2013

FishEYE: A Forensic Tool for the Visualization of Change-Over-Time in Windows VSS

Jin-Ning Tioh
Iowa State University

FishEYE: A forensic tool for the visualization of change-over-time in Windows VSS

by

Jin-Ning Tioh

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering

Program of Study Committee:
Guan Yong, Major Professor
Arun K. Somani
Mani Mina

Iowa State University

Ames, Iowa

2013

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

For the digital forensic examiner, being able to perceive change-over-time supports the goal of being able to explain "what happened." In our thesis, we focus on the improvements brought to digital forensic analysis by the visualization of forensic data and its application to digital forensic data that records change-over-time, specifically for a directory-tree structure and its content. By perceiving digital evidence visually, investigators are able to speed up the forensic analysis process, and at the same time better comprehend new unique relationships between data as well as more easily comprehend it in terms of its global context.

To provide multiple snapshots of a directory-tree structure, we chose to utilize Shadow Copy (also known as Volume Snapshot Servie or Volume Shadow Copy Service or VSS), a technology included in Microsoft Windows which allows for the taking of manual or automatic backup copies or snapshots of data (including whole volumes) over regular intervals. VSS was chosen since it is a potential gold mine of forensic information, having been included in every version of Microsoft Windows since Windows XP.

In this thesis, we propose and develop a tool to take advantage of the information contained within VSS by applying the fisheye focus+context visualization approach to the directory tree structure, with a series of segmented boxes for each to represent change-over-time for each directory/file, accomplishing our goal of providing investigators a clear picture of how a directory-tree structure has changed over time at a glance.

ACKNOWLEDGEMENTS

Appreciation is a wonderful thing. It makes what is excellent in others belong to us as well.

-Voltaire

First and foremost, I would like to thank Dr. Yong Guan who supported and advised me throughout my sojourn at Iowa State. He instilled in me and reinforced the value and importance of being a life-long learner, continually challenging and encouraging me. I hold great esteem for his unfailing enthusiasm and physical insight. I am also deeply indebted to him for giving me a tremendous amount of latitude in setting my own goals and seeking solutions to problems, yet remaining approachable and encouraging whenever I reached an impasse. Additionally, I am thankful to Prof. Arun K. Somani, and Prof. Mani Mina who agreed to serve on my committee and oversee my work. I would also like to thank Vicky Thorland-Oster from the Electrical and Computer Engineering department, as well as Dr. David K. Holger and Dr. William R. Graves for their instrumental assistance in arranging for my final defense and graduation.

Last but not least, I would like to acknowledge my family for their tireless love and support over the years. My brother, Jin-Wei Tioh who was also at Iowa State, for his camaraderie and the unique companionship of a sibling. My parents, Ngee-Heng Tioh and Meng-Gek Lim, for working hard to provide me with the best opportunities in life, embodying the ideals I aspire to achieve and teaching me far more than mere words can express.

CHAPTER 1. INTRODUCTION

Although recent years have seen a steady decline in violent crimes including murder, forcible rape and armed robbery within the United States, the phenomenal growth of the internet has seen a drastic increase in cyber crimes such as child pornography, extortion, financial scams and sabotage. A recent case involving a theft ring that attempted to steal some 220 million dollars with the Zeus Trojan horse highlights the severity of the situation [7], and that cyber crimes are not simply an annoyance. Besides the use of a computer in the commission of a crime, these criminals share one more similarity - the chances of their being caught and successfully prosecuted are relatively small [1]. Tracing down the evidence found in plain text documents, log files, or even embedded in image and system files utilizing computer forensic tools can already be a difficult task at the best of times, let alone if more tech-savvy criminals attempt to conceal information by deleting it, encrypting it, or employing any number of other tricks available.

Computer forensics is the science of acquiring, retrieving, preserving, and presenting data that has been processed electronically and stored on computer media [2]. Being a relatively new discipline, there is a lack of complex evidence analysis methods that is intuitive towards the investigator. Some of the most widely used forensic software tools such as EnCase [3] and AccessData's Forensic Toolkit (FTK) [4], simply load up a digital image of computer media and present information on the file system and the files contained within in a textual format. And while some level of automation exists, by and large the investigator has to do the analysis of the information presented him or herself. Add to that the rapidly dropping cost per gigabyte for hard drives [5], and an investigator has to spend an inordinate amount of time sifting through a very large amount of data to find all the relevant

information pertaining to a criminal case [6]. Thus, it goes without saying that there is a need to improve the computer forensic tools involved to reduce the tedium for forensic examiners. The two improvements that we are interested in is the visualization of forensic data, and applying it to digital forensic data that records change-over-time, specifically for a directory-tree structure and its content.

Our contributions to the field of computer forensics include the modification and application of the fisheye view as well as a segmented 'change-over-time' box of our own devising to a Windows Explorer like interface that helps represent temporal information about files such as modification and access times. This should allow investigators to gain the ability to quickly sum up at a glance the changes to a file of interest. Lastly, our software framework should allow for additional features and data mining techniques to be applied in the future.

.

CHAPTER 2. BACKGROUND

Before proceeding further, it behooves us to closer examine the forensic process since it greatly impacts our work. It's essential to understand that any forensic software needs to satisfy the strict legal guidelines and procedures set in place by presenting any evidence to the investigator in an unbiased manner. This basically boils down to presenting the truth without making any conclusions or decisions for the investigator, allowing the legal process to judge something based only on the facts. For example, a piece of forensic software which suggests that certain files are suspect would be unethical. Thus, the following section briefly discusses the history of computer forensics and the established methodology to help establish that our methods are both unbiased and truthful. Additionally, we explore the motivation behind utilizing visualization as well as recording digital forensic data that records change-over-time.

2.1 Forensic Process

Computer forensics first started as a child of necessity for the purposes of law enforcement. Computers found on the scene of the crime were investigated by early practitioners who usually operated without academic education or any formal forensic training, relying only on their intrinsic knowledge of operating systems such as DOS, Windows and UNIX. Fewer still had any prior experience working in a structured computer forensics environment. It was only during the late 1980s that these early practitioners coined the term "computer forensics", using it to refer to the examination of stand-alone computers for digital evidence of crime. The term also eventually expanded to include the examination of information flowing over a network [8].

Diving further back for a moment however, the word "forensic" is derived from the Latin word *forensis*, meaning "of the forum". Dating back to the 17th century, the term is usually used to describe evidence relating to or dealing with the application of scientific knowledge to legal problems, suitable to courts of judicature or to public discussion and debate [9]. In the same vein, Kruse [2] emphasizes the importance of treating every case as if it will end up in court, while describing the process and why each step is essential to the investigation. The basic methodology of every forensic game plan consists of the following, which is oftentimes referred to as the three As:

- Acquire the evidence without altering or damaging the original.
- Authenticate that recovered evidence is the same as originally seized data.
- Analyze the data without modifying it.

While the three steps above form the framework for any investigation, the acquisition phase can be further broken down to the collection, identification, transportation and storage of evidence. In addition, the documentation of the investigation should also be taken into account, to help deter any allegations that the evidence was tampered with - a favorite tactic of defense attorneys. As such, each step of the forensic process is outlined in Figure 1 and discussed in further detail below.

2.1.1 Collection

Great care should be taken to respect the chain of custody during the handling and collection of evidence. Second nature to most law enforcement personnel, the goal of maintaining the chain of custody is to protect the integrity of evidence and to head off

accusations of tampering. This involves documenting the complete journey of a piece of evidence during the lifetime of a case, including who collected it, had possession of it, the time at which they took or returned possession of it, and the purpose of each retrieval. Furthermore, Kruse [2] advises to collect everything possible within the bounds of the law at the time of collection. A crucial piece of evidence could easily be misplaced or maliciously destroyed by the time the investigator decides to return. He cites log files in particular, since they can easily have been overwritten due to their sheer size, especially in cases involving Internet service providers (ISP).

2.1.2 Identification

Every item seized or taken from a crime scene should be carefully documented. An evidence custodian and at least one witness is usually designated during large investigations, ensuring that each piece of evidence (which could include computers, routers, hard drives and other storage media) is identified, labeled and documented with the case number, a brief description of its appearance as well as time stamps. This helps ensure that nothing is overlooked. Pictures are also taken to record the locations of each piece of evidence, as well as provide descriptions of the environment in which they were found.

2.1.3 Transportation

A majority of computer evidence is not made to be moved, and thus can be sensitive to environmental hazards such as shock, electromagnetic fields, and so on. Even mobile computing devices such as laptops can be damaged if not handled properly. As such, care should be taken ensure that preventive steps are taken so that nothing is damaged during

transport. Anti-static bags and bubble wrap are oftentimes used as protective materials to prevent shock. Securing hardware to prevent sliding or falling is also important. In addition, common sense measures such as keeping food or drinks away from the evidence during transport should be taken. Finally, authorized personnel should seal containers with tamper evident labels. A signature across the seal helps indicate that it was not opened by anyone other than an authorized person.

2.1.4 Storage

Once at the crime lab, it is important to store the evidence in a cool, dry environment appropriate for electronic equipment. In addition, it should be stored in a secure area with limited access. Each lab usually has such a facility, typically known as an evidence locker. Again, common sense dictates that the same precautions taken during transport be taken in this facility as well, with the appropriate protective materials being used and harmful substances being kept at arm's length. And as mentioned earlier, the chain of custody should be strictly adhered to in this situation as well, with names, dates and times associated with each instance in which any piece of evidence is handled. Additionally, Kruse [2] suggests limiting the number of persons with access to one primary custodian and one alternate.

2.1.5 Authentication

Unlike traditional kinds of evidence, which are susceptible to adverse environmental conditions such as mold, dust or insects, digital evidence enjoys an advantage in that it is easily possible to provide proof of intergrity as well as timestamping by calculating a value that functions as a unique signature for any file. This value is known as a hash, and is in turn

generated by an algorithm known as a cryptographic hash. Hash values are usually taken at the outset of data collection, and again after the analysis phase of the investigation. Since even minute alterations to a file would result in a different hash, it helps proves that a file was not tampered with in any way during the process of investigation. Two of the most common hashing algorithms currently in use are MD5 (Message Digest) and SHA (Secure Hashing Algorithm), with SHA being the official algorithm in use by U.S. Government agencies.

2.1.6 Analysis

As a further precaution, when a piece of evidence is checked out of the locker for evidence discovery, a bit-for-bit (also known as a bit stream) clone of the original drive should be made. This forensic backup is commonly referred to as an 'image'. An important distinction to make is that a "normal" backup does not contain an exact copy of the original drive, since it only contains the logical portions and doesn't include deleted files, swap files, slack space, boot records and so on. A forensic backup on the other hand, does. In addition, a forensic backup helps ensure that any actions taken by the investigator does not inadvertently modify or destroy the original data.

As mentioned above, a process known as evidence discovery then takes place. An investigator can often develop a sense of the suspect's technical prowess and capabilities based on the software and operating system found. For example, if only standard software appears to have been used, the suspect might only be using less sophisticated methods to conceal evidence. The next step is more art then science, with the investigator searching for keywords based on the case being investigated. For example, in a case dealing with drug dealing, terms such as mary jane, marijuana, money and dealer might be appropriate. String

searches are an effective way to quickly and efficiently canvas the hard drive to identify files of interest. Finally, these files will oftentimes need to be more closely examined with a hex editor.

2.1.7 Documentation

Much like the rest of the forensic process, proper documentation is crucial during the analysis phase of the investigation. Opening folders, examining files and any other action taken should be recorded, along with the forensic tool used, the time at which it took place, and the investigator involved. At any point during the investigation, it should be clear which individual carried out a task, and why it was done. This is essential, since the success of a court case is based on the investigator's ability to clearly explain the relevance and manner in which a piece of digital evidence was found, keeping in mind that the audience might not be as technologically inclined. In addition, it is important to note that some forensic tools are more credible in court than others due to their proven track records. Two such tools at the moment are EnCase [3] by Guidance Software and Forensic Toolkit (FTK) [4] by AccessData.

2.2 Impetus for Data Visualization

Data visualization is the interdisciplinary study of "the visual representation of large-scale collections of non-numerical information, such as files and lines of code in software systems, library and bibliographic databases, networks of relations on the internet, and so forth" [10]. It is a valuable tool, due to the fact that "humans have the ability to visually interpret and comprehend pictures, video, and charts much faster than reading a textual description of the same". Quite simply, we are relying on the old adage "A picture is worth a

thousand words" [11]. Case in point, imagine attempting to describe the beauty of a natural vista to someone else, instead of simply showing them a picture. One would have to spend an inordinate amount of time describing each little element to convey the same amount of detail they'd achieve by simply handing over the picture.

Figure 1 : A picture of the Forbidden City compared to an incomplete textual description of the same vista.

