Evaluating Three-Dimensional Information Visualization Designs: a Case Study of Three Designs

Ulrika Wiss David Carr Håkan Jonsson
Dept. of Computer Science and Electrical Engineering
Luleå University of Technology
971 87 Luleå, Sweden

Abstract

A number of three-dimensional information visualization designs have been invented during the last years. However, comparisons of such designs have been scarce, making it difficult for application developers to select a suitable design. This paper reports on a case study where three existing visualization designs have been implemented and evaluated. We found that the three information visualization designs have inherent problems when used for visualizing different data sets, and that certain tasks can not be supported by the designs. A general methodology for evaluation is presented, which comprises evaluation of suitability for different data sets as well as evaluation of support for user tasks.

Keywords: 3D information visualization, evaluation, visualization design, VRML, Java.

1 Introduction

Visualization's primary goal is to make it easier for people to understand and use vast amounts of data. Information visualization, as opposed to scientific visualization, aims to visualize abstract data that may have no natural visual representation. Because of this, information visualization requires an additional activity compared to scientific visualization: inventing an information visualization design.

An *information visualization application* is a software system that uses information visualization. It is targeted towards a set of user tasks that are designed with certain user skill levels in mind and intended for a specific platform, input devices, etc. Embedded in such an application is the graphical image of the information. The *information visualization design* is the conceptual description of how this graphical image visualizes the information. (I.e., what visual elements the image should contain and how these visual elements are mapped to the underlying data.) Such a design can be suitable for a number of different information visualization applications.

Technological advances in computer graphics have made 3-dimensional (3D) information visualization feasible on personal computers. Meanwhile, the World-Wide Web (WWW) has made a vast amount of information available to individuals. In parallel with these developments a number of 3D information visualization designs have been invented by both researchers and commercial software developers.

But, the research community has still made very few comparisons and evaluations of these design inventions. We do not really know how different designs compare for different data sets or user tasks. This means that it is difficult to make a decision as to which information visualization design to use when designing an information visualization application.

As a first step towards such an understanding, we have implemented three existing 3D information visualization designs: the Cam Tree [10], the Information Cube [8], and the Information Landscape [1, 13]. Our implementation was used to visualize a number of different data sets – two of which we use as a basis for our discussion in this paper.

We begin by discussing previous work in classifying and evaluating information visualization design and by giving examples of 3D information visualization designs (Section 2). Next, we describe the three information visualization designs that we have implemented (Section 3) and the details of our implementation (Section 4). This is followed by a comparison of the three visualizations based on our implementation (Section 5). Finally, we draw some tentative conclusions and point out further research directions (Section 6).

2 Previous Work

2.1 Related Designs

Information visualization designs similar to the Cam tree include the work of Munzner and Burchard [7] which displays directed graphs with cycles (such as the WWW) in hyperbolic space. The SeeNet3D information visualization application [4] visualizes global networks on a sphere and local networks on a map image.

The Bead [3] system is similar to the Information Landscape. It displays bibliographic data with documents as cubes interconnected by triangles in a landscape-like space.

Nested designs, such as the Information Cube, are more uncommon. Feiner and Beshers [5] present the n-Vision system for visualizing multi-dimensional data (more than 3 dimensions). The visualization consists of nested 3D coordinate systems with axes. The Web Forager and the Web Book [2] adopt a metaphorical nesting, where WWW pages are contained in books, that in turn can be contained in book shelves placed in a 3D room.

Another type of 3D information visualization design (not represented in our selection) is raised surfaces, where information is displayed on a surface that can be raised towards the user to provide extra detail. An example of this is the Document Lens [9], which is used for laying out pages of a document on a rectangular surface. 3DPS (3-Dimensional Pliable Surfaces) [11] is another example of this type of design, an information visualization application for distortion based display of maps and graphs.

2.2 Evaluation of 3D User Interfaces

Previous work in evaluating 3D user interfaces mainly concentrates on experimental evaluation of the lower level cognitive aspects. The work of Hubona, Shirah and Fout [6] suggests that users' understanding of a 3D structure improves when they can manipulate the structure. The work of Ware and Franck [14] indicates that displaying data in three dimensions instead of two can make it easier for users to understand the data. Our work complements this type of evaluation by considering higher-level properties specific for 3D information visualization designs – suitability for different data sets and support for user tasks.

