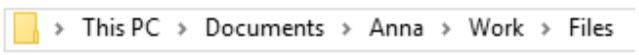


Figure 2: A toolbar or pathbar.

The toolbar is a great way to keep track of the path from the root folder.
It is clickable and thus provides a good and quick way to go to a parent folder.
It is implemented in a lot of devices and softwares, most of the time to complete the directory tree.

2.3 Treemaps

In the two previous visualizations, we had no informations over the file we were looking at.
In 1990, Ben Shneiderman [4] tried to create a compact visualization of directory tree structures which gives various informations of the files we are looking over, named Treemap.
This Treemap is about turning a tree into a planar space-filling" representation.

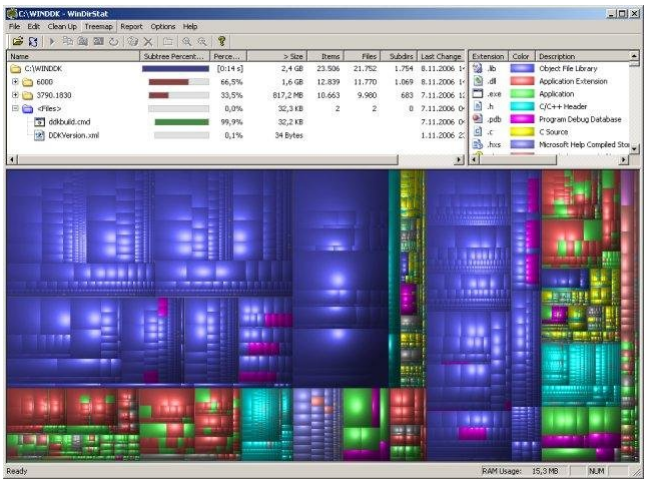


Figure 3: A Treemap.

A lot of information is provided in this visualization.
Firstly, each square is a folder or a file, and thus if a square is inside another one, it represents a file/folder and its parent.
Then there is the size. Each square size is proportional to the associated filer/folder size. It is then easy and quick to see how big is a file and it is a good approach for hard drive managing.
Finally, the colors can represent many things depending on what we are interested in.
For example, it can show the file extension, the last modification time or last access time, the owner of each file and so on.

2.4 Circular Treemaps

A variant of the Treemap as seen before. [5]
Instead of using squares, this visualization uses circles. Since there is more void between each file/folder, it is prettier and easier for the eye to understand.
We used this visualization as a base for our project, combined with a directory tree and a toolbar in order that each visualization makes up for the defects of the others.

2.5 Degree of Interest

We got inspired by the work of Frank van Ham and Adam Perer[6] about Degree of Interest in a graph.
Their method computes the Degree of Interest of a selected node in a graph to display the node and its neighbourhood only. This partial graph allows a clear view for a better understanding and highly decrease the computing time, which is a plus for slow devices.

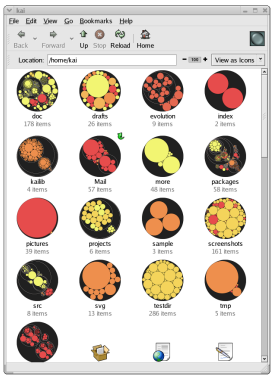


Figure 4: A circular Treemap.

3 Project Description

3.1 Dataset

We chose to visualise the hard drive of a Raspberry Pi microcomputer running the Raspbian distribution. Raspbian is an adapted version of Debian. Microcomputers like this one are very small and have a very low power consumption. They are designed to be used as personal home servers. The computer we collected our dataset from is running a node.js server connected to a local mongo database.
The dataset was extracted using a bash script to generate a csv file. Then regular expressions were used to clean the file and enriching it with more information. Thus, we calculate the height and the file extension of every node in order to reduce computation time during visualization.

3.2 GUI parts

The interface we designed is divided in two main parts and a toolbar. Figure 5 shows a screenshot of the whole view. The toolbar shows the path of the currently selected file or directory. It also contains a select list that allows the user to choose what parameter is used to colour the nodes. The last element of this part of the GUI is a hidden legend for the colors, that shows up when the mouse points at it.