In fact, Erbacher and Teerlink conducted a set of user experiments in which subjects were asked to search a hard drive for altered and hidden files using two methods - traditional Linux-based shell commands versus developed visualization techniques. The result was a 53% increase in files located and a 35% reduction in time utilizing the visualization techniques. Moreover, a renamed JPEG file hidden in a vast shared library was easily located by many of the subjects using the visualization technique, whereas none of

those using traditional Linux-based shell commands managed to locate it. While this was not meant to be a complete evaluation, it does emphasize the enormous value and effectiveness of visualization [11].

These user experiments also serve to highlight several reasons to further explore visualization - namely, increasing the perceptual bandwidth by which investigators perceive forensic data, as compared to perceiving text serially [11]. This in turn speeds up forensic analysis and the productivity of investigators. In addition, as highlighted by the subjects using the visualization technique locating the hidden JPEG file, information visualization allows users to both more easily discover unique relationships between data as well as more easily comprehend it in terms of its global context (i.e. its place in the hierarchy) [12]. In short, this allows the investigator to perform a more thorough analysis of the data as an added bonus.

2.3 Digital Forensic Data that Records Change-Over-Time

Time is an illusion that is created by the mind's response to its perception of a world of continuous change. Time and change share a common quality that is often expressed as the single concept of "change-over-time". For the digital forensic examiner, being able to perceive change-over-time supports the goal of being able to explain "what happened?"

Certain types of digital forensic data record a history of change-over-time. One such example are log-based file systems such as YAFFS (Yet Another Flash File System), LogFS and Fossil maintains files in a circular log. When a file is modified, it is copied into memory before it is changed and written into the log. Thus, several versions of the same file is maintained, allowing for a comparison of both older and newer versions to paint a picture of change-over-time.

The application that we are utilizing for our purposes here is the Volume Shadow Copy Service (VSS). First introduced in Microsoft Windows XP, and included in every subsequent version of Windows, it allows users to take manual or automatic backup copies or snapshots of data, even if it has a lock, over regular intervals. This in turn allows one to restore individual files or even entire disk partitions to a previous state, allowing for the recovery of data that has been accidentally deleted, modified, or otherwise corrupted.

CHAPTER 3. OBJECTIVES OF THIS WORK

With computer forensics being a relatively new discipline, there is a lack of complex evidence analysis methods that is intuitive towards the investigator. The majority of commercially available forensic software tools such as EnCase [3] and AccessData's Forensic Toolkit (FTK) [4] simply presents information on the file system and the files found within in a textual format. With storage media rapidly dropping in cost per gigabyte [5], investigators are forced to spend increasingly longer amounts of time to sort and locate all the relevant information pertaining to a criminal case [6].

Our goal for this project is to research, design and develop a forensic data visualization and management system capable of the following requirements:

- Easily conveys to the user that a change has occurred in a file or directory.
- Able to convey how a directory or file has changed over time.
- Preserves the global context of the directory.
- Intuitive and user-friendly.

The challenge here lies in displaying the enormous amount of data with a limited amount of computer screen real estate while meeting the requirements above. To this end, we researched several focus+context visualization approaches not normally applied to forensic data analysis, which we will get into in a later section. It is our hypothesis that this would greatly reduce the amount of tedium for forensic examiners to locate files of interest, as well as allow for a more thorough analysis of the provided data.

CHAPTER 4. AN ARCHITECTURAL OVERVIEW OF FishEYE

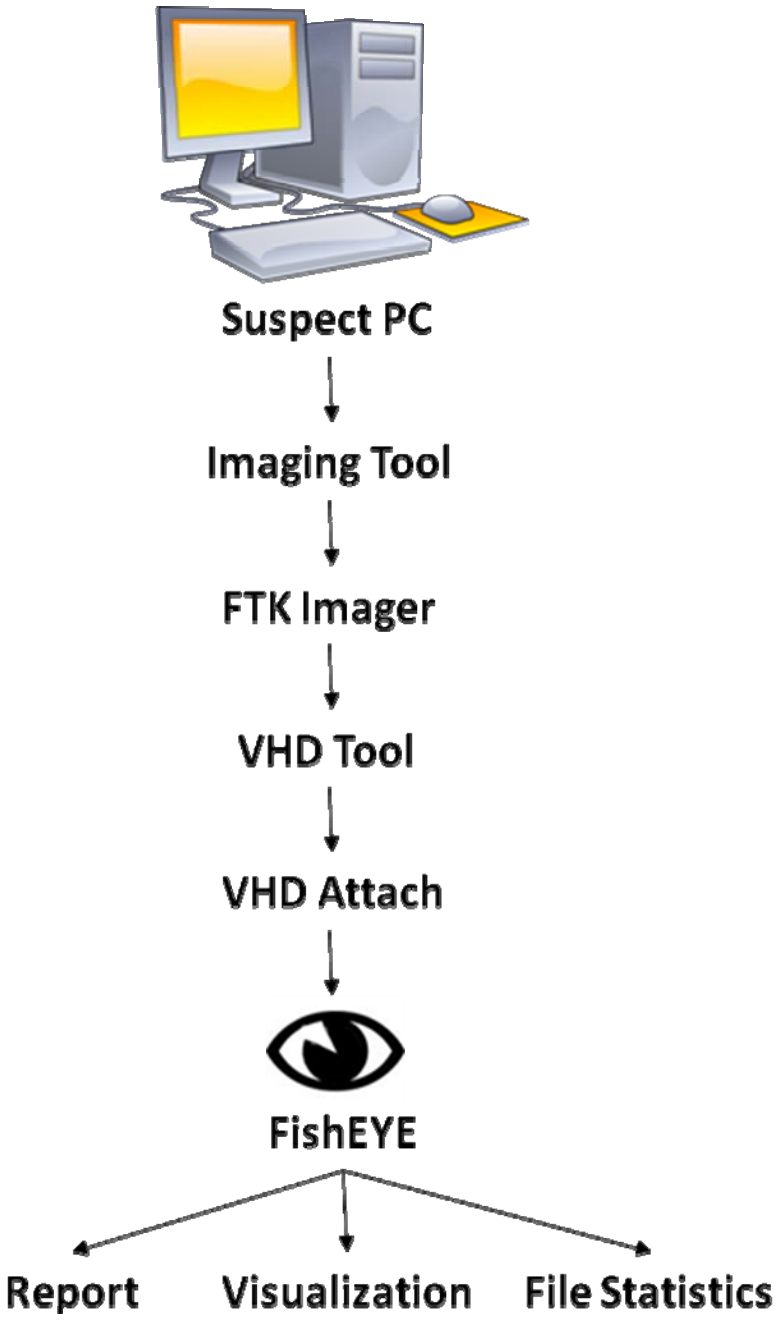

Figure 2 : An architectural overview of FishEYE, from the suspect PC to the final 'products' generated by FishEYE.

In this chapter, we provide a brief overview of the operation of FishEYE, beginning with the suspect PC and tracing each step till the final 'products' generated by our FishEYE tool.

Following the steps of the forensic process we discussed earlier in Chapter 2, the suspect physical drive is collected, identified, transported and securely stored before being imaged by an imaging tool. Utilizing this process, which attempts to use only freeware wherever possible, four formats are supported - Advanced Forensics Format (AFF), SMART, E01 (EnCase image file format) and the RAW image format (dd). This allows for the support of a vast of majority of forensic imaging tools, including EnCase, SMART, Sleuthkit, Autopsy, FTK Imager and so on.

Once an image has been acquired via any of the supported imaging tools, FTK Imager [13] (which is available for free) is used to convert it into the RAW image format. This is subsequently fed into VHD Tool [14], a free tool which converts RAW disk image files into a fixed-format VHD (Virtual Hard Disk). This VHD is then attached using the VHD Attach utility [15], which allows VHDs to attached and detached without a trip to the Disk Management console. As a quick aside, the VHD Attach tool is only supported for Windows 7 and above - investigators using Windows XP will have to manually attach the VHD from the Disk Management console.

From there, our FishEYE tool can be used to analyze the attached VHD. The most obvious and important result is the visualization of change-over-time in a directory-tree structure, an example of which can be found later in Chapter 8, and which we will explore further later. In addition, several options for further analyzing the tree structure and generating file statistics are available, including finding recently modified files, generating a list of file types, and so on. A full list of statistics can be found under Chapter 6. Finally, a

fully featured word processor is included in the right panel of the main screen for the investigator to take notes and generate a report based on the information gleaned. All the basic features one would expect such as changing font size, color, type, paragraph justification, inserting images and a find and replace function are all easily accessible via the toolbar located above the work space or via the Report menu item in the menu bar. In addition, the File menu item has functions to create a new report, save it, open a previously saved report or even printing it.

CHAPTER 5. OUR FISHEYE DESIGN

As we established earlier, data visualization can be a valuable tool in both providing a broader context to the forensic examiner as well as speeding up the forensic process considerably by reducing the time to identify suspicious files and increasing the probability of locating criminal evidence. Thus, it seems only prudent that we delve a little further into the topic here, examining the various visualization schemes in general, as well as explore a multitude of visualization approaches, each of which we will list and elaborate upon below. Finally, we will explain why we decided on using a fisheye view combined with a segmented change-over-time box of our own creation.

5.1 Visualization Schemes

Data visualization schemes can be sorted into two broad general categories, non-hierarchical and hierarchical schemes. Here, we take some time to explore both to determine which is more suitable for our purposes.

5.1.1 Non-Hierarchical

This category encompasses those visualization approaches which presents all data points of interest without any consideration to the relationship between each individual data point. To draw a comparison, it would be similar to displaying all files and subdirectories within a directory without any consideration being given to the relationship between each file. A simple example of such a scheme would be to have different shaded blocks represent file sizes. Larger files would be lighter colored, while smaller files would be darker colored, as shown below. This would allow an examiner to easily spot larger files within a directory,

as it would stand out from the other darker blocks. Of course, this scheme could easily be extended to filter the time files were created, modified or accessed, allowing an examiner to more easily distinguish files that show more recent activity as compared to files that have not been modified in months or even years. This could ostensibly help investigators identify files that are more likely to be relevant to their investigation [11].

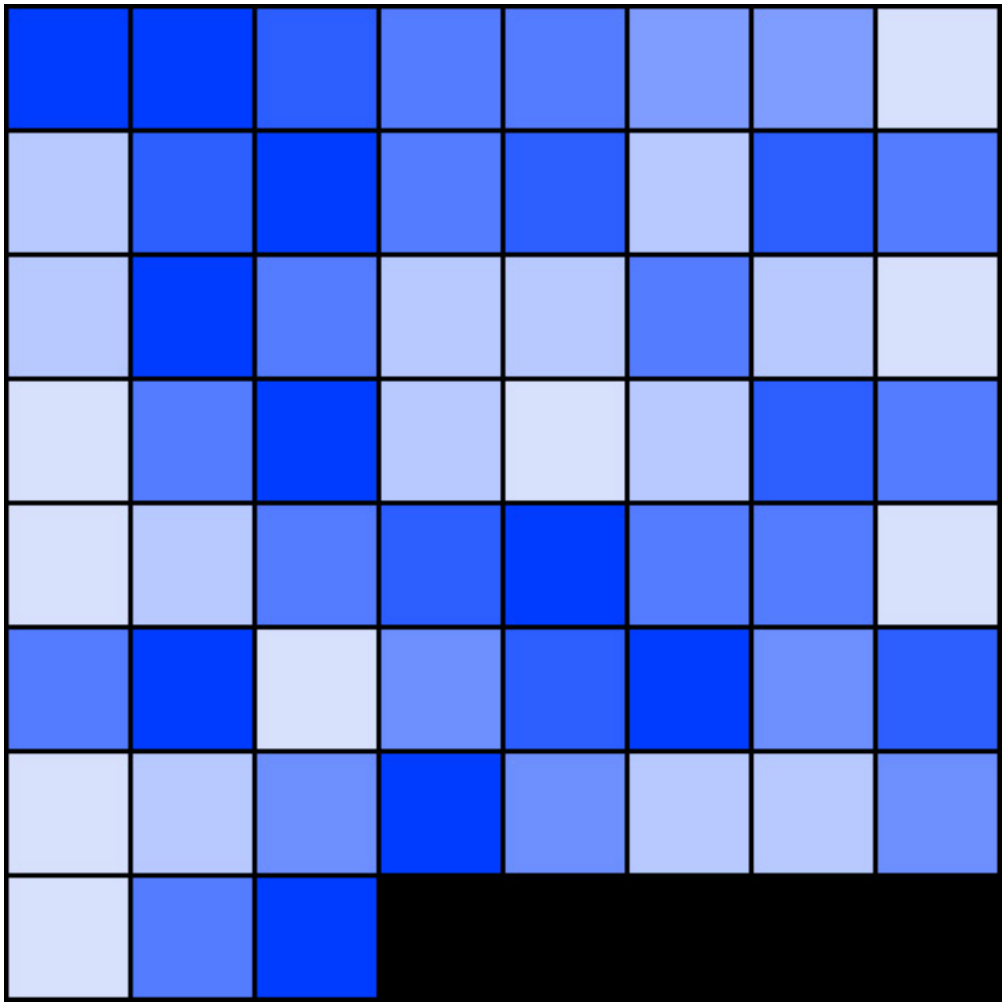

Figure 3 : An example of a square block diagram depicting the file sizes of files within a directory. The darker colored blocks represent smaller files while the lighter colored blocks depict larger files.