3 The Three Designs

Since our aim was to compare visualizations of the same data sets, we implemented three existing 3D information visualization designs that all visualize hierarchical data: The Cam Tree [10], the Information Cube [8] and the Information Landscape [1, 13].

We chose these three information visualization designs since they are all different in the way they visualize the hierarchical data. The Cam Tree and the Information Landscape both lay out the data in a "traditional" tree, but differ in their use of the 3D space, the Information Landscape being a "2.5 D" visualization. The Information Cube instead represents parent-child relationships by nesting children inside parents.

3.1 The Cam Tree

The Cam Tree visualizes hierarchies as trees of labeled rectangular shapes representing nodes and leaves, interconnected by lines. Each subtree is laid out as a cone with its root at the top of the cone and the children along the cone base. Rectangles and cones are semi-transparent in order to reduce problems with occlusion. Figures 1 and 4 show our implementation of the Cam Tree. In the cited paper [10], the information visualization design included the shadow of the tree placed underneath the tree. We did not implement this since its importance was said to be small.

Interactions with the Cam Tree includes rotation of the tree when a node or leaf has been selected with the mouse. This brings the path to the selected rectangle closest to the user and highlights the rectangles on that path. It is also possible to prune the tree via a control panel with buttons.

3.2 The Information Cube

The Information Cube uses semi-transparent, nested cubes to represent leaves or internal nodes. The parent-child relationships are represented by nesting child cubes inside their parent cubes, scaling cubes to enclose the contained cubes. Textual labels are displayed on cube surfaces. Color and transparency level indicate the currently selected cube. Figures 2 and 5 show our implementation of the Information Cube.

The original system is designed for use with special virtual reality equipment. Our implementation relies entirely on the VRML browser's interface for interactions. We found this sufficient, since the purpose of our study was to evaluate the graphic visualization rather than the interactions.

3.3 The Information Landscape

Our implementation of the Information Landscape is based on the File System Navigator (`fsn`) [13] from Silicon Graphics and the Harmony Information

Landscape [1]. These two information visualization designs are similar, basically differing only in how they are used by the surrounding information visualization application.

In the Information Landscape, nodes are represented as pedestal shapes standing on a flat surface with lines connecting pedestals to form a tree. Leaves are represented as box shapes standing on the pedestals. This makes the pedestal cross-section proportional to the number of leaf children. The height of the boxes encodes an attribute such as the size of the data element represented by the box. Being a "2.5 D" visualization, the pedestals are restricted to a flat surface in 3D space. Only the box height makes use of the third dimension. Figures 3 and 6 show our implementation of the Information Landscape.

Interactions in the Information Landscape include selecting boxes or pedestals with the mouse and "flying" up to a viewpoint close to the selected element.

4 Implementation

Our entire system for 3D information visualization is implemented in the programming language Java^{tm} from Sun Microsystems, and the Virtual Reality Modeling Language (VRML). It should be further noted that we make no use of special hardware. The interaction with a particular visualization is performed only by using a mouse and a keyboard. The actual visualization is shown on a conventional screen.

The system consists of two parts:

1. A visualization generator written in Java that takes hierarchical data as input and produces VRML code for visualizations consistent with the three designs mentioned above, and
2. A client application consisting of a WWW page defined by VRML code produced by the visualization generator and a Java applet that controls the visualization via the External Authoring Interface, as described below.

4.1 The Visualization Generator

The visualization generator transforms hierarchical data into VRML code representing a 3D visualization. The hierarchical data is represented internally as a tree which is built as the data is read. Currently, the generator is able to read data from two sources, a database with articles of an electronic newspaper and a subtree in a file system.

Once all the data has been read and the tree has been built, an initial layout of the tree in a virtual 3D (Euclidean) space is computed. The layout specifies that a 3D object occupies some distinct portion of 3D space. Depending on the layout rules used during construction, different algorithms for producing layouts differ tremendously in complexity. Some layouts contain optimization problems which are NP-Complete. (For example, general problems of packing 3D objects into minimal space.) Since the three papers do not clearly state which algorithms were used to produce the layouts and our objective was not to improve their implementations or optimize resources such as memory or CPU time, we used fairly simple non-optimizing algorithms. We prioritized creating layouts and visualizations closely corresponding to those given in the papers.

