10-1993

Visualizing Object Oriented Software in Three Dimensions

Colin Ware
University of New Hampshire, Durham, colin.ware@unh.edu

David Hui
University of New Brunswick

Glenn Franck
University of New Brunswick

Visualizing Object Oriented Software in Three Dimensions

Colin Ware, David Hui and Glenn Franck
Faculty of Computer Science
University of New Brunswick
P.O. Box 4400, Fredericton, NB.
E3B 5A3
cware@UNB.ca

Abstract

There is increasing evidence that it is possible to perceive and understand increasingly complex information systems if they are displayed as graphical objects in a three dimensional space. Object-oriented software provides an interesting test case - there is a natural mapping from software objects to visual objects. In this paper we explore two areas. 1) Information perception: we are running controlled experiments to determine empirically if our initial premise is valid; how much more (or less) can be understood in 3D than in 2D? 2) Layout: our strategy is to combine partially automatic layout with manual layout. This paper presents a brief overview of the project, the software architecture and some preliminary empirical results.

Introduction

The management and understanding of complex bodies of software remains one of the most challenging issues facing software engineering. Graphical techniques are in widespread use as an aid in many phases of the development process, from planning to code writing. Unfortunately the quantity of information is often so great that conventional 2D diagrams become extremely cluttered (see Consens 1991 for one example). There is growing evidence, some of which is described below, that representing diagrams in 3D can allow more complex information to be comprehended (Robertson et al, 1991, Xiao and Milgram, 1992, Ware, Arthur and Booth, 1993).

The goal of the research described here is to build a prototype system for the visualization and manipulation in 3D of substantial directed graphs with multiple attributes on the nodes and arcs. These graphs will be used to represent such things as software modules and software usage or inheritance relations. Subsidiary goals are advances in the interactive navigation and manipulation of 3D scenes and advances in the visual representation of multidimensional discrete data.

Our primary methodology is to build a proof of concept prototype called GraphVisualizer3D. This will be capable of displaying a substantial body of software comprising an existing commercial application and will be evaluated by professional programmers. A secondary methodology is to use focal studies in data representation which will be evaluated using techniques from visual psychophysics.

This paper is divided into three major subsections. In the first we describe some results relating to 3D visualization of networks of information. In the second we describe our strategy for interactive algorithm assisted graph layout, and in the third we describe the software architecture of the system.

Information visualization in 3D

Wickens suggested that creating visual objects can be the best way of understanding the relationship between variables in a multidimensional data object; this is called an object display (Wickens, 1991). Colour coding is a highly effective technique for multi-

The IBM contact for this paper is Arthur Ryman, Centre for Advanced Studies, IBM Canada Ltd. 895 Don Mills Road, North York, Ontario, M3C 1W3

dimensional discrete data display; colour dimensions can impart almost as much as spatial dimensions for representing clusters in a multidimensional data space (Ware and Beatty, 1988). We plan on exploiting the natural and obvious device of representing software objects as graphical objects. We will map attributes of the software objects (such at type, structure and size) to the graphical attributes of the 3D graphical objects (such as colour, shape and size).

Fish Tank VR

In recent years the concept of the virtual reality (VR) display has received considerable publicity. The system that is most commonly described has the display monitors mounted on a helmet, which immerses the user in the graphical environment. However, Deering (1992) has shown that a much higher quality VR image is obtainable using a more conventional monitor by coupling the perspective of the image to the users measured eye position, this localized the virtual scene to the vicinity of the monitor screen. Ware et al (1993) used the term Fish Tank Virtual Reality to describe this kind of display to distinguish it from the immersive VR systems. In the current project we are using Fish Tank VR for the visualization of object-oriented software. The basic elements of the system are shown in Figure 1.