5.1.2 Hierarchical

As opposed to non-hierarchical schemes, hierarchical visualization schemes presents data points while still maintaining the relationship between each. For a directory tree system, this would translate to the relationship between directories, subdirectories and files being preserved. A simple hierarchical scheme is illustrated in the figure below as a basic tree diagram. Note that the tree diagram is only being presented as a basic example, and wouldn't be suited to representing an actual file system which would contain far too many files to display easily. Naturally, this category of visualization scheme most naturally lends itself to our particular problem, allowing us to present information graphically at a glance while still preserving the relationship between files [11].

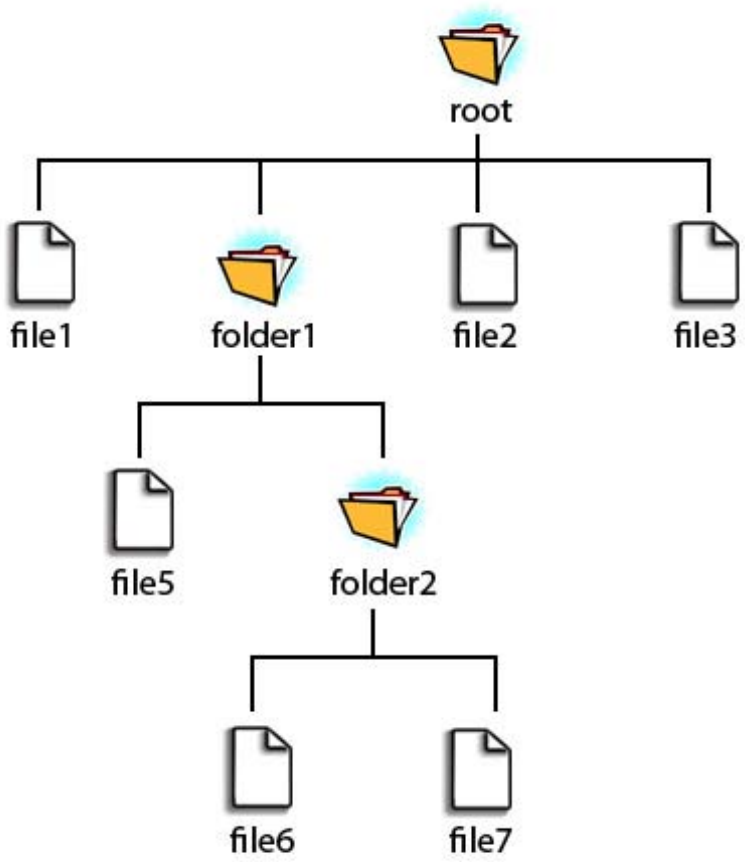

Figure 4 : An example of a hierarchical scheme. A simple tree diagram which preserves the relationship between files

5.2 Visualization Approaches

To approach this problem, we explored several classical focus+context visualization approaches. The basic idea behind these approaches is to show the selected region in greater detail (focus), while preserving the global view at reduced detail (context) [16]. Hence, viewers should be able to see the object of primary interest presented in full detail while at the same time getting an overview (or impression of all the surrounding information)

available. For each of the approaches explored below, we explain why they aren't suitable for our purposes (i.e. the visualization of a directory tree structure).

5.2.1 Perspective Wall

The perspective wall visualizes linear information by smoothly integrating detailed and contextual views. This technique takes advantage of hardware support for 3D interactive animation to imitate the architecture of the eye system. It folds a 2D layout into a 3D wall that smooth integrates a region for viewing details with perspective regions for viewing context. The middle panel presents a detailed representation of the selected region, while the side panels project the remainder of the data in lesser detail. This approach supports efficient use of space [17].

The perspective wall is unsuitable for our needs since it only works with linear data sets. It doesn't work for our directory structure data set that extends in both directions. In addition, it distorts the global context to too great a degree for our directory tree structure.

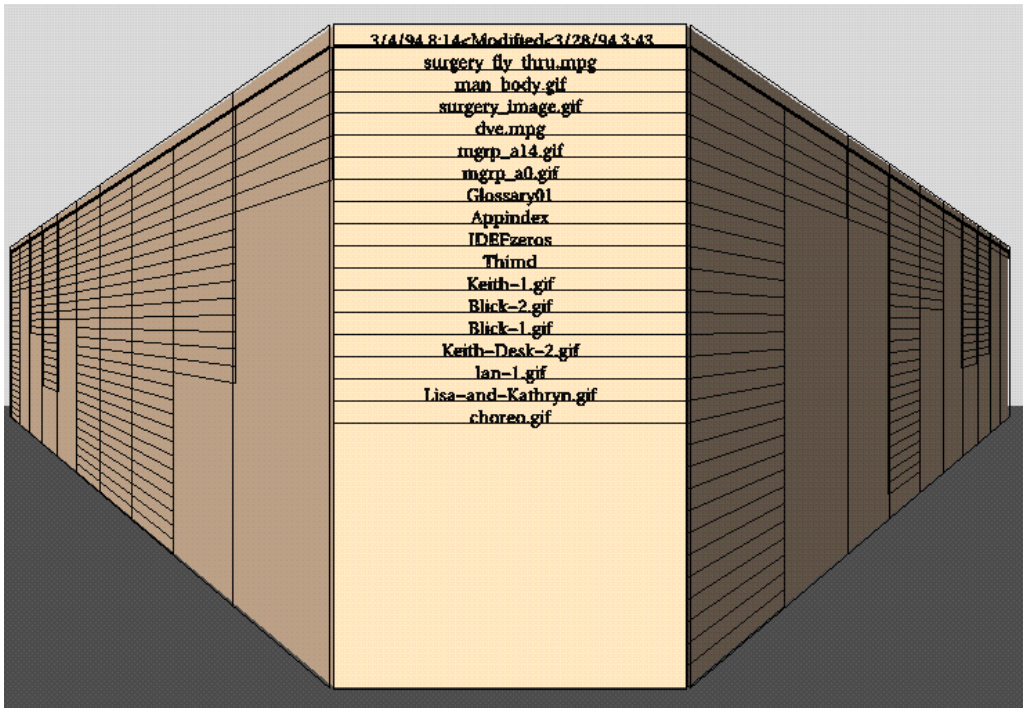

Figure 5 : An example of a Perspective Wall. The middle panel presents more detail on the region of interest, while the rest is folded to the side panels [22].

5.2.2 Bifocal Display

The bifocal display is in many ways similar to the perspective wall. It arranges data into three vertical strips instead of walls. The middle strip presents a detailed representation of the selected region, while the side strips project the remainder of the data. The side strips are 'squashed' and distorted to draw attention to the middle panel. While the squashed view may not allow much detail to be discerned, but with appropriate color and/or positional encoding, both the presence and nature of items outside the focus region can be interpreted [18].

Unfortunately, the bifocal display is not suited to our needs as well. Like the perspective wall, it too distorts the global context of our directory tree structure, which can lead to either incorrect analysis or conclusions.

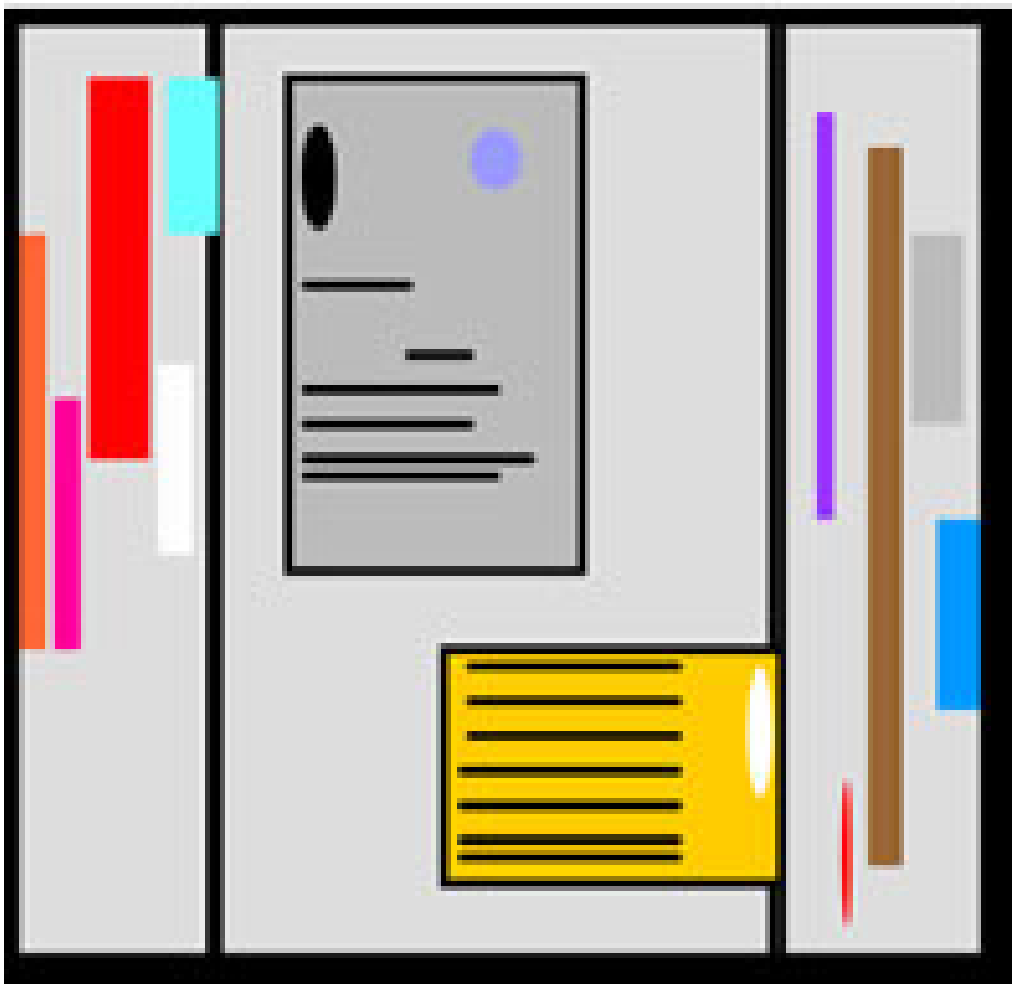

Figure 6 : An example of a Bifocal Display. The middle strip presents more detail on the region of interest, while the rest is squashed into the side strips to focus attention on the middle strip [18].

5.2.3 Document Lens

Similar to the bifocal display and the perspective wall, the document lens presents the data of interest in the middle of the screen, in a rectangular focal box. Unlike the previous approaches however, the document lens utilizes space more efficiently by occupying the areas above and below the middle box, allowing the user to scroll up and

down as well as left and right. It is usually used for domains that are rectangular, such as text documents [19].

The document lens was judged unsuitable for our particular needs since it is harder for the viewer to intuitively comprehend the global context of the data set at a glance when expanding in four directions.

Figure 7 : An example of a Document Lens. It is more efficient in terms of space usage compared to the Perspective Wall and Bifocal Display since it utilizes the areas above and below the document of interest as well [19].

5.2.4 Tree-Maps

Making use of nearly all available screen space, tree-maps are a rectangular, space-filling approach for visualizing hierarchical data. They use 2D visualization of trees where the tree nodes are encapsulated into the area of their parent node. The size of the single nodes is determined proportionally in relation to all other nodes of the hierarchy by an

attribute of the node [20]. This efficient approach thus allows for the display of enormous amounts of data. Combined with its inherent design meant for hierarchical data that is arranged in trees, this approach seems at first glance to be perfect for our needs.

However, although this approach easily and clearly displays the data within each directory node, the actual structure of the hierarchical data is sadly obscured. This makes it unsuitable to our particular problem.

