Source Viewer 3D (sv3D) – A Framework for Software Visualization

Jonathan I. Maletic, Andrian Marcus, Louis Feng
Department of Computer Science
Kent State University
Kent Ohio 44242
jmaletic@cs.kent.edu, amarcus@cs.kent.edu, lfeng@cs.kent.edu

Abstract

Source Viewer 3D is a software visualization framework that uses a 3D metaphor to represent software system and analysis data. The 3D representation is based on the SeeSoft pixel metaphor. It extends the original metaphor by rendering the visualization in a 3D space. New, object-based manipulation methods and simultaneous alternative mappings are available to the user.

1. Description

Source Viewer 3D (sv3D) is a software visualization framework that builds on the SeeSoft [1, 2] metaphor. It brings a number of enhancements and extensions over SeeSoft-type representations. In particular it creates 3D renderings of the raw data and various artifacts of the software system and their attributes can be mapped to the 3D metaphors at different abstraction levels. It implements improved, object-based user interactions, is independent of the analysis tool, and it accepts a simple and flexible input in XML format. The output of numerous analysis tools can be easily translated to sv3D input format and the design and implementation of our system is extensible.

SeeSoft-like tools have a variety of uses in assisting the user solving software engineering and comprehension tasks. sv3D can be used for all these tasks such as: fault localization [4], visualization of execution traces [6], source code browsing [3], impact analysis, evolution, complexity, and slicing [1], etc. In addition, by allowing visualization of additional information (via 3D), sv3D can be used for solving other more complex tasks. For example, in the case of Tarantula [4], using height instead of brightness would improve the visualization and make the user's task easier.

Most software engineering tasks during maintenance and evolution require understanding of various elements of the software system and also of data resulted from analysis. The main features of sv3D, namely, advanced user interactions and usage of the 3D space for visualization directly support the user in achieving a better understanding of analysis data. This process, in turn, directly supports a variety of tasks.

2. Support for User Interaction

We focus here on the types of user tasks and interactions that are supported by sv3D. While this is not directly related to solving/visualizing specific software engineering tasks it is prerequisite for a software visualization tool.

One of the strongest features of sv3D is its *overview* features. The underlying 2D visualization construct used in designing the poly cylinder containers is the pixel bar chart [5], which generalizes the concept used by SeeSoft. Thus sv3D can show large amounts of source code in one view just as the SeeSoft metaphor. Figure 1 shows a 3D overview of a small system with 30 C++ source code files and approximately 4000 lines of code. Each file is mapped to one container. Each container is made up of a number of poly cylinders. Each poly cylinder represents a line of source code. In this simple example shading (color) is used to represent the type of control structure a statement is in and the height is used to represent the nesting level. On top of each container the name of the associated file is visible. When manipulating a container in the 3D space, the name of the file always faces the camera.

sv3D supports *zooming* and *panning* at variable speeds. This is especially important because the visualization space can be quite large. Each container in the visualization can be manipulated individually (rotate, scale, translate). The user can also zoom in and out on the entire space. Files can be brought into a closer view and manipulated for a better camera angle.

At this point sv3D directly supports a number of *filtering* methods. Un-interesting units can be filtered through their attributes or by direct manipulation. Transparency is used to deal with both occlusion and filtering. The user can chose various degrees of transparency on each class of cylinder, based on their attributes (color, shape, or texture). With semi-transparency the global context is preserved and heuristic information is retained. Elevation can also be used to filter out un-interesting units by lifting them into separate levels.

Currently our emphasis with regard to *details-on-demand* is for simplicity. It is important to be able to support user interaction, therefore performance is