Network visualization

The most significant previous work on the 3D visualization of information networks is the SemNet project of Fairchild et. al. (1988) which used 3D representation to allow users to visualize large knowledge bases as nodes and arcs in a three dimensional space. The present project borrows heavily from ideas present in SemNet both in the basic concept and in some of the details. However, we hope to go beyond this important pioneering work in a number of aspects. SemNet allowed users to manually place nodes according to semantic content, but much more emphasis was placed on automatic layout. In our work we place much more emphasis on manual layout. SemNet used uniformly sized panels to represent nodes and lines to represent arcs. We have gone substantially beyond this in the variety of nodes and arcs we can represent. SemNet used rotation to reveal the 3D structure of the displayed networks. We are using head-coupled stereo display to increase the 3D information available. There was no formal evaluation of SemNet and because of this it is difficult to ascertain how successful they were; both positive and negative opinions are available. We are engaged in a series of empirical evaluations of the validity of different representation schemes.

More recently, the Cone Tree concept developed at Xerox Parc has also played a seminal role in arousing interest in 3D versus 2D display (Robertson et al, 1991). The Cone Tree system allows for the display of tree structured graphs displaying all the children of a node in the form of a cone of information. The authors claim that as many as one thousand nodes may be displayable using Cone Trees without visual clutter - this is clearly more than could be contained in a 2D layout, although the Cone Trees require certain user manipulations to access some of the information.

Of particular relevance to the display of information networks is work that has shown that the number of errors in detecting paths through tree structures is substantially reduced if a 3D display method is used (Sollenberger and Milgram,1991; Ware, Arthur and Booth, 1993). Sollenberger and Milgram showed that both motion and stereopsis helped reduce errors in a path tracing task. In their motion conditions the stimulus pattern rocked back and forth about a vertical axis. They found that motion was more valuable than stereopsis in reducing errors. Ware et al's experiment used a similar task with a head coupled stereo display in which the perspective view was coupled to measured eye position of the observer. Although the motion was caused by head movement in Ware et al's display, the results were similar to those obtained previously by Sollenberger and Milgram.

There are a number of unanswered questions posed by the above studies. The first is the question of what kind of motion of the image is better for perceiving structure in information networks:

1) motion induced by perspective coupled to eye position,
2) automatic rotation of the object, or
3) motion caused by linking the user's hand movements to the object.

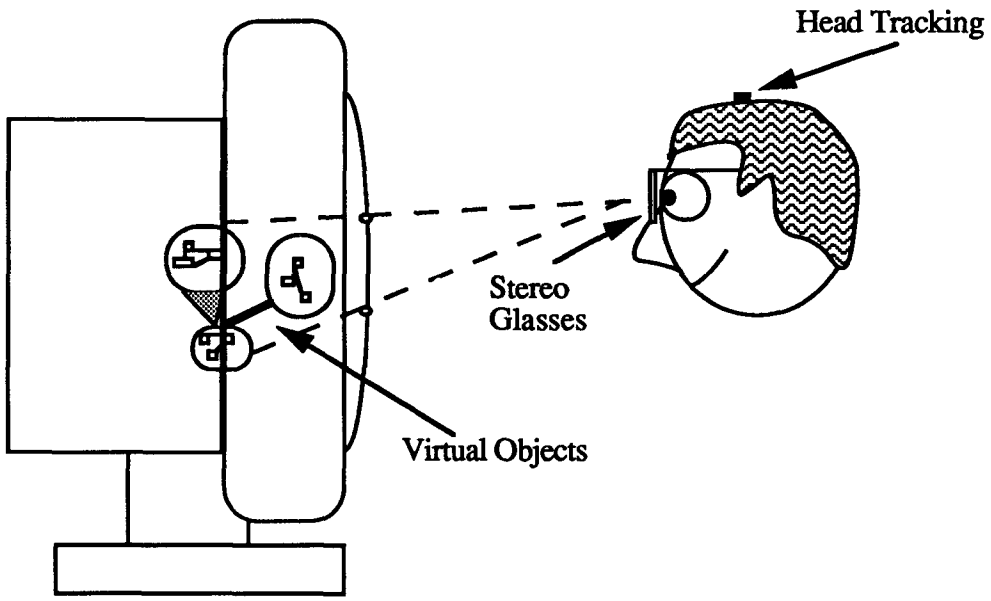

Figure 1. The preferred method for visualizing complex networks of information may be using head coupled stereo views. This results in a virtual 3D view of a scene placed in the vicinity of the monitor.