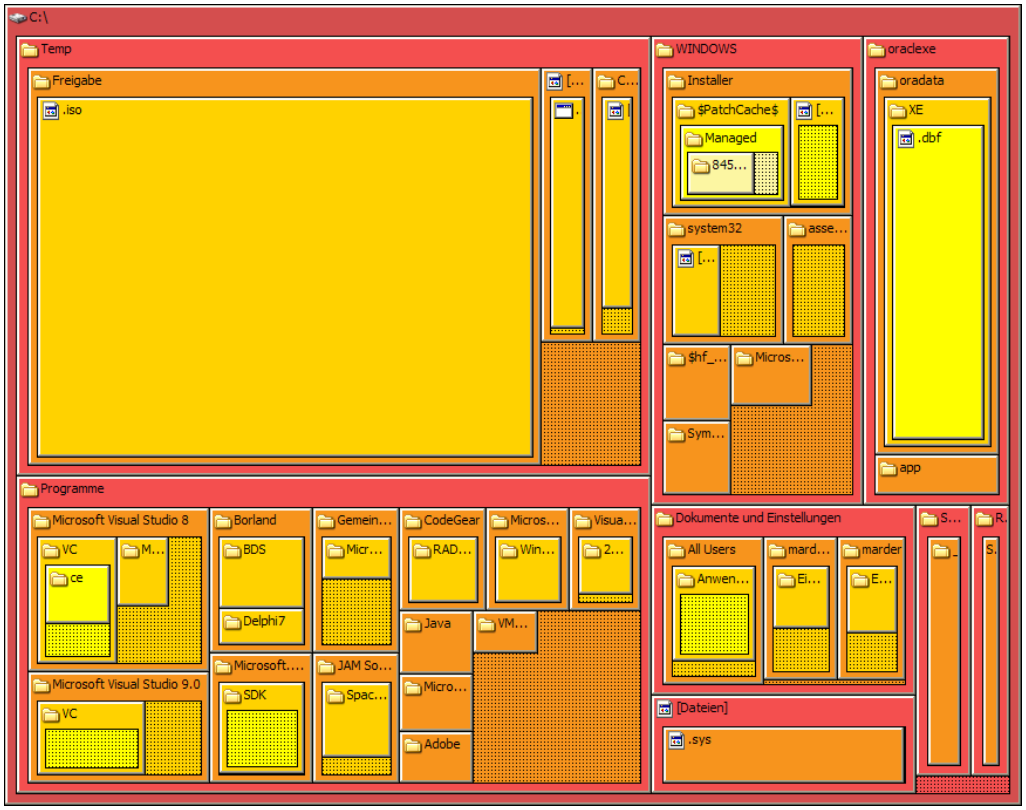

Figure 8 : An example of a Tree-Map. One of the most efficient ways of displaying enormous amounts of hierarchical data.

5.2.5 Hyperbolic Browser

The essence of the Hyperbolic Brower is to lay out the hierarchy in a uniform way on a hyperbolic plane and map this plane onto a circular display region. This supports a smooth blending between focus and context, as well as continuous redirection of the focus. Two salient properties of the figures are: First that components diminish in size as they move outwards and second that there is an exponential growth in the number of components [21].

Unfortunately, while again this approach seems naturally suited to our needs, the scattering of data caused by this approach makes it difficult to display change-over-time information for our directory-tree structure, and thus, unsuitable.

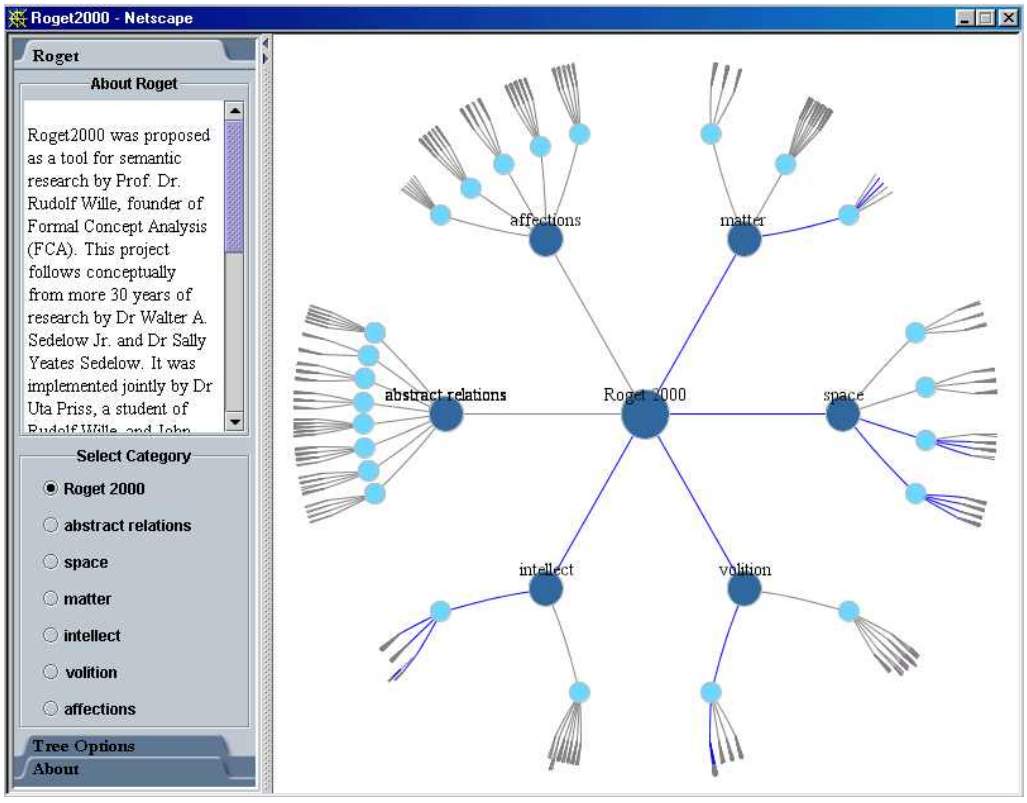

Figure 9 : An example of a Hyperbolic Browser.

5.2.6 Fisheye View

Graphs with hundreds of vertices and edges are common in many application areas of computer science. Displaying all this information on a screen has the disadvantage of losing the details. Alternatively, zooming into a part of the graph does show local details but loses the overall structure of the graph. Displaying two views - one view of the entire graph and one of a zoomed portion of it - has the advantage of seeing both local detail and global structure but the drawback of requiring extra screen space and forcing the viewer to mentally integrate the views. A fisheye view of a graph shows an area of interest quite large while the rest of the graph remains smaller and in less detail. The viewer can position the mouse to the area of interest of the graph and determine the focus point in that way. The appearance of the fisheye view can be varied by changing the distortion factor of the lens. With the distance from the focus and a number, which is assigned to each vertex to represent its relative importance in the global structure, the visual worth for each vertex is computed. By means of the parameter "visual worth cutoff" the depth of details displayed on the screen can be adjusted [23].

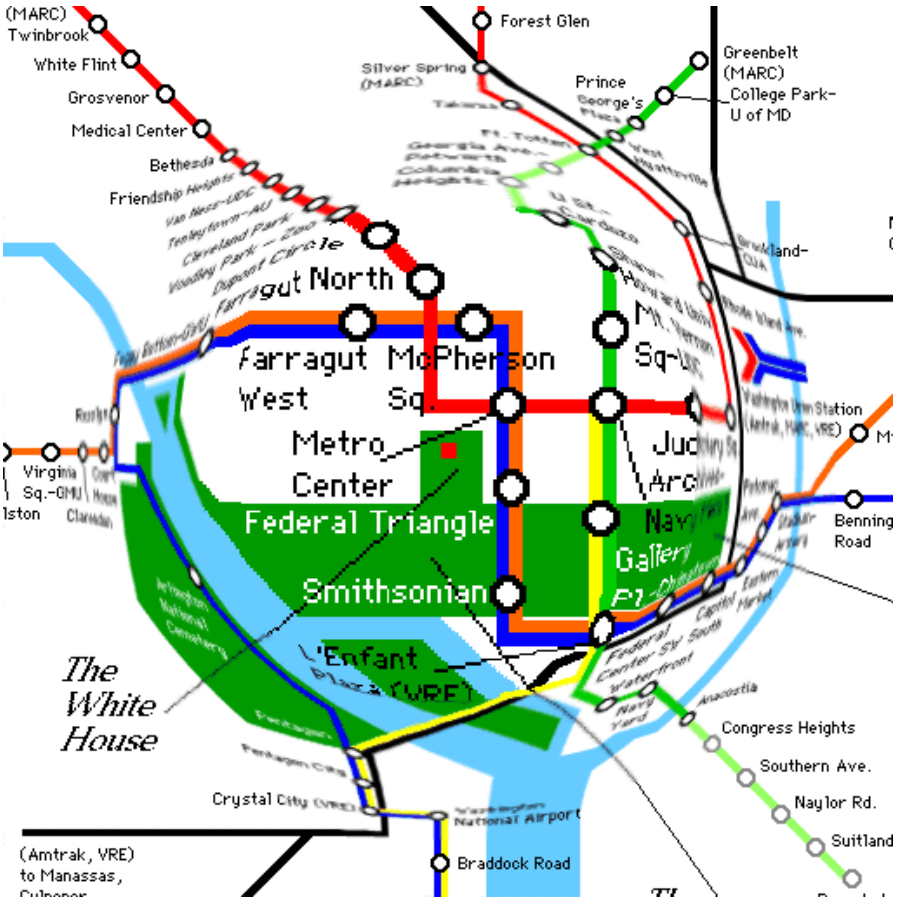

Figure 10 : An example of a fisheye view used on a metro map.

Since this approach does not distort data (can be eliminated by completely removing data beyond a certain threshold), works well with hierarchical data, expresses both content and context information, and can be applied to large data sets, the fisheye view was deemed suitable for our needs. We thus adapted it for our directory-tree structure. A default threshold limit of 2 was set for each level of the directory being expanded, thus displaying detailed information for folders within 2 spaces. A '...' entry denotes the fact that folders (i.e. information) is being hidden. Finally, a rectangular focal box highlighting the information being explored is displayed as well, allowing the viewer to keep track of what information is being investigated.

5.3 Our Design

Here we examine the our own contributions, including a segmented change-over-time box of our own design, and its combination with a fisheye view.

5.3.1 Segmented Change-Over-Time Box

To easily convey how a file or directory has changed over time to the user, we have created visualization icons we dubbed segmented change-over-time boxes to represent a file system directory. Each segmented box is a rectangle which is evenly divided into segments, with each representing a time period ordered from left-to-right (i.e. from the oldest to the most recent version). Each segment is color coded to represent one of three states. Red means that a particular directory/file did not exist at that point in time, yellow means that there was no change to that directory/file, and green means a change has occurred since the last time period to that particular directory/file.

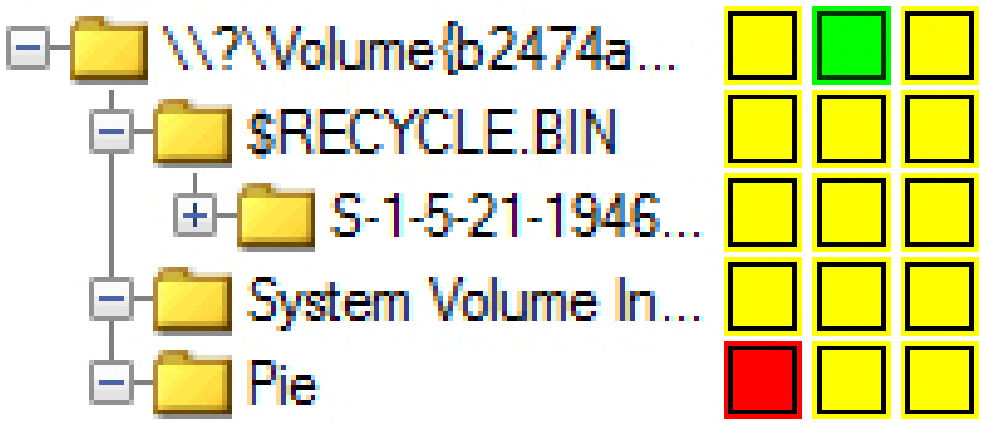

Figure 11 : A closer look at the segmented change-over-time box from above.

Our use of red, yellow and green to color-code and denote the state of a directory/file is an effective means of layering information [12]. The intention of this technique is to allow the viewer to logically identify and focus on each subset of data, while clearly separating each segment to help the viewer identify where change has taken place. The choice of colors was based on color symbolism, drawing on common everyday experiences - in this case, a traffic light. Red is used to represent non-existence since in everyday experience, a red traffic light represents "STOP", i.e. a warning. Yellow, being in between red and green, is used to represent a lack of change. And finally, green represents change since a green traffic light represents "GO", i.e. a symbol of a change of state [24].

To represent the parent-child relationship of a directory, we defer back to the tree view utilized in Windows Explorer, with lines being drawn to connect the directories and child directories being indented under the parent directory. The figure below shows an example of the segmented change-over-time box that are used to represent a directory-tree structure that have parent-child relationships. In this example, the Pie directory is indented below the Volume's root directory, indicating the parent-child relationship. Furthermore, note the red-colored first segment for the Pie directory, indicating that it did not exist during the first time period. With its creation during the second period, the root Volume directory is colored green, to indicate that a change has occurred.

The creation of the segmented change-over-time box checks off one of our primary goals, which is to portray how a directory-tree structure changes over time in a visual medium. The second of these goals is to present this information within a global context, or to allow the user to trace the root of a directory or file at a glance.