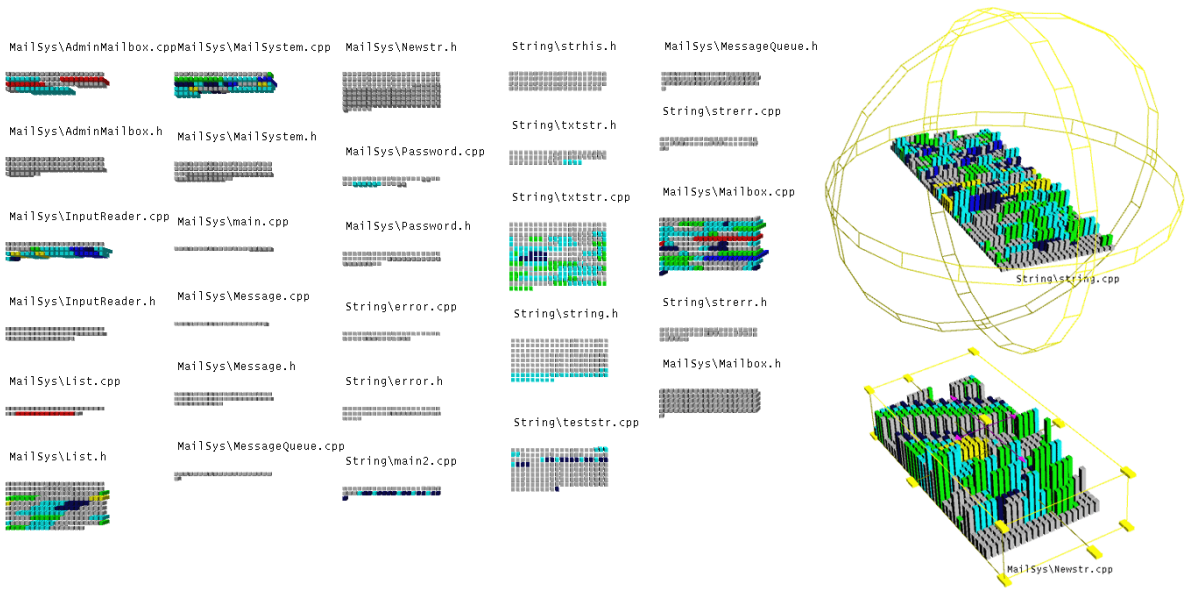

Figure 1. Overview in the 3D space of the mailing system. Color represents control structure and height represents nesting level. Two files have active manipulators (handle box for scaling on the left and track ball for rotating on the right). For a color view see www.sdml.cs.kent.edu

important. Two types of 3D manipulators (i.e., track ball and handle box) are available to the user to interact with the visualization. An information panel displays the data values on selected items.

The *relationships* between items are shown through the elements of the visualization that do not directly support representation of quantitative data (such as shape, texture, and position). The other elements (such as color and height) can also be used to show relationships. The 3D space allows arranging the containers in any place. We are investigating ways to use links between the 3D containers and arrange them in a graph layout.

The user can take snapshots of the current view to track a *history*. The current view is described by a scene graph, which is composed by the attributes of the camera and all 3D objects. These snapshots of the scene graph can be saved and reviewed. A sequence of such snapshots can be played, thus representing a path within the visualization. More than that, we intend to build into sv3D change tracking based on individual users.

3. Current and Future Work

sv3D is implemented in C++ and uses Qt for the user interfaces and Open Inventor for 3D rendering. See www.sdml.cs.kent.edu for additional information and downloads.

In the future versions of sv3D, position of the cylinder within a container can represent another type of information (or dimension). We need to define these visual attributes very carefully to ensure their usefulness. Containers in the 3D space can possibly be connected by edges to form a 3D graph. This will allow representation of hierarchical data and also diagrammatic visualizations (e.g., UML class diagrams). A number of user experiments to evaluate this system are being planned.

This work was supported in part by grants from the Office of Naval Research N00014-00-1-0769 and the National Science Foundation CCR-02-04175.

4. References

[1] Ball, T. and Eick, S., "Software Visualization in the Large", *Computer*, vol. 29, no. 4, April 1996, pp. 33-43.

[2] Eick, S., Steffen, J. L., and Summer, E. E., "Seesoft - A Tool For Visualizing Line Oriented Software Statistics", *IEEE TSE*, vol. 18, no. 11, November 1992, pp. 957-968.

[3] Griswold, W. G., Yuan, J. J., and Kato, Y., "Exploiting the Map Metaphor in a Tool for Software Evolution", in Proceedings of ICSE'01, Toronto, 2001, pp. 265-274.

[4] Jones, J. A., Harrold, M. J., and Stasko, J. T., "Visualization for Fault Localization", in Proceedings of ICSE 2001 Workshop on Software Visualization, Toronto, 2001, pp. 71-75.

[5] Keim, D. A., Hao, M. C., Dayal, U., and Hsu, M., "Pixel bar charts: a visualization technique for very large multi-attribute data sets", *Information Visualization*, vol. 1, no. 1, March 2002, pp. 20-34.

[6] Reiss, S. P., "Bee/Hive: A Software Visualization Back End", in Proceedings of ICSE 2001 Workshop on Software Visualization, Toronto, 2001, pp. 44-48.