The layout algorithms for the three designs are very similar and work in two steps. During the first step, the layout of the visualization is computed. The tree is traversed recursively, and the layout of a node is based on the layout of its children. In all three designs, the layouts are done so that no pair of 3D objects intersect. The layout is also done to enable 3D objects to be moved in 3D space without bumping into each other. For the Cam Tree, the algorithm calculates the necessary base circumference for each node's underlying cone. For the Information Cube, the algorithm instead calculates how big each cube must be to fit all its children inside. For the Information Landscape, the algorithm calculates how much space in the X direction each node needs for all of its children.

The second step consists of creating VRML code. The VRML code created includes a 3D shape for each node in the tree and positional information for each node based on the information from the first step. For the Cam Tree, sibling nodes and leaves are positioned at equal distances around the bottom circumference of their parent cone. Nodes and leaves alternate to minimize the risk for intersecting subtrees. For the Information Landscape, the algorithm takes a pessimistic approach: the sides of each cube are made big enough to fit the $\lceil \sqrt[3]{n} \rceil$ largest child cubes, where n is the number of children. For the Information Landscape, the algorithm positions nodes so that no subtrees overlap. It also adjusts the space between the rows that make up the levels so that we do not get connecting lines that are difficult to follow.

4.2 The Client Application

We implemented the client application using the External Authoring Interface (EAI) which is a proposal for a VRML 2.0 Informative Annex. The EAI is currently implemented in the Cosmo Player^{tm} by Sili-

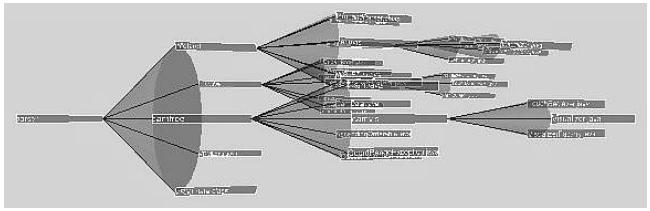

Figure 1: Our implementation of the Cam Tree visualizing the file system data set.

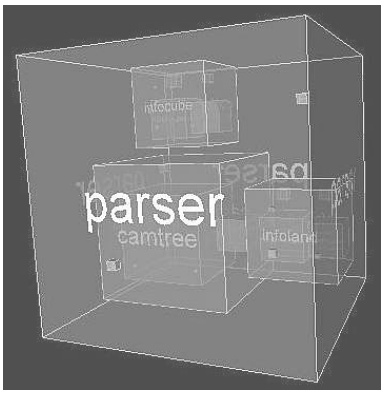

Figure 2: Our implementation of the Information Cube visualizing the file system data set

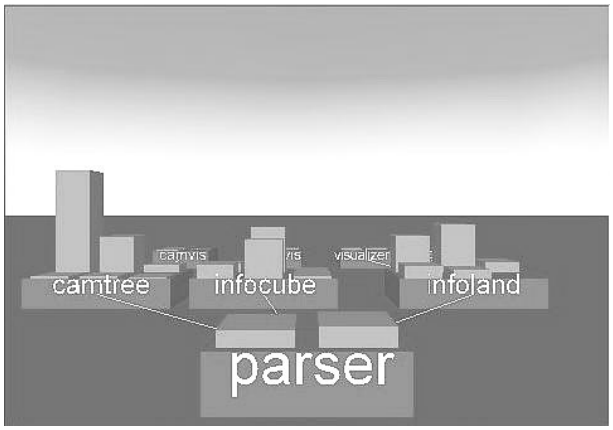

Figure 3: Our implementation of the Information Landscape visualizing the file system data set

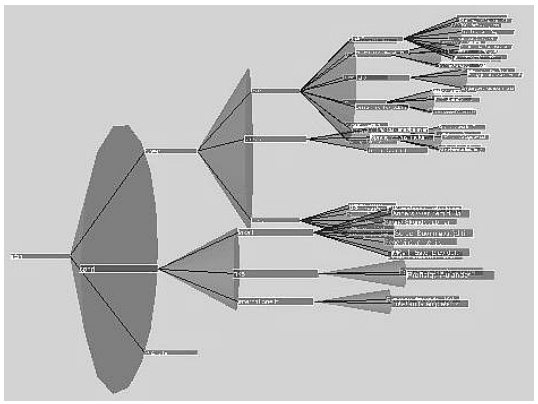

Figure 4: Our implementation of the Cam Tree visualizing the newspaper TOC data set. Note the occlusion on the right hand side of the tree.