5.3.2 Combination of Fisheye View and Change-Over-Time Box

There are two possible ways we could have gone to apply the fisheye view to our change-over-time box. The first way would involve reducing the height of the boxes less relevant (i.e. further away from) to the directory under investigation. This would be similar to non-linear magnification [25]. Note that the width of the boxes would not be reduced so that each segment representing a time period still remain vertically aligned, allowing for logical association by the viewer. The other way, which by process of elimination is the method we chose to pursue, is to remove entire change-over-time boxes completely. As we discussed in an earlier section, this approach helps eliminate the problem of distorting data and unnecessarily confusing the viewer [23]. Thus, in our implementation, only the two closest directories in either 'direction' of the target directory are displayed and absent directories are represented by the '...' symbol. In addition, we recognize that the needs of our users can't be known ahead of time. Thus, in a future revision of our program, we intend to include a slider for the user to adjust the number of directories shown adjacent to the target directory. They will thus be able to adjust the amount of global context to better suit their needs, for instance increasing it to support a more general search, or inversely decreasing it for a more focused investigation.

CHAPTER 6. FILE STATISTICS

As an initial foray into the field of data mining involving the information our program is capable of uncovering, we decided to include several small sub-utilities to display statistics involving information recovered from each file during various time periods. This conversely doubles as a demonstration of what the framework we've built here is capable of achieving, with the potential for future methods to perform further data mining and revealing more regarding the habits and actions of the subject under investigation. The algorithms used in generating these statistics are included as well. Below is a general algorithm run before any of the specific sub-utilities are run to generate a list of stats for access, modified and creation times.

```
foreach (FileLink f in root.files)
{
	newStat = new FileStats();
	newStat.name = f.name;
	newStat.creationTime = (f.file.creationTime);

	// Create a new FileStats object for each file.
	for (int i = 0; i < f.snapShotTimes.size; i++)
	{
		if(f.file.modifiedTime[i] != null)
		{
			if (newStat.modifiedTimes != 0 && newStat.modifiedTimes[i-1]
!= f.file.modifiedTime[i])
				newStat.modifiedTimes.Add(f.file.modifiedTime[i]);
				newStat.numModified++;
			else if (newStat.modifiedTime == 0)
				newStat.modifiedTimes.Add(f.file.modifiedTime[i]);
				newStat.numModified++;
		}

		if(f.file.accessedTime[i] != null)
```

```
                {
                        if (newStat.accessedTimes != 0 && newStat.accessedTimes[i-1]
!= f.file.accessedTime[i])
                                newStat.accessedTimes.Add(f.file.accessedTime[i]);
                                newStat.numAccessed++;
                        else if (newStat.accessedTime == 0)
                                newStat.accessedTimes.Add(f.file.accessedTime[i]);
                                newStat.numAccessed++;
                }

        }

        stats.Add(newStat);
}
```

6.1 File Types

```
foreach (FileLink f in root.files)
{
        extensionExists = false;

        // Check if extension type already exists in FileStats. If it does, add to the count.
        foreach (FileStats s in stats)
        {
                if (f.extension == s.name)
                {
                        s.count++;
                        extensionExists = true;
                        break;
                }
        }

        // Create a new FileStats object for a new extension.
        if (extensionExists == false)
        {
                newStat = new FileStats();
                newStat.name = f.extension;
                newStat.count++;
                stats.Add(newStat);
        }
}
```

The first of our sub-utilities is a simple file type statistics generator. In the figure below, a list of file types, the number of each and the percentage out of the total being explored is shown.

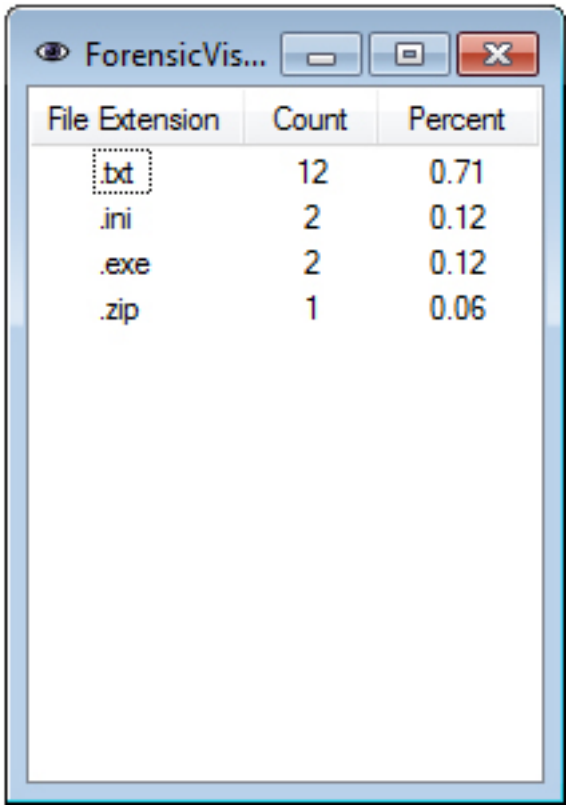

Figure 12 : The file statistics window generated for a particular data set.

6.2 Most Frequently Modified Files

This sub-utility checks the last modified time property for each file and displays the top ten most frequently modified files in the directories being investigated.

6.3 Recently Modified Files

This sub-utility checks the last modified time property for each file and displays the recently modified files within a certain period of time specified by the user from the time stamp of the last snapshot.

6.4 Modified Files

This sub-utility checks the last modified time property for each file and displays the files that were modified within a time period specified by the user.

6.5 Most Frequently Accessed Files

This sub-utility checks the last accessed time property for each file and displays the top ten most frequently accessed files in the directories being investigated.

6.6 Recently Accessed Files

This sub-utility checks the last accessed time property for each file and displays the recently accessed files within a certain period of time specified by the user from the time stamp of the last snapshot.

6.7 Accessed Files

This sub-utility checks the last accessed time property for each file and displays the files that were accessed within a time period specified by the user.

6.8 Recently Created Files

This sub-utility checks the created time property for each file and displays the recently created files within a certain period of time specified by the user from the time stamp of the last snapshot.

6.9 Created Files

This sub-utility checks the created time property for each file and displays the files that were created within a time period specified by the user.

CHAPTER 7. PROTOTYPE IMPLEMENTATION AND EVALUATION

As our program deals with the visualization of forensic information and utilizes the fish eye methodology, we naturally named it 'FishEYE'. Having clarified our goals, our motivations to achieve them, several classic focus+context approaches earlier, and finally the approach we decided upon, we now delve into the nitty-gritty details of their implementation.

7.1 Software

Here, we take a closer look at the architecture and implementation of our program, providing a look at the data objects used.

.

7.1.1 Capabilities

Our use of the fisheye view combined with the segmented change-over-time box in computer forensics has never been examined thus far. Our system not only uses visualization to represent a file system, but is designed with the forensic process in mind, the goal of which is usually to detect files that have changed and are thus suspect. Identifying altered system files can help the investigator know how to further direct his search for evidence. It is fully interactive and sensitive to mouse clicks. Clicking on each individual segment displays the details of that file during that particular period in the bottom center panel, including file type, file name, access time, last modified time and creation time. Double clicking opens up that file in the default program associated with that file type in Windows. Hovering over each column header also displays the exact time that the snapshot was taken.

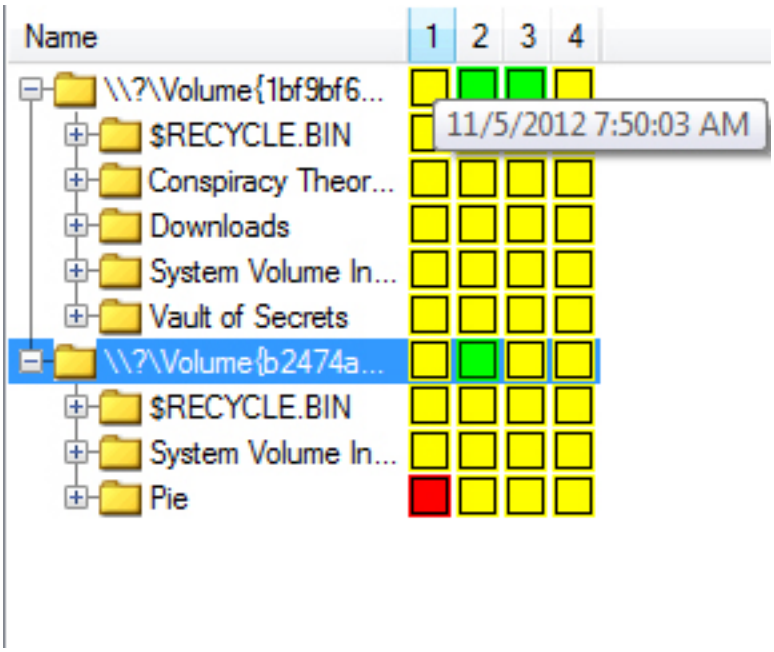

Figure 13 : Hovering over each column header displays the time a snapshot was taken.

7.1.2 Architecture

Naturally, Windows was selected as the operating platform for our forensic software mainly because of its native support for the Windows Volume Shadow Copy Service (VSS). This provides a convenient source of directory-tree information which changes-over-time with a standardized and relatively well-documented interface. However, using VSS on the .NET platform in C# is problematic. The reasons are somewhat unclear, but it seems to have to do with there being COM interfaces without an IID, and also that several interfaces of the VSS AP is not actually COM interfaces but rather C++ interfaces. This means there is no type library available for importing in a .NET application. The only viable solution then would be to write a custom wrapper in managed C++/CLI, that provides a managed interface to the VSS API. The effort involved in dealing with the sheer number of interfaces,

structures and functions in the Volume Shadow Copy API however would have made writing a complete wrapper an unattractive undertaking. To further muddy the waters, there are multiple versions of VSS depending on which version of Windows you're running. Luckily, we found the AlphaVSS library, a .NET class library providing a managed API for the Volume Shadow Copy Service [26].

Figure 13 shows a high level view of the different components in our system and how they are connected together. ShadowControl is just a simple control method for calling each of the sub-classes. The ShadowExtractor method reads volume information for each shadow copy, and arranges the copies for the same volume in a VolumeList data structure based on the date of each snapshot. This is followed by the RootExtractor method, which extracts the root of each copy and recursively builds up the tree in a series of DirectoryLink and FileLink data structures. Next, the PaddingCode method goes through the constructed trees to pad out directories and files that did not exist during a certain period. This is to help prepare it for visual display later in the TreeListView. Finally, the ColoringTree method is responsible for determining the 'color' for each file or directory depending on what changes occurred in the previous timestamp. Figures 14 to 16 showcase the members of each data structure. The criteria for evaluating change will be discussed in greater detail in the algorithms section.

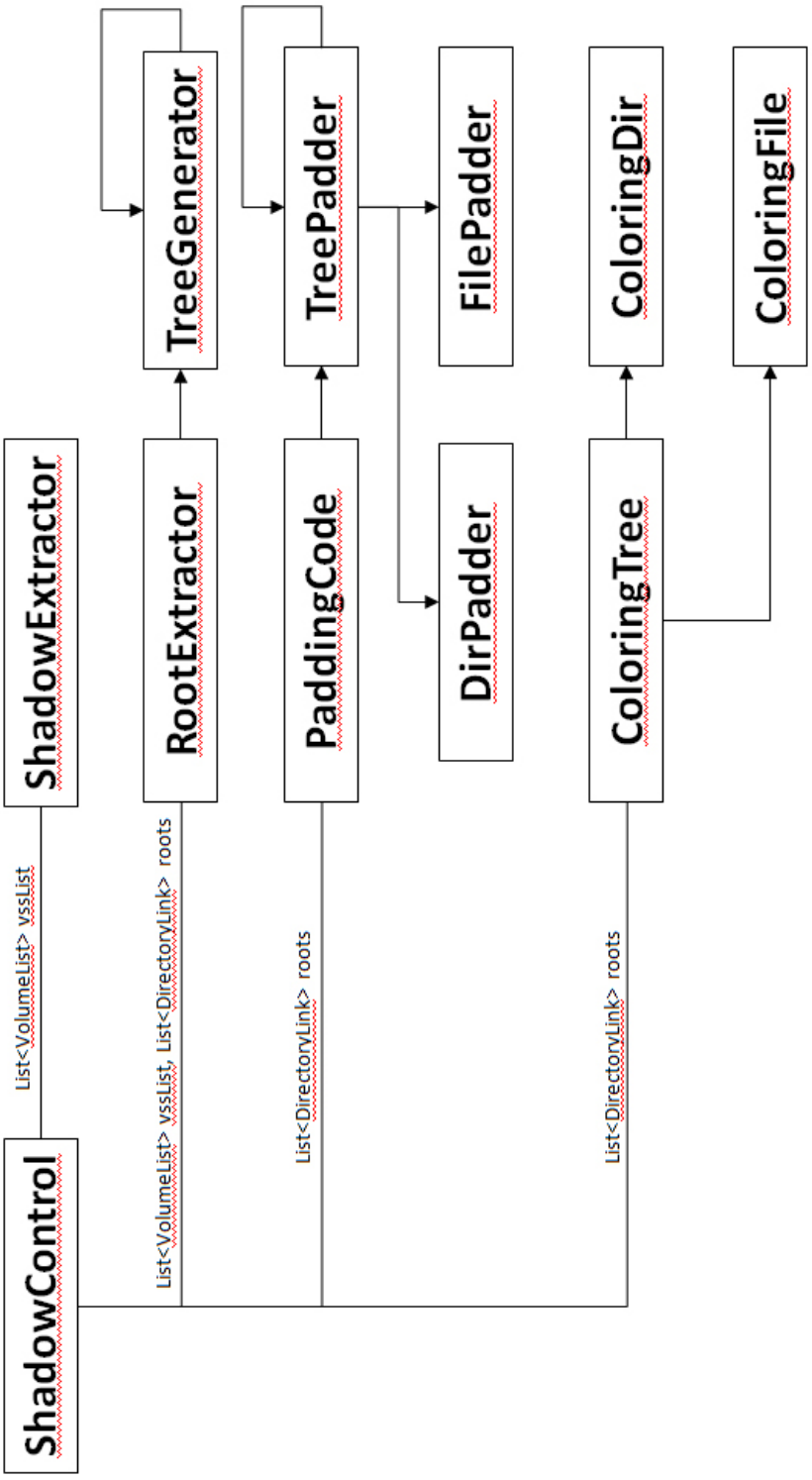

Figure 14 : High-level view of the VssUtilities class, which is where the shadow copies are organized before being visualized.