A second key question is whether the results of Sollenberger and Milgram and Ware et al generalize to directed graphs. This is important because tree layout is a relatively simple, well understood process, and it is trivial to lay out a pair of trees in a plane so that they do not overlap. Hence the visualization problem posed by the previous studies could easily be solved without resorting to a 3D display. This is not the case for an arbitrary directed graph where the layout problem is more difficult and the advantages of 3D visualization may be more pronounced. While much of the work on 2D layout of directed graphs has been directed at minimizing arc crossings, the algorithms are often complex and in some cases are NP hard. We hypothesize that 3D visualization will to some extent reduce the graph crossing problem because arcs will no longer appear in the plane of the screen. In part, the following experiment was designed to test this hypothesis.

Experimental study of 3D network visualization

Task description

The task presented to each subject in the experiment was to decide whether there was a path connecting two nodes which were highlighted in a randomly laid out graph. The computer generated a random, 3D graph consisting of 75 nodes, arranged in a simulated 18 cm^3 volume, with characteristics defined as follows.

The nodes were divided into three groups of 25: groups A and B each contained 25 leaf nodes, and group C which contained 25 connection nodes. Each of the nodes in group C was connected via arcs to two nodes in group A and to two nodes in group B. Thus there were a total of 100 connecting arcs. All nodes were placed at random within the working volume of the display space.

The graph remained the same throughout a given condition, but for each trial, a different pair of nodes was highlighted. Whether or not the nodes were connected was predetermined by the program, with half being connected, and half not. Connected and unconnected pairs of nodes came up in a (pseudo-) random order.

There were 18 trials conducted under each of nine conditions (described below). For each trial, one node from the A group and one from the B group

was highlighted, and the subject's task was to determine whether or not there was a path of length two that connected the two highlighted nodes

Conditions

There were nine conditions under which trials were conducted.

1) **2D:** no stereo, no rotation, two dimensions- the 3D graph was projected onto a 2D plane using an orthographic (parallel) projection by removing Z axis information, hence no overlap information was available.

2) **Perspective:** no stereo, no rotation, three dimensions - this is essentially the same task as in 1) above, except that the graph is displayed using a perspective projection with the depth cues of relative size and overlap/occlusion.

3) **Stereo:** no rotation, three dimension - this condition made use of a pair of StereoGraphics CrystalEyes LCD shutter glasses to provide auxiliary depth cues.

4) **Passive Rotation:** no stereo, three dimensions - here, the scene rotated at a constant angular velocity of 20 degrees/sec about the Y-axis. Stereo was not employed, so the only auxiliary depth cue was from relative movement.

5) **Stereo, passive rotation:** three dimensions - same as above except with stereo;

6) **Hand coupled:** no stereo, hand coupled, three dimensions - the subject could move the scene around with the mouse to get a better look from almost any angle; while providing a large degree of freedom, movement was slightly restricted to ±128° horizontal rotation (about Y-axis) and ±49.1° tilt (about x-axis);

7) **Stereo, hand coupled:** three dimensions - same as above, except with stereo.

8) **Head coupled:** three dimensions - the scene's perspective would change according to where the subjects head position was, thus always producing a proper perspective for the current head position, in relation to the center of the monitor screen;

9) **Stereo, head coupled:** three dimensions - same as above, except with stereo.

The subjects were made aware before each condition of what apparatus would be of use in the next set of trials (the subject wore the head tracker and stereo glasses regardless of the condition, in order to avoid effects related solely to wearing the equipment).

Each subject was given as much time as required to determine whether or not he or she believed there was a connection; after deciding, the subject would indicate his or her response by pressing a specified mouse button.

The entire experiment was repeated a second time on a different day to allow for the accumulation of more data. Before beginning the experiment each day, the subject was given a short warm-up which presented four trials under each of the same nine conditions.

There were a total of eleven subjects who took part in the experiment, five of whom had used similar apparatus and/or graphics systems before.