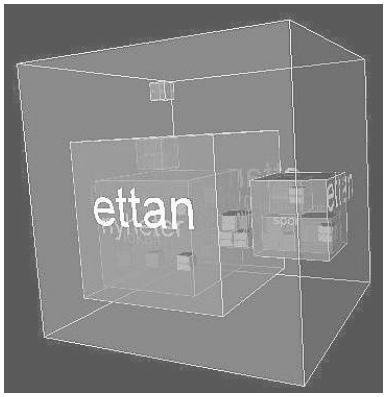

Figure 5: Our implementation of the Information Cube visualizing the newspaper TOC data set. Note the excess space in the cube, and the small child cube in the top left corner.

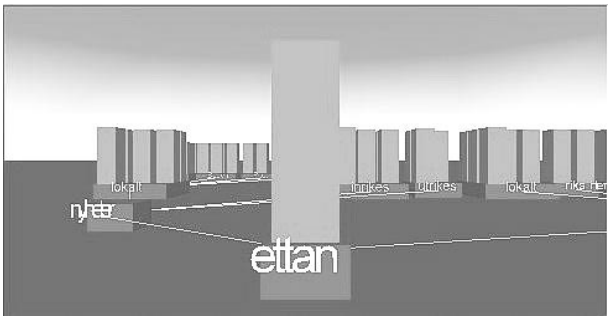

Figure 6: Our implementation of the Information Landscape visualizing the newspaper TOC data set. Note that the rightmost part of the tree can not be seen.

con Graphics. Via the EAI, Java applets on a WWW page can obtain access to objects within a VRML world on that same page. We choose this implementation strategy to achieve a platform independent, easily distributed solution. The speed and ease by which the system could be implemented was another reason for our choice. Beta-quality EAI tools, and a relatively slow rendering speed for the 3D world, were the main problems with this implementation choice.

Alternative implementation choices included implementing the visualizations in VRML only, or in Java only. As for VRML only, we found that the interaction possibilities built into VRML were a bit too limited. We also found that it is difficult to let actions within the world have any effects outside of the world, something we anticipated needing for future work. Implementing in Java only was rejected because of the lack of standard 3D libraries for Java at the time. We also anticipated that a Java applet with graphics possibilities would involve the need to download a great number of classes to the client while VRML is a relatively lightweight description language.

5 Comparing the Visualizations

Our methodology for comparing the three visualizations aim to give guidance to the suitability of each design for specific usage situations. The methodology proposes the following heuristics:

1. Select a number of information visualization designs that seem suitable for the problem at hand.
2. Evaluate suitability for different data sets:
   (a) Construct two or three different data sets that represent possible data for the application.
   (b) Create visualizations of these data sets with all the selected information visualization designs. This can be done with simple paper mockups if an implementation is regarded as too time-consuming.
   (c) Use the visualizations to find problems with visualizing the different data sets.
3. Evaluate support for user tasks:
   (a) Do a task analysis in order to identify important tasks that the visualizations need to support.
   (b) Compare the selected designs against these important tasks. The visualizations created in the previous step may be used as a help in this evaluation.
4. Use results from these two evaluations to decide:
   (a) Which of the selected information visualization designs to use in the information visualization application (possibly more than one).
   (b) What additional functionality to include in the information visualization application.

Section 5.1 presents step 2 (evaluating suitability for different data sets) for our three visualization designs, and Section 5.2 presents step 3 (evaluating support for user tasks).