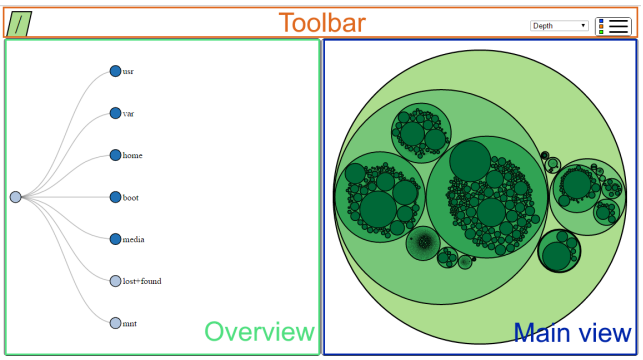


Figure 5: Global View.

3.3 The overview zone

The main display zone is divided into two equal size zones. The first one is called overview. It displays a standard tree view that allows the user to locate himself within the directory tree. In the overview, every node is manually expendable. The user can explore the tree

[4] http://www.cs.umd.edu/hcil/treemap-history/
[5] http://lip.sourceforge.net/ctreemap.html
[6] https://d17oy1vhnax1f7.cloudfront.net/items/1I0F0J0F1P0T2q340A34/10.1109_tvcg.2009.108.pdf

step by step, digging through the tree one layer at a time. To keep this view easy to read, very few information is displayed about the nodes. Only the file or directory name is displayed. We use a color code to indicate whether the node is collapsible or expandable. Dark blue nodes have children that are currently hidden but that can be displayed by clicking the parent node. When a node is clicked in the other part of the screen, the direct path from the root to this node is highlighted in the overview.
This zone is zoomable and draggable so the user might explore a large portion of the tree without worrying about the space taken. The purpose of this view is to give a customisable landmark to find the information they seek in the second part of the interface.

3.4 The main view zone

The second part is called main view. It displays the tree in a circular pack layout. It is a fractal-like structure where each file or directory in the tree is represented by a circle. A directory circle contains all of its children inside its area. The size of each circle is proportional to the size of the directory or file on hard drive. Every circle is colored. The user can choose which parameter is used as color. Available parameters are file owner, file extension, modification date and depth.
The file owner parameter can be useful to quickly locate a specific file. For example, a system admin might want to find all files created by a software deamon on his system. On multi-user systems, it can show the relative disk space usage of different users. Each user is assigned a unique color.
Coloring files by extension may help discover specific files such as configuration files in installed programs. As for owner, each file extension has its own color.
Modification date coloring might be the most useful feature. It allows the user to track software that hasnt been used for a long time, old files that need to be archived and removed, involuntarily newly installed adwares, etc. The most recent files are painted dark red and it lightens up as the files get older, until becoming almost white. These colors were chosen because the eye is both attracted by the dark red and the white files among the majority of light red and pink files.
The last parameter, depth, is probably the least useful for advanced users. However, it generates very beautiful views. The color tone is green and it goes darker as the nodes go farther from the root of the tree.
As the mouse moves over a node, information about this node is displayed in a tooltip (figure 6). We can read the full path, the name of the owner, the size on disk and the modification date of the file or directory. Tooltips provide a very convenient way of displaying information without overloading the interface. It brings the desired information close to the users focal point.

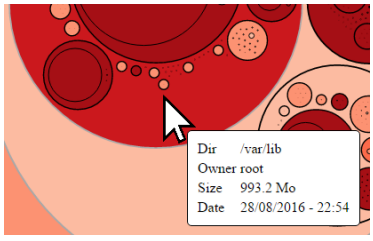


Figure 6: Main View.

Moreover, to make the interface clearer and easier to read, we use a degree of interest function that hides distant nodes until the user explores the area near them.
We based our implementation of the degree of interest on two parameters that are the horizontal and the vertical distance between a node and the focus point.

3.5 The toolbar

As explained above, the toolbar shows the path of the currently selected file (figure 7). Parent directories are clickable and are colored the same color as the node they represent in the main view.

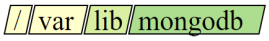


Figure 7: The Path.