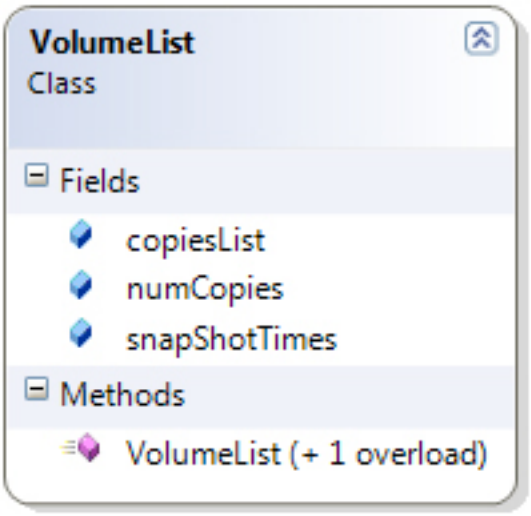

Figure 15 : Fields and methods for the VolumeList data structure.

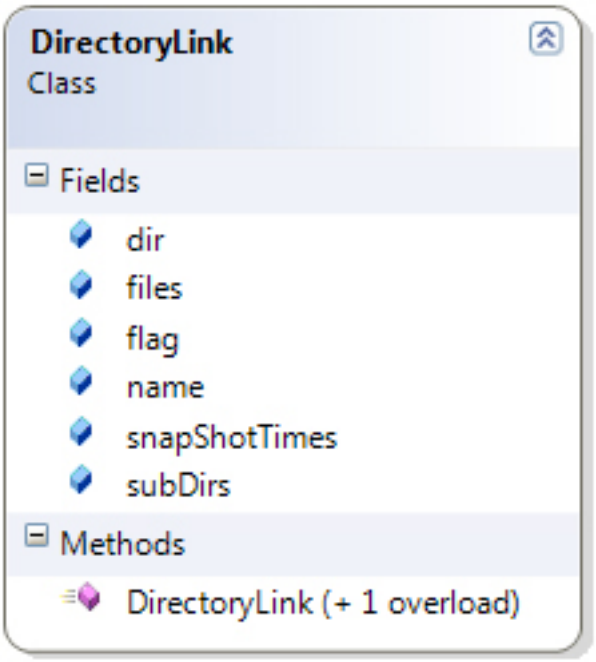

Figure 16 : Fields and methods for the DirectoryLink data structure.

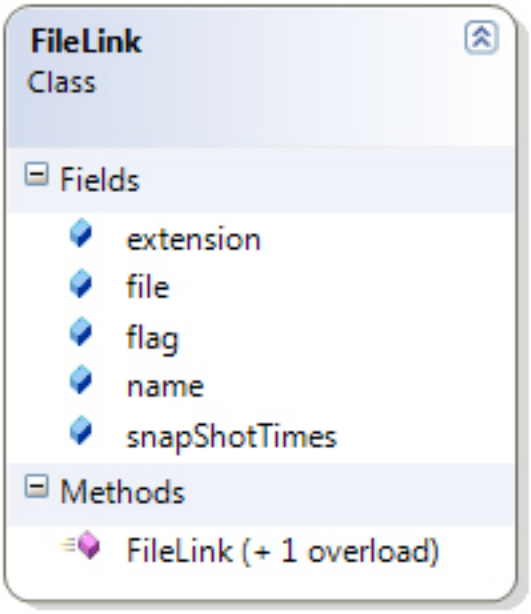

Figure 17 : Fields and methods for the FileLink data structure.

7.1.3 Algorithm

Our first algorithm is located in the ShadowExtractor method. Its sole responsibility is to utilize the AlphaVSS API to collect the SnapshotProperties of every shadow copy present on the system, and gather them up into a VolumeList object for every distinct volume, which has a copy of each SnapshotProperty as well as a list of DateTime objects for every snapshot. This is followed by the RootExtractor method, which is responsible for creating a list of DirectoryLink objects containing the root of every shadow copy gleaned from the SnapshotProperty located within the VolumeList object. In addition, the RootExtrator recursively calls on the TreeGenerator method to generate a list of DirectoryLink and FileLink objects for every sub-directory and file under the root directory, i.e. growing the tree from the root. Next, we have the PaddingCode method, which goes through each

DirectoryLink and FileLink object to pad each out with null objects equivalent to the total number of time periods being observed, to denote non-existence. The purpose of this is to make it easier for the tree list visualization method to go through the DirectoryLink linked list.

Finally, we have the ColoringTree method, which fills up the flag object in each DirectoryLink with either red, yellow or green, which as we covered earlier, denotes non-existence, no change, and change in a directory or file since the last time period respectively. For our initial prototype, change is detected in a file by its file size, which in turn would indicate a change in content. For a directory, change is detected by the number of sub-directories, files and the size of each file. Of course, this will not detect deliberate attempts to hide information, such as changing file extensions, or changing content while maintaining the exact same file size. These problems will be dealt with in our future work by reading the file header of each file to detect the actual file extension, actually comparing the content of each file instead of just the file size, and so on.

7.2 Evaluation

Although we were unable to conduct an evaluation of this product by the time of this writing, we are planning a controlled, human-computer interaction experiment. Each subject would be looking for three altered or hidden files related to drug trafficking and noting the changes in these files over a period of time using two different programs - our developed visualization tool 'Forensic Vision' and another established forensic tool such as EnCase. At the start of the study, each subject would be handed a list of information they would be looking for, similar to that of a normal forensic investigation. By the end of the study, each subject should have recorded three crucial pieces of information - the time at which they began, the name of each suspicious file, the changes that occurred and the time it was

discovered, and the time at which they completed the task. The gathering of this information would determine if we succeeded in making it easier and faster for forensic investigators to track change-over-time of suspicious files.

Half the subjects would conduct the experiment using our developed visualization tool, while the other half would utilize the established forensic tool. We would select six to ten subjects with a general knowledge of computers, with a pre-test questionnaire beforehand to categorize their abilities and level of experience. At the end of the half hour time limit for the test, each subject would also be asked to fill out a short questionnaire to aid us in comprehending which system was easier for them to use and aided in the locating of hidden and altered files. A space for additional comments and suggestions would be made available as well to further improve our product.

CHAPTER 8. USE STUDY

Here we quickly explore the level of experience and knowledge we expect from our users, as well as provide a quick step-by-step case study for the use of our program.

8.1 Evaluation

Armed now with a better understanding of the overall forensic process, the emphasis of our attention will be on the analysis portion of the process. Once an image of the original hard drive has been made, our tool can be utilized to search for evidence within it. The target audience for this tool would be mainly those with a professional interest in the field, including law enforcement personnel, computer security professionals as well as various forensic service groups. As one might expect, the technical savvy and level of experience with the forensic process varies widely between each of these three groups. For instance, police officers and detectives might have an innate understanding of the forensic process, but not necessarily possess a technical background. By contrast, computer security professionals will almost certainly have a technical background, but might not possess the requisite experience in dealing with the forensic process. Forensic service groups on the other hand would usually have a firm grasp of both, offering their services to smaller police departments and companies unable to justify full-time positions for specialized computer security professionals.

For the purposes of our tool, we would expect our target audience to have at least a basic understanding or knowledge of file systems, Windows, directory traversal, various common file formats, file attributes and regular expressions. With this, users would be able to begin searching for the relevant files within the image.

8.2 Example

Here we run through a simple example for our tool to demonstrate its basic use.

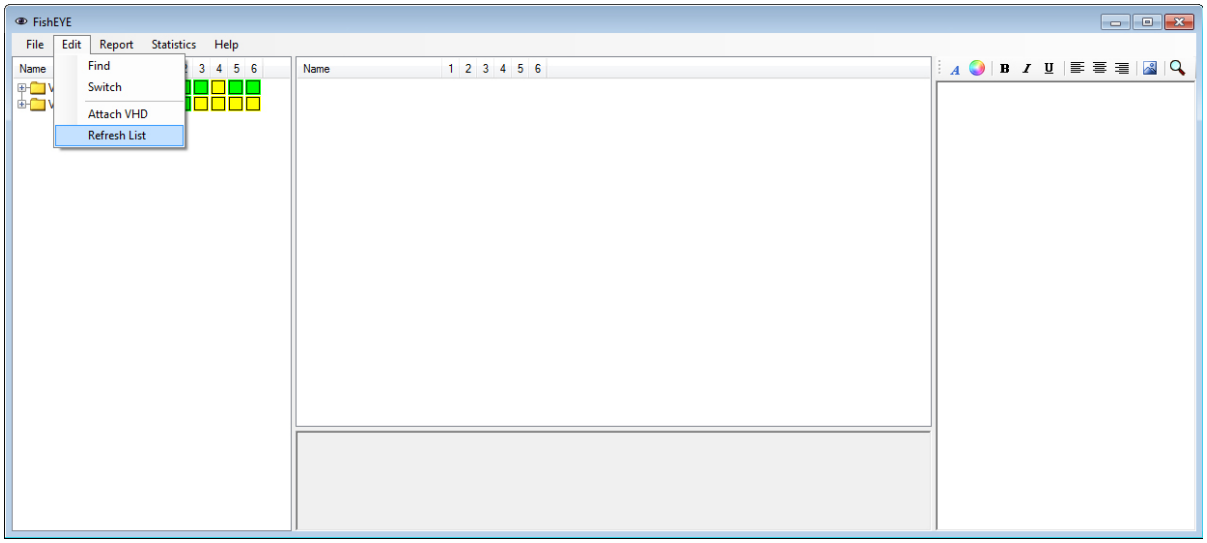

Figure 18 : To scan the system for any VSS copies after attaching the desired VHDs, click Edit → Refresh List.

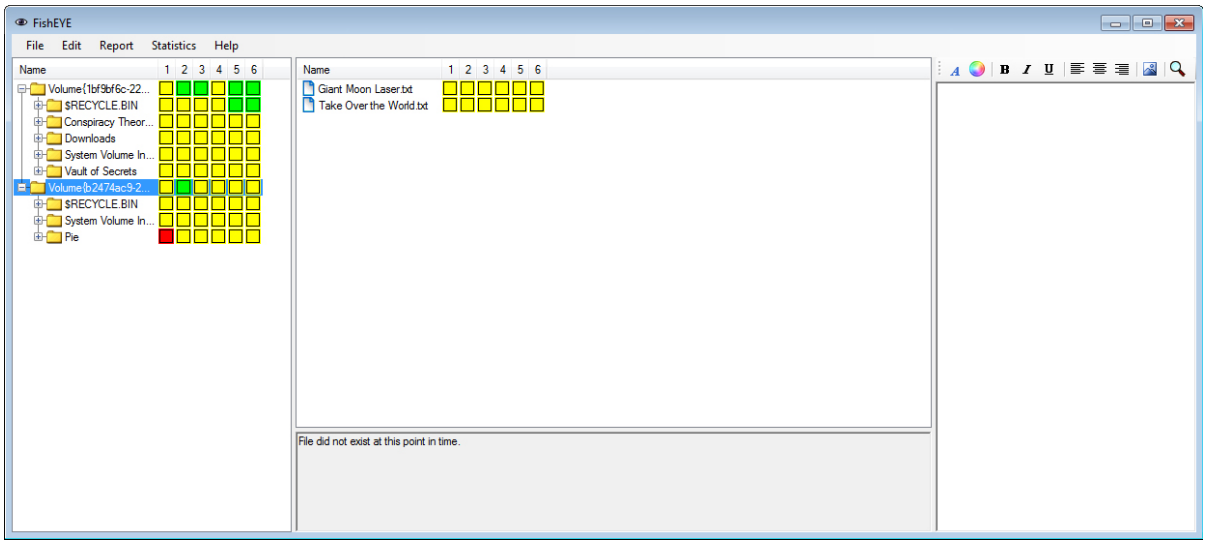

Figure 19 : Highlighting a folder on the left displays any files directly under that folder in the middle panel. For both files and folders, a red box represents non-existence, yellow means no change, and green means a change took place.