5.1 Suitability for Different Data Sets

As described in Section 4.1 above, our algorithms for laying out the visualizations aimed to make the resulting visualization similar to the description in the articles. Naturally, we can not be sure that our layout algorithms were similar to the ones used by the authors. Nevertheless, we will attempt to evaluate how the information visualization designs work with data sets that are different than the ones described in the referenced publications.

We chose to visualize an electronic-newspaper table-of-contents (TOC) and part of a file system with the three different information visualization designs. In both data sets, interior nodes and leaves have different properties. The interior nodes primarily act as containers for data while leaves make up the actual data of interest (the file or the newspaper article). The file system data set is a relatively small and well balanced data set. It contains 30 leaves and 8 internal nodes on four levels, and most of the leaves are on the third and fourth level. The newspaper TOC data set, on the other hand, is larger and more unbalanced. It contains 56 leaves and 14 internal nodes on five levels. One of the branches contains 34 leaves and is the only one extending to the fifth level. Screen shots of the file system visualizations can be seen in Figures 1, 2, and 3. Figures 4, 5, and 6 show screen shots of the newspaper TOC visualizations.

For the Cam Tree, a data set with many levels and many subhierarchies will result in occluded subtrees. In the pictures in Robertson et al [10], the visualized data sets have few levels and not many subhierarchies on each level. As a result, the occlusion is not very disturbing. But for both of our data sets, we get a Cam Tree with many occlusions. Even though the Cam Tree uses transparency to mitigate occlusion, viewing is difficult even for the relatively small and well balanced file system data set (Figure 1). As seen in Figure 4, the occlusion is even worse with the newspaper TOC data set.

For the Information Cube, we will get excess space inside each cube if there are fewer than $\lceil \sqrt[3]{n} \rceil^3$ chil-

dren or if the children are of varying sizes. The resulting size of the surrounding cube will then not represent the size of the contents very well.

Yet another problem for this type of data is that if the difference between the biggest and the smallest subhierarchy is large, the smallest child cube will be so small that it is difficult to see. This can be seen in both of our data sets. In our visualization of the file system, files on the first level are very small compared to directories (Figure 2). In our visualization of the newspaper table-of-contents the sizes of leaves do not vary (Figure 5). However, the effect can still be seen even though it is not as disturbing.

For the Information Landscape, we get some excess space in the X direction when subhierarchies are of varying size, which in turn makes the landscape wide. A typical viewpoint for the Information Landscape is "floating" above the surface right behind a node and viewing its subtrees as they spread out into the distance. With a very wide landscape, it becomes difficult to see the entire subtree from this position. Either we will need to move the viewpoint back causing the boxes and pedestals to shrink, or part of the subtree will not fit on the display. With the file system data set shown in Figure 3, this is not necessary. But, the more unbalanced and larger newspaper TOC data set (Figure 6) illustrates how the rightmost nodes disappear from sight. This is probably part of the reason why both Information Landscape applications described include an overview map.

So, in conclusion, all three information visualization designs create problems with different data sets. The Cam Tree produces a clutter for "bottom heavy" data sets (i.e., hierarchies with many wide subhierarchies). Even with the relatively small file system data set this is a problem. The Information Cube will show misrepresented sizes as soon as the contained cubes are of varying sizes. This is often the case when a parent node contains both leaves and subhierarchies. The ideal data set for the Information Cube would be a hierarchy where all leaves are at the same level. This is a property that neither of our data sets have. The Information Landscape has problems with data sets where a node has many children. This creates a wide layout that can not be seen all at once. The placement and number of leaves is of less importance.

If it is possible to predict what structure the data will have, this may help in deciding what information visualization design is most suitable. But if the application should be able to support arbitrary data sets which can be expected in many types of applications, selecting an information visualization design will be difficult. One strategy for handling this is to design the information visualization application so that it can alleviate these problems, such as adding an overview map to an Information Landscape application. Another strategy is to provide several alternative visualizations in the same application, giving users multiple views of the same data set.

5.2 Support for User Tasks

Our task analysis is based on Shneiderman [12], who presents seven high level tasks that an information visualization application should support. For evaluation purposes, we must refine these into lower-level tasks for the two application domains represented by our two data sets, the newspaper TOC and the file system data set.