In addition to the path view, the toolbar contains a selector that allows the user to choose one color parameter amongst the ones listed above. Changing this setting updates the main view and path view.
The color scales are explained in a legend panel (figure 8). To save screen space, this panel is hidden. It shows up when the mouse pointer moves over the legend icon in the top left hand corner of the screen. We could allow ourselves to hide the legend because it only matches colors with values that are already displayed the tooltips that provide more detailed information about the file or directory pointed by the user in the main view.

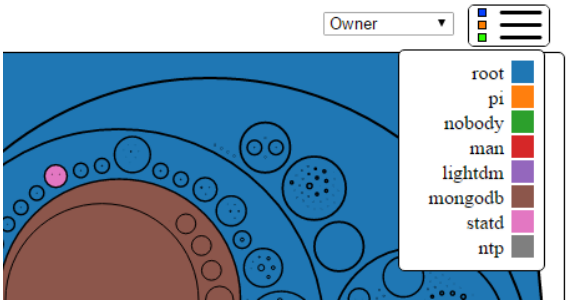


Figure 8: The legend.

3.6 Interactivity

When a circle is clicked in the main view, the view is zoomed so the selected file or directory is centered and takes most space on the view. At allows faster exploration than the overview because depth layers can be skipped. In order to allow the user to locate the selected directory or file, the overview is updated. Every node in the direct path of the selected node is expanded and painted red, as well as links. This makes the selected node visible in both views. This is the only operation that updates one of the main views from another. Each view needs to be manipulated separately for the user to keep his bearings.
However, because of this zooming effect on the main view, it is not easy to climb back the tree. Indeed, as the selected node is displayed very big, the user can only click on this nodes parent. This is the reason why the file path is displayed in the toolbar. Each parent directory of the selected node is clickable. This has the effect of zooming the main view out, and center it on the clicked directory.

4 Discussion

This work is based on rectangular treemaps in which we chose to use circles instead of rectangles to improve the visual comfort. Indeed, contrary to rectangles that can fit very well one against another, packed circles waists a bit of screen space. Thus, the interface is clearer and the different layers are more easily distinguishable. The lost space is compensated by the use of tooltips and collapsible legends. Due to the technologies used, the tree displayed in the project cannot be too deep or too broad. As a matter of fact, processing a circular treemap is rather expensive in calculations

and in memory usage. We decided to develop this project in a web format in order to be able to share it easily, however Javascript is not as efficient as other languages can be for visual computation.
An improvement we tried to implement was to not load all the datas at once but only the first levels at first, and then running what remains in the background. It allows the user to start his task without waiting a huge amount of time, depending on the numbers of files stored in the hard drive.
Another improvement quite easy to make is to create an interactivity between the directory tree and the circular treemap. There already is one, when you click on a folder in the circular treemap view it opens the corresponding branch in the directory tree. We now want the other way, clicking on a node in the directory tree (or rather hodling CTRL key and then clicking to allow free exploration of the tree without changing the circular treemap view) to open it in the circular treemap.
Finally, a search bar can be added to open a folder from typing its path in it or to search all folders/files containing the desired expression.

The initial idea of the project was to design an interface for file management. It was supposed to be used by system administrators to have a dynamic global view on their system to monitor and correct faults from a web interface. Files can be moved from one directory to another using mouse and keyboard controls. Files can be filtered by owner, date or extension. This management tool is pretty intuitive to use but it might lack technical features such as scripting and task automation.

5 Conclusion

This paper offers a new way to visualize a directory tree. In order to improve the way users interact with the interface, we designed an interface composed by two main parts, each one covering a specific purpose. The overview provides a global vision over the disk directories whereas the main view allows the user to freely explore the tree. Overlay tooltips provide information without impeding the interface.
Our interface can be used to explore any type of tree-structured data such as a family tree or a decision tree. Our work could also be improved by adding more parameters to the data that would provide different possibilities for coloring nodes.

Acknowledgments

We wish to thank Aurlien Tabard, Romain Vuillemot and Nicolas Bonneel for their support and their pieces of advice, and all the D3.js community for their inspiring work.