Results

The results are summarized in Figure 2. This shows that times were relatively uniform across conditions with one condition that stands out, the hand coupled condition without stereopsis; this had an average time of 12.4 seconds as compared to the overall average of 9.5 seconds. It is also evident that the use of stereo shortened response times by an average of 1.9 seconds (approximately 20%). This figure was obtained by comparing the conditions which only differed by the presence or absence of stereo.

The main difference between conditions is found in error rates. These ranged from 26% in the 2D condition down to a low of 6.1% in the stereo hand coupled condition. These data confirm previous studies which show that motion is more important than stereo in reducing errors (the average for the motion alone conditions was 11.4% whereas the average for the stereo alone condition was 15.4%) However they also show that the combination of stereo and motion is the most effective (average error 7.5%), and interestingly, they suggest that the method for producing the motion is not particularly important. Overall, the data do appear to support the hypothesis that 3D visualization reduces the errors due to arc crossings. However, there are also other possible explanations for this

decrease; such as the possibility that the *multiple views* afforded by motion and stereopsis allow for a reduced error rate despite the fact that in each view arc crossings existed.

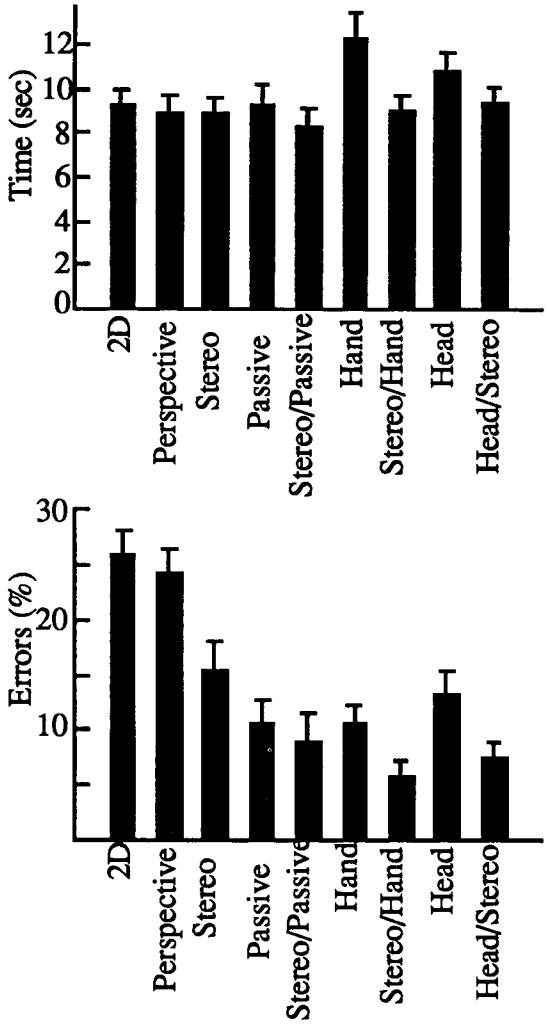

Figure 2. The results show small differences between times for the different conditions but large differences in error rates.

The practical conclusion which can be drawn from these results is that combining motion with a stereo view is a useful network visualization technique which may be useful in understanding the structure of object oriented code. Which type of motion should presumably depend on the application. For example, if the selection of objects is important then automatic rotation is not desirable because selecting moving objects is difficult. On the other hand, if head coupling is available then this would probably not interfere with 3D selection. This is because the motor control systems used for visually guided hand placement presumably have evolved to work in conjunction with simultaneous head motion.

Layout

The areas of graph layout in the plane and automatic graph drawing have attracted numerous researchers over the years. Eades and Tamassia () give an extensive survey of the topic complete with some 186 references. Eades and Xuemin () suggested three general criteria that consitute "good" drawing of directed graphs in the plane:

- avoid upward pointing arcs,
- distribute nodes evenly over the plane,
- minimize arc crossings.

We have only borrowed a small part from the large pool of literature on graph layout since we feel that much of this literature is irrelevant to the most crucial issue in graph layout, namely the problem of semantic clustering. The most important criterion for information layout will usually be the *meaning* of the nodes and arcs, and since most work on layout largely ignores this factor, we do not find it useful.