Overview: *Gain an overview of the entire collection.* For the overview task, the user might want somewhat different information about a file system than a newspaper TOC. For the file system, it is interesting to see the size of the leaves, i.e. the files. For the newspaper TOC, sizes of leaves (articles) are not as interesting as the actual number of leaves, both totally and in each section.

Zoom: *Zoom in on items of interest.* When zooming, it is important that global context can be retained. If it can not, it may be necessary to add some type of overview map in the application where the visualization is embedded.

Filter: *Filter out uninteresting items.* Filtering by removing parts of the visualization will necessarily disturb the global context. Therefore, it is important to see whether the design supports some kind of abstraction of the removed parts.

Details-on-demand: *Select an item or group and get details when needed.* Getting details on a selected item is usually implemented by the embedding application. One variant of this that could be supported by the information visualization design is level-of-detail: increasing the amount of detail showed in the visualization when the user gets close to it.

Relate: *View relationships among items.* For a hierarchical data structure, it is absolutely necessary that a visualization shows parent-child relationships. The three designs do this in very different ways. This affects how well the user can separate nodes from leaves – something that is interesting both for the newspaper TOC and file system.

History: *Keep a history of actions to support undo, replay, and progressive refinement.* A possible use of this for the newspaper TOC would be showing what articles the user has read, "visited", and possibly in what order. For the file system, if the information visualization application allows us to delete, create, and move files and directories, we need some way to show the previous state of the data.

Extract: *Allow extraction of sub-collections and of query parameters.* This task concerns saving the current state of the visualization. This is related only to the application and the underlying data set. How the data is visualized does not affect this. The extract task is therefore excluded from our evaluation.

An information visualization application normally consists of an embedded visualization where interactions can be done either directly in the visualization or via some type of control panel. This entire application should support the seven tasks. However, evaluating the information visualization design in isolation, we can only look at whether it can be used for implementing a specific task or not. It is then up to the application to implement the tasks. Table 1 summarizes how the three information visualization designs support implementing these tasks.

This evaluation shows that the way a certain information visualization design visualizes information may make it impossible to support a certain task. For example, it is not possible to retain global context while zooming-in with an Information Cube or an Information Landscape. Some designs may be modified to support a certain task. For example, a visitation path can be supported by the Cam Tree if we color nodes and links that are part of this path. However, this may interfere with the original design where color highlighting is used to denote the path to the currently selected node. Such modifications must be done very carefully in order to avoid undesirable side effects.

When creating an information visualization application, it is important to identify primary tasks before choosing an information visualization design. If a task can not be supported by the selected design, we must either consider modifying the design, or we must add extra functionality to our application. The overview map used in Information Landscape applications is an example of the latter strategy. Another alternative would be to include several different visualizations in the application.

Task	*Subtask*	*Cam Tree*	*Information Cube*	*Information Landscape*
Overview	Size of leaves	not supported	supported, but occluded on lower levels	supported
	Number of leaves	supported	supported, but occluded on lower levels	supported
Zoom	Global context retained	supported	not supported	not supported
Filter	Abstraction	not supported	partly supported (by lower level occlusion)	not supported
Details-on-demand	Level-of-detail	not supported	supported (by diminished occlusion)	not supported
Relate	Separating nodes and leaves	minimally supported	somewhat supported (color)	well supported
History	Visiting path	can be supported	can not be supported easily (due to nesting)	can be supported
	Previous state	can be supported	can be supported	can be supported

Table 1: Task support by the three information visualization designs.

6 Conclusions and Future Work

We have implemented three different information visualization designs and visualized two different data sets with them. The visualizations have then been compared and evaluated with regard to their suitability for different data sets, as well as their support for user tasks. We have outlined a methodology (Section 5) for this evaluation, which can be applied to any type of information visualization and does not necessarily need to include implementation. We believe it can be very useful when designing an information visualization application.

Our evaluation results have pointed out some important differences in these three information visualization designs which were not obvious from the original descriptions. These results can serve as an important help in choosing information visualization designs for specific information visualization applications.

Firstly, the three information visualization designs behave differently when used for visualizing different data sets. While the descriptions often concentrate on data sets that are suitable for their designs, our work has shown that data sets with other properties may cause significant problems in viewing the visualization.

It is very important to consider this when using an information visualization design in an application.