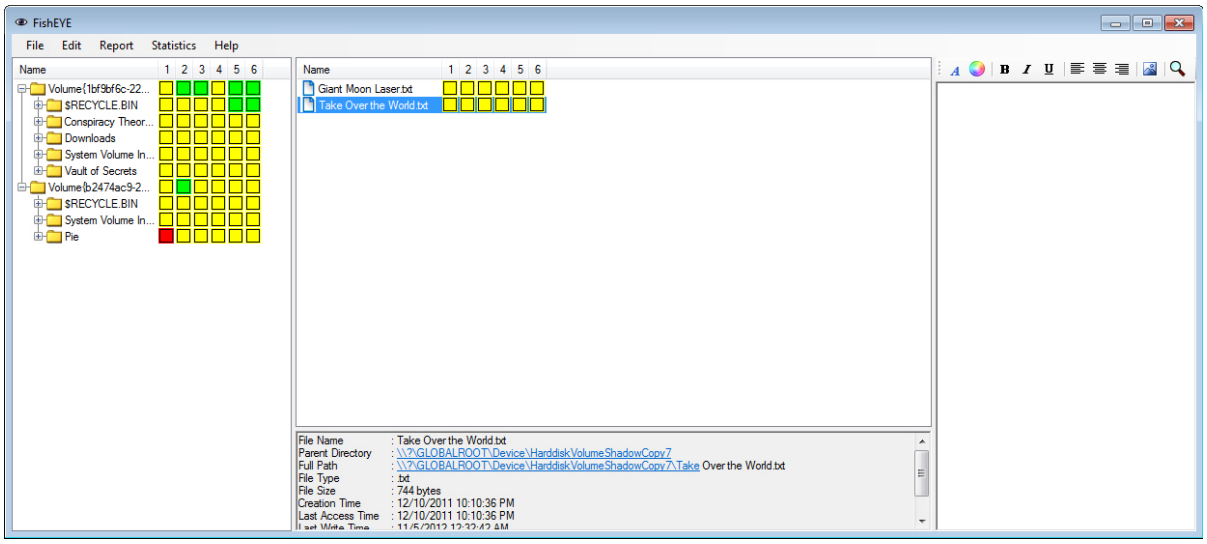

Figure 20 : Clicking each box for the file desired (in this case 'Take Over the World') displays further details for that file at that point in time in the middle bottom panel.

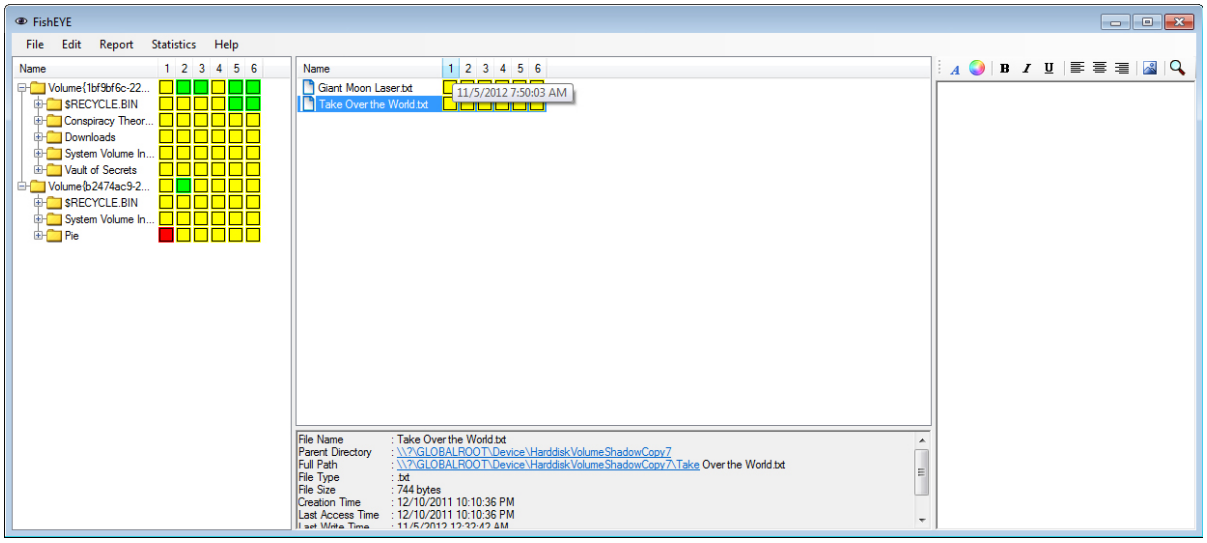

Figure 21 : Hovering your cursor over a column header displays the exact time and date a particular snapshot was taken.

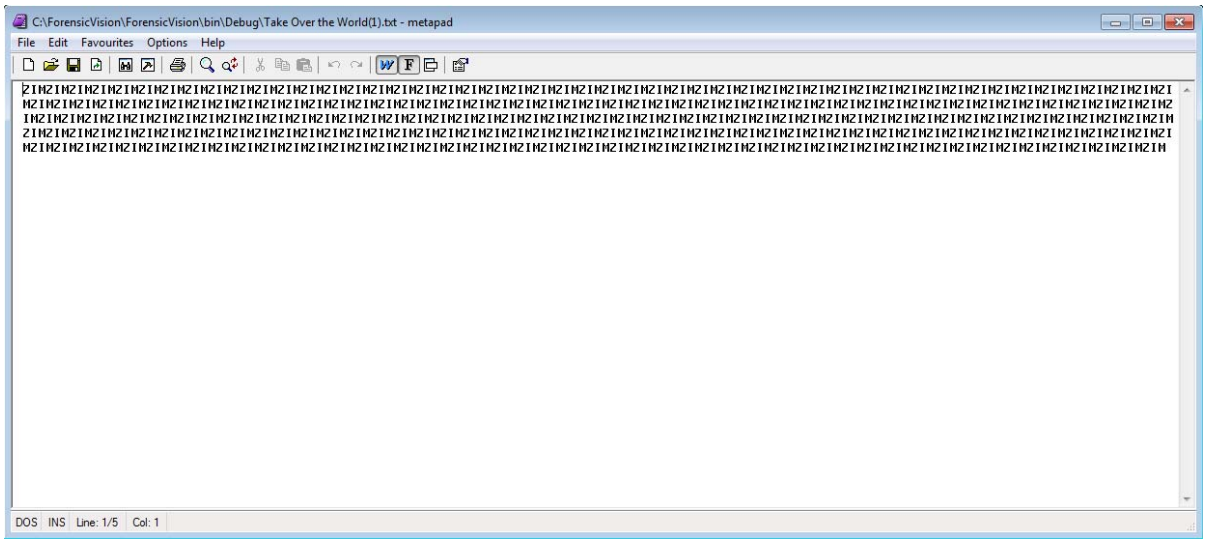

Figure 22 : Double clicking instead of a single click on the box for a file opens up a copy of that file at that point in time using the default program for that file type.

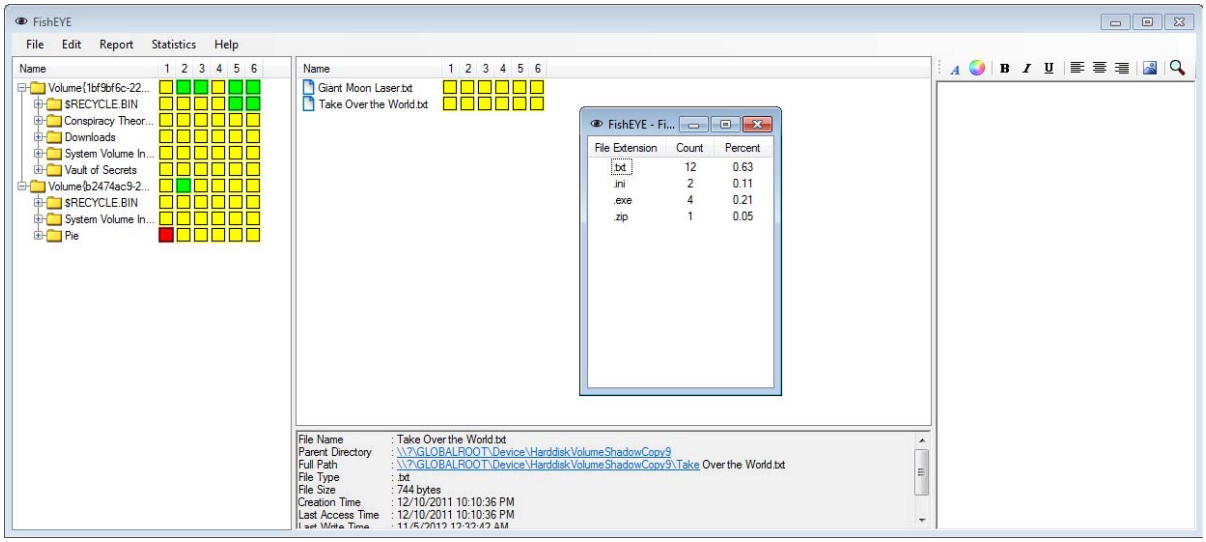

Figure 23 : Picking a file statistic from the Statistics menu bar brings up a separate window with the desired statistic, in this case File Types.

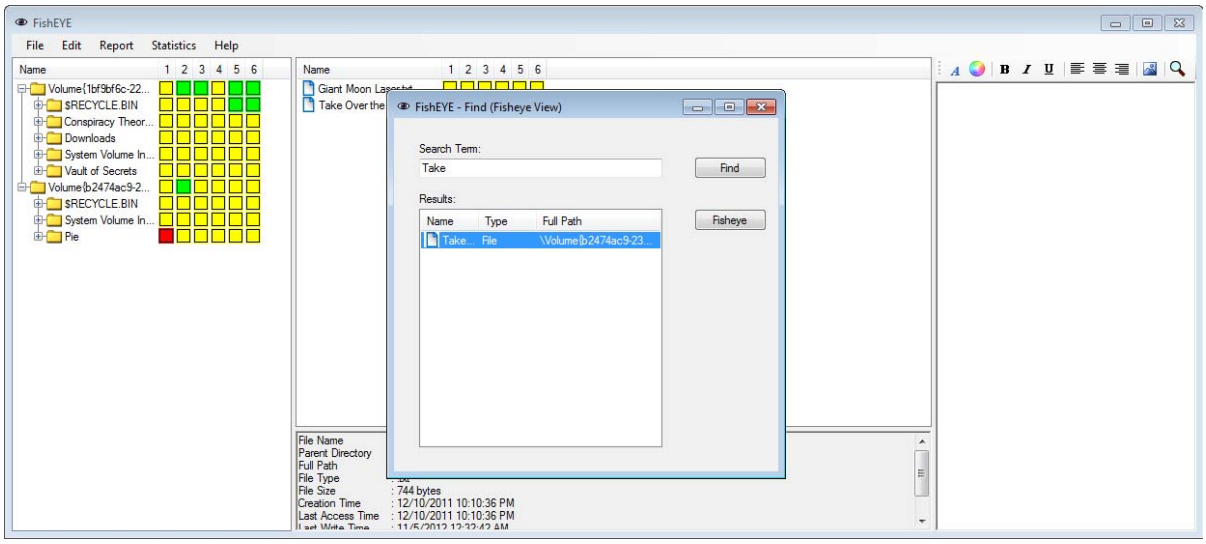

Figure 24 : To find a particular file or folder and generate a fisheye view of it, go to Edit → Find to bring up the window, type your search term and highlight the desired result before clicking Fisheye.