If a three dimensional software representation is to be useful as the primary high level interface to a significant body of software, it is reasonable to expect a project manager to spend many tens of hours working with the diagram, particularly if it is also the interface to the textual code itself and to the documentation. It is completely unreasonable to expect algorithms to achieve more than a first approximation to the layout of software entities in space because good layout is based, to a large extent, on high level semantics which cannot, at present, be formalized. Therefore, the placement used in the 3D interaction package will be based on a combination of algorithmic constraints and manual layout. Our strategy is to combine the most elementary graph layout algorithms with a highly interactive system. Some of the criteria which we think may be important are listed

below, together with some of the interactive requirements.

C1: Nodes should be laid out in a top-down fashion in horizontal layers.

C2: Arcs should point downward or horizontally on each layer. There should be a minimum of upward pointing arcs.

C3: Nodes should be laid out in a regular grid pattern within layers.

C4: It should be possible to drag and drop nodes in order to move them to new positions.

C5: Nodes should be constrained by gravity points at the grid intersections in order to assist in the manual layout.

C6: Anchors should be supported to prevent automatic layout methods from affecting manual work that has already been done.

C6: Nesting nodes within nodes should be supported.

C7: Nested nodes should be laid out within the parent nodes according to the same criteria that are used for the parents.

Criterion C1 can be achieved by obtaining a topological sort of the nodes based on a particular arc relation. The prior requirement of topological sorting is that the digraph must be *acyclic* or a DAG and this is commonly achieved by reversing certain arcs. Unfortunately optimal cycle elimination is NP-hard (Gary and Johnson, 1979). However we use a simple and fast

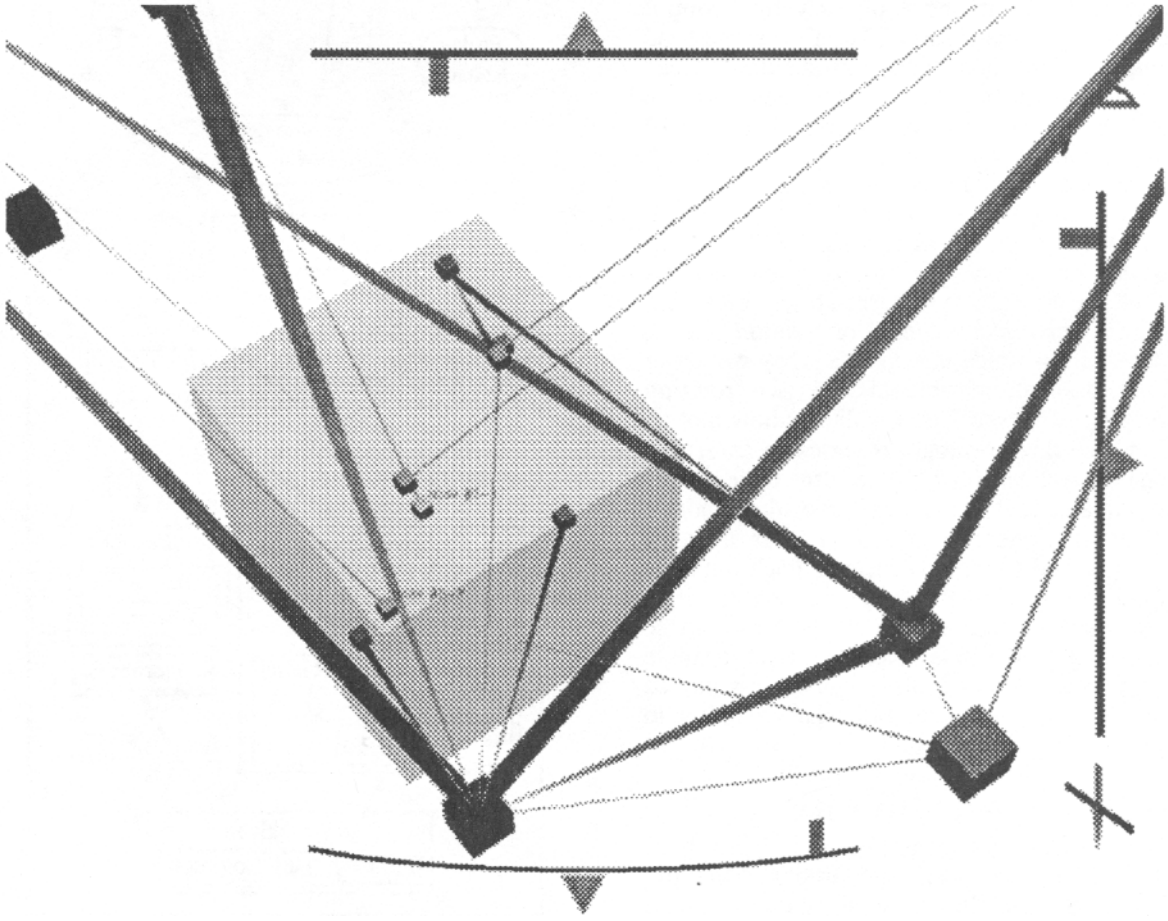

Figure 3. A view of some data objects stored in the system. The objects at the right, top and bottom of the picture are the 3D widgets described in the text.

heuristic based on the depth first search technique to explore the digraph with respect to a chosen root node; in this process previously visited nodes are marked and thus backarcs can be ignored (a similar method can be found in Eades and Xuemin, 1989).

Nesting nodes within nodes will also be done somewhat automatically; if a node is designated as nested then it will automatically be drawn smaller and inside the parent node.

The rest of the layout criteria will be defined by user interactions, although these will often be constrained by the system. One important part of the interactive layout system is the set of 3D widgets used to control the user's view of the graph.

3D Widgets

There are two methods of viewpoint control employed in this system. The first is the head coupling described in the introduction. Head coupling is useful in that it provides a natural method for changing the viewpoint, but it is also limited. Moreover we wish to have a system in which head coupling is optional, and thus the system should be fully functional without it. Therefore, we have designed a set of 3D widgets for this and similar applications. These widgets allow for the scene to be rotated and translated using either velocity control or position control. They are illustrated in Figure 3. They consist of three translation widgets and two rotation widgets. The translation widgets allow motion along the three orthogonal cartesian axes. The rotation widgets allow for the scene to be rotated around a central vertical axis and tilted about a horizontal axis. Each widget has a small rectangle and a triangle attached. Each rectangle is a direct manipulation position control; when it is selected the scene moves with the mouse according to the axis selected. Each triangle provides a velocity control; when it is moved the scene starts to rotate or translate according to the amount of displacement.

Software architecture

The proof of concept prototype is being built using C++ on a Silicon Graphics Indigo2 Extreme workstation or equipped with a high performance graphics subsystem capable of approximately 600,000 filled 3D polygons/second. The system is being written using SGI's graphics library (GL) for all 3D graphics, and X-Windows with Motif widgets for the standard menu and window interface.

In the center of the main box in Figure 4 is an elliptical shape denoting the Prioritized Adjacency List (PAL). This represents the central internal data structure in the system. For moderately sized systems this will contain all of the information that can be displayed in a graph in addition to the layout information. Only a subset of the information stored in this structure will be displayed at a given time. Aside from this there is a file I/O module to transfer data into and out of the system, and two modules relating to system visualization and layout respectively.

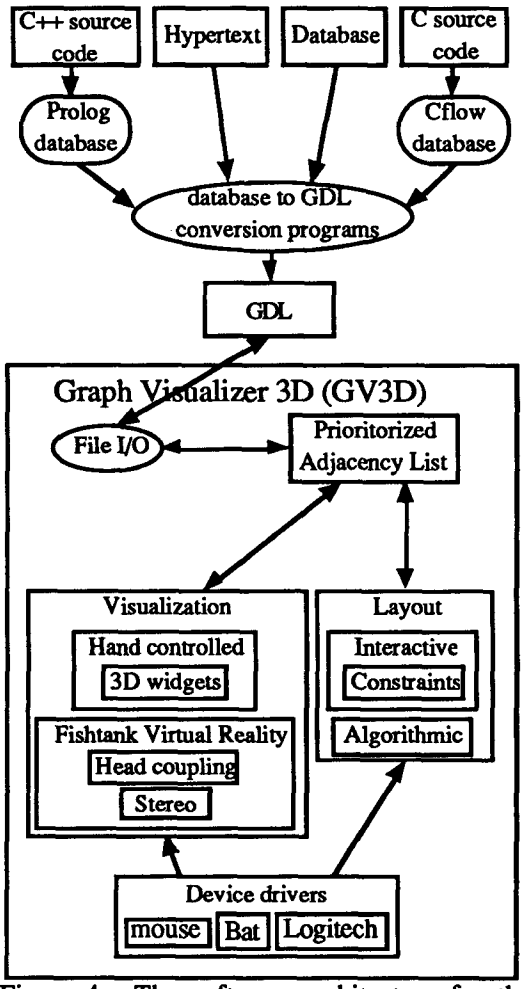

Figure 4. The software architecture for the GraphVisualizer3D system.

Outside of the main GV3D box is an ellipse denoting the Graph Description Language (GDL). This is a language that we have designed to allow for the permanent storage of graph structure information and also information relating to layout and visual representation of a graph. This language is also the primary interface to the various diverse application systems we plan to support. In the case of C++ code, we plan to use the IBM compiler for the RISC System 6000 which constructs an internal Prolog data base containing extensive information about the code structure (Jarvey et al, 1992). We will extract the information we need by means of Prolog queries and transform the results into GDL. In order to visualize another application, hypertext for example, it will be necessary to transform the data structure into the GDL.

Conclusion

The system described here is still in its early stages of development. By the time this paper appears we hope to have version 1.0 complete and operational. Nevertheless, we have already obtained some encouraging results in providing empirical evidence that 3D graph visualization can substantially reduces error rates over visualization in only two dimensions. In the next two years we hope to show that this mode of viewing has substantial benefits for understanding object oriented software.

This paper describes only part of the system. There are other large areas which have not been covered or have just been hinted at. One of these areas is the method by which the users will query the system to obtain information that is not currently displayed, or to highlight a certain subset of the arcs. Another is the sequence of steps whereby the code is analyzed for the purposes of display. Yet another is the method for displaying textual information. Some of the decisions about what to include in the system have not yet been made. In other cases information has been left out for reasons of brevity.

About the authors

Colin Ware is an assistant professor of computer science at the University of New Brunswick. His research interests include data visualization and visual user interfaces.

David Hui has nearly completed a masters degree in computer science at the University of New Brunswick in the area of software visualization. He is currently employed at IBM Toronto Labs.

Glenn Franck is working as a software engineer on the software visualization project. He has a BSc in computer science from Dalhousie University.

Acknowledgments

The primary source of support for this project is an NSERC strategic grant. We are grateful to Arthur Ryman at IBM Toronto Labs for his encouragement and support and for his assistance in providing code analysis tools. We also thank Tim Dudley at Bell Northern Research for providing the initial impetus to start this project and for ongoing enthusiastic support.

References

Bier, E.A. (1990) Snap-Dragging in Three Dimensions, Computer Graphics,24(2)

Consens, M., Medelzon, A., and Ryman, A. (1991) Visualizing and Querying Software Structures. IBM Canada Technical report TR 74.053

Cox, P.T., Giles, F.R., and Pietrzykowski, T., Prograph: a step towards liberating programming from textual conditioning. IEEE Workshop on Visual Languages. 1989, Rome. Proceedings pp 150-156.

Deering, M. (1992) High resolution virtual reality. Computer Graphics, 26,2,195-202.

Eades, P. and Tamassia, R. Algorithms for drawing graphs: An annotated bibliography. Technical report. Department of Computer Science. University of Illinois at Urbana-Champaign, Urbana-Champaign. Illinois, June 1990.

Eades and Xuemin. How to draw a directed graph. In IEEE Workshop on Visual Languages. 13-17, 1989.

Fairchild, K.M., Poltrock, S.E. and Furnas, G.W. (1988) SemNet: Three-Dimensional Graphic Representations of Large Knowledge Bases. In