Secondly, information visualization designs are often targeted towards a specific problem and support tasks associated with this problem very well. But, this also means that the designs make it difficult or impossible to support other user tasks. This is seldom mentioned in the descriptions, but is crucial when choosing a design for a specific set of tasks.

In order to make use of all the research in inventing information visualization designs, we propose the use of several different designs in the same information visualization application. In our opinion, this would be advantageous both for the users and the developers. The developers would be able to reuse the work of information visualization researchers and concentrate on integration, presentation, and additional functionality. The users would be given a greater freedom to explore the information in many different ways making use of the advantages of each design as needed.

To support this, future work in this area should include experiments that address the usability of such a multi-visualization view of the data. We need to find out whether the user's understanding of the data and task performance will benefit from this and if they will be able to form a consistent mental model of the underlying data in spite of the differences in visual appearance. Future work will also need to include the exploration of software architectures to support coordination and presentation of several visualizations, maintaining data consistency, and achieving acceptable performance in the rendering of visualizations.

Finally, we have gained some experience in implementing 3D information visualization designs. As for layout algorithms, it is crucial to be aware of the criteria we have for the visualization. Do we want to minimize the amount of unused 3D space? Do we want to minimize overlapping? Since an optimal layout is often impossible, we need to be aware that there is a tradeoff between different criteria.

Acknowledgments

We would like to thank Kåre Synnes, Peter Parnes, Serge Lachapelle, and Jan-Erik Moström (all of Luleå University of Technology) for advice and comments. Thanks also to Siv Wiss for proofreading.

References

[1] Keith Andrews. Visualizing Cyberspace: Information Visualization in the Harmony Internet Browser. In *Proceedings of Information Visualization*, pages 97–104. IEEE, 1995.

[2] Stuart K Card, George G Robertson, and William York. The WebBook and the Web Forager: An Information Workspace for the World-Wide Web. In *Proceedings of CHI'96*, pages 111–117. ACM, 1996.

[3] Matthew Chalmers, Robert Ingram, and Christoph Pfranger. Adding Imageability Features to Information Displays. In *Proceedings of UIST'96*, pages 33–39. ACM, 1996.

[4] Kenneth C Cox, Stephen G Eick, and Taosong He. 3D Geographic Network Displays. *Sigmod Record*, 25(4):50–54, December 1996.

[5] Steven Feiner and Clifford Beshers. Worlds within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds. In *Proceedings of UIST'90*, pages 76–83. ACM, 1990.

[6] Geoffrey S Hubona, Gregory W Shirah, and David G Fout. 3D Object Recognition with Motion. In *Extended Abstracts of CHI'97*, pages 345–346. ACM, 1997.

[7] Tamara Munzner and Paul Burchard. Visualizing the Structure of the World Wide Web in 3D Hyperbolic Space. In *Proceedings of VRML '95*, pages 33–38. ACM, 1995.

[8] Jun Rekimoto and Mark Green. The Information Cube: Using Transparency in 3D Information Visualization. In *Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS'93)*, pages 125–132, 1993.

[9] George G Robertson and Jock D Mackinlay. The Document Lens. In *Proceedings of UIST'93*, pages 101–108. ACM, 1993.

[10] George G Robertson, Jock D Mackinlay, and Stuart K Card. Cone Trees: Animated 3D Visualizations of Hierarchical Information. In *Proceedings of SIGCHI'91*, pages 189–194. ACM, 1991.

[11] M Sheelagh, T Carpendale, David J Cowperthwaite, and F David Fracchia. 3-Dimensional Pliable Surfaces: For the Effective Presentation of Visual Information. In *Proceedings of UIST'95*, pages 217–226. ACM, 1995.

[12] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of 1996 IEEE Visual Languages*, pages 336–343. IEEE, 1996.

[13] J Tesler and S Strasnick. FSN: 3D Information Landscapes, 1992. Man page entry for an unsupported but publically released system from Silicon Graphics, Inc.

[14] Colin Ware and Glenn Franck. Viewing a Graph in a Virtual Reality Display is Three Times as Good as a 2D Diagram. In *Proceedings of 1994 IEEE Visual Languages*, pages 182–183. IEEE, 1994.