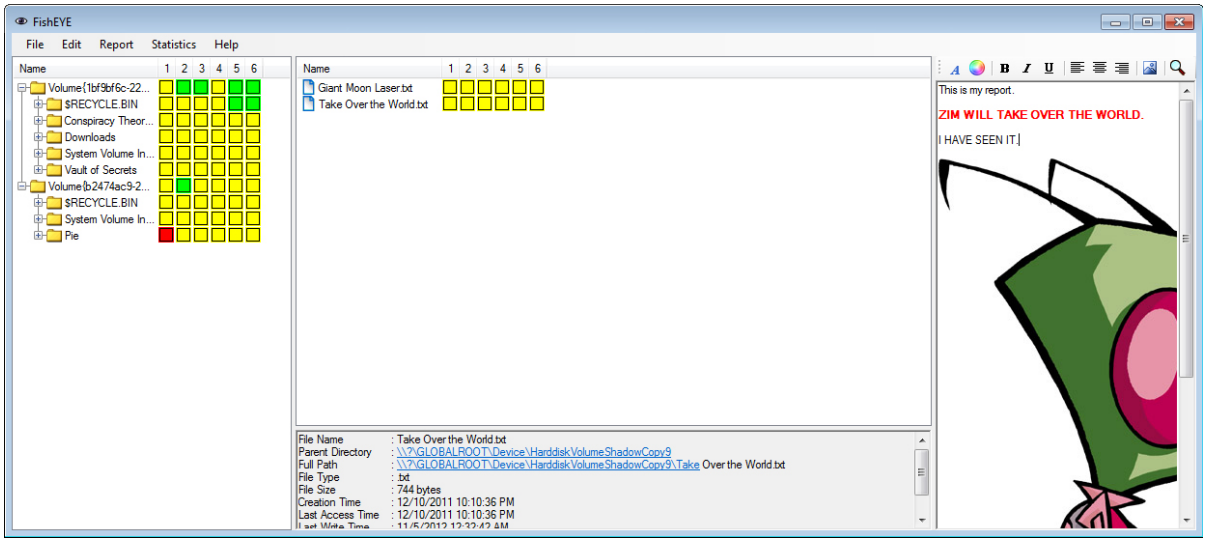

Figure 25 : The rightmost panel is a fully-featured word processor. The toolbar allows for inserting images, changing fonts etc.

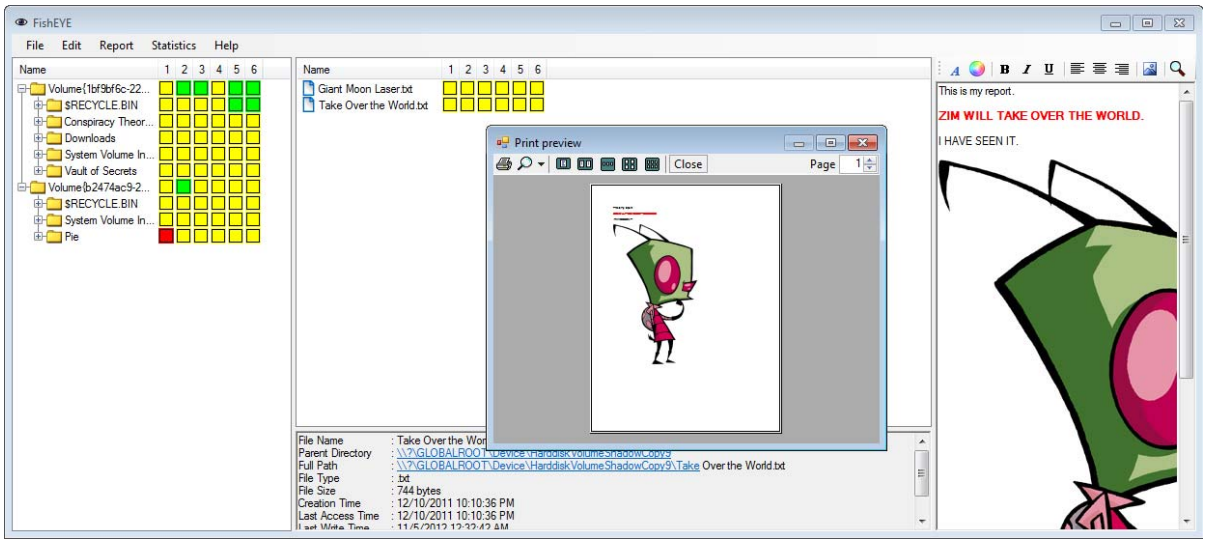

Figure 26 : To preview your report, use File → Print Preview to bring up the preview window.

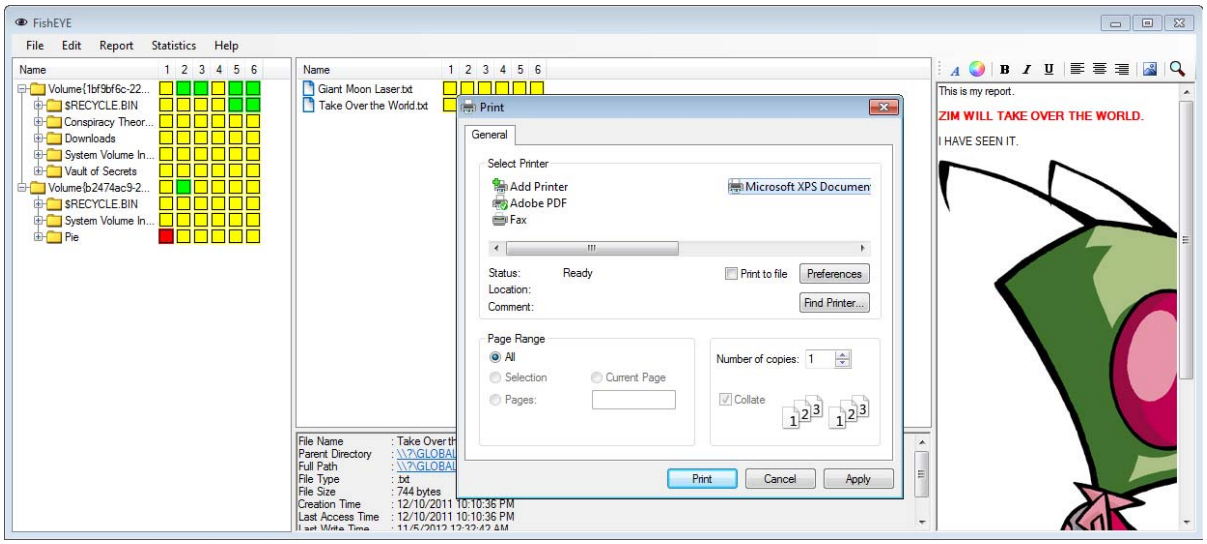

Figure 27 : To print your report, use File → Print to bring up the print dialog.

CHAPTER 9. CONCLUSIONS AND FUTURE WORK

We began this project attempting to create an original and useful product by combining the fields of computer graphics and computer forensics. We explored the motivations behind data visualization and digital forensic data that records change-over-time, and their benefit to digital investigators. Simply put, by understanding how digital evidence has changed-over-time at a glance, investigators will be better able to explain "what happened." To find the best approach to this problem, we reviewed several classic and well-known focus+context visualization approaches, while explaining why each one wasn't suited to our particular purposes before settling on an approach involving the fisheye view and a series of segmented boxes of our own design. By utilizing this approach on a directory-tree structure similar to that displayed by Windows Explorer, we believe we have developed a data visualization technique which enhances the examiner's ability to explain "what happened", which is the primary goal of every digital forensic examination. The resulting end product is an engineered piece of software with the potential of seeing everyday use by law enforcement and security experts to help curb the rise in computer crimes by helping to solve the tedious search and comparison problem of computer forensics. Moreover, we have provided a framework to build future enhancements to further analyze the digital evidence under investigation, and is scalable to large amounts of data.

However, although we have accomplished the goal of this research with the development of a system capable of visualizing a directory tree structure and the change-over-time to it by completing the implementation of a usable prototype for it, more work is necessary to further develop this into a fully featured digital forensic tool. We have divided up this work into several areas.

To begin with, comparisons are currently completed by comparing file sizes, as well as the number of files and sub-directories within each directory. Further work is necessary to allow the program to make comparisons based on changes in content, and to detect attempts at data hiding as well. Functionality to compare the content of two versions of a file in more detail would be desirable as well, with differences and changes being highlighted and displayed visually.

In addition, since adding options to reduce the number of files and directories viewed is always desirable to further focus attention on the desired directory in question, it would be useful to investigate both automatic and manual methods of reducing the number of files viewed. To this end, we suggest the implementation of a thresholding algorithm to be added to our fisheye view function for search results to adjust the number of surrounding directories displayed.

To further analyze the data set in question, more methods of data mining might be looked into and implemented as well. For now, we have laid down the basic framework by implementing a simple function to determine the file types being investigated and the number of files for each type.

Finally, once some of these additional improvements have been added to the basic framework, we intend to conduct human-computer interaction experiments to evaluate the effectiveness of our program when utilized by actual human subjects. We would evaluate our visualization scheme based on several criteria, including ease of use, faster analysis, better comprehension and the discovery of new relationships between files compared to traditional tools.

BIBLIOGRAPHY

[1] T.J. Jankun-Kelly, J. Franck, D. Wilson, J. Carver, D. Dampier, and J.E. Swan II. "Show me how you see: Lessons learned from studying computer forensics experts for visualization," Proceedings of the Fifth International Workshop on Visualization for Computer Security, Sep 2008, pp. 80-86.

[2] W.G. Kruse and J.G. Heiser. Computer Forensics: Incident Response Essentials. Addison Wesley, Boston, MA, 2002.

[3] EnCase. Internet: http://www.guidancesoftware.com/computer-forensics-ediscovery-software-digital-evidence.htm. [July 2011]

[4] AccessData. "Forensic toolkit 3.0." Internet: http://accessdata.com/products/computer-forensics/ftk. [July 2011]

[5] Cost of Hard Drive Storage Space. Internet: http://ns1758.ca/winch/winchest.html. [July 2011]

[6] T.J. Jankun-Kelly, D. Wilson, A.S. Stamps, J. Franck, J. Carver and J.E. Swan II. "A Visual Analytic Framework for Exploring Relationships in Textual Contents of Digital Forensics Evidence," Proceedings of Workshop on Visualization for Cyber Security (VizSec 2009), Oct 2009.

[7] FBI - Cyber Bust. Internet: http://www.fbi.gov/news/stories/2010/october/cyber-banking-fraud. [Nov 2012]

[8] A. Yasinsac, R.F. Erbacher, M.M. Pollitt and P.M. Sommer. "Computer Forensics Education," IEEE Security & Privacy Magazine, 2003, pp. 15-23.

[9] Merriam-Webster. "Forensic - Definition and More from the Free Merriam-Webster Dictionary." Internet: http://www.merriam-webster.com/dictionary/forensic. [Feb 2012]

[10] M. Friendly and D. Denis. "Milestones in the History of Thematic Cartography, Statistical Graphics, and Data Visualisation," Nov. 2006.

[11] Erbacher, R. and Teerlink, S. "Improving the Computer Forensic Analysis Process through Visualization." Communications of the ACM. February 2006. Vol. 49.,No.2.

[12] Ball, Thomas, and Eick, Stephen. Software Visualization in the Large. Computer, Apr 1996, Volume: 29 Issue: 4, pages 33-43.

[13] Product Downloads | AccessData. Internet: http://www.accessdata.com/support/product-downloads

[14] VHD Tool. Internet: http://archive.msdn.microsoft.com/vhdtool

[15] VHD Attach. Internet: http://www.jmedved.com/vhdattach/

[16] Card, S.K., Mackinlay, J.D., and Shneiderman, B. (Eds.) Readings in Information Visualization: Using Vision to Think, pp. 1-34, Morgan Kaufmann Publishers, San Francisco, Califomia, 1999

[17] George Robertson, Jock D. Mackinlay, Stuart Card. The Perspective Wall: Detail And Context Smoothly Integrated. In Proceedings of CHI '91 Conference, pages 173--179, April 28 - June 5, 1991, New Orleans, Louisiana, June 1991. Association for Computing Machinery.

[18] Spence, Robert and Apperley, Mark (2011): Bifocal Display. In: Soegaard, Mads and Dam, Rikke Friis (eds.). "Encyclopedia of Human-Computer Interaction". Available online at http://www.interaction-design.org/encyclopedia/bifocal_display.html

[19] S.K. Card, G.G. Robertson, and W. York, "The WebBook and the Web Forager: an Information Workspace for the World-Wide Web", Human Factors in Computer Systems, CHI'96 Conference Proceedings, ACM Press, pp. 111-117, (1996). Also in S.K. Card et al.

[20] Johnson, B. and Shneiderman, B. Treemaps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures. In Proceedings of the IEEE Information Visualization '91, pages 275-282, IEEE, 1991.

[21] John Lamping, Ramana Rao, Peter Pirolli, A Focus Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies, Xerox Palo Alto Research Center, Last Retrieved at: April 27, 2006. http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/jl_bdy.htm

[22] Sougata Mukherjea, James D. Foley, and Scott Hudson. Visualizing complex hypermedia networks through multiple hierarchical views. In Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems, volume 1 of Papers: Creating Visualizations, pages 331-337, 1995.

[23] Sarkar Manojit, Brown H. Marc, Graphical fisheye views, Communications of the ACM 37 (12) (1994) 73 - 83, http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=198384 , last visit: 24th of April 2006.

[24] Did You Know? Why are Traffic Lights Red, Yellow and Green? Internet: http://www.myuniversalfacts.com/2005/10/why-are-traffic-lights-red-yellow-and.html. [July 2011]

[25] Keahey, T. Alan, and Marley, Julianne. Viewing Text with Non-linear Magnification: An experimental Study. Technical Report 459, Department of Computer Science, Indiana University, April 1996.

[26] AlphaVss. Internet: http://alphavss.codeplex.com/. [July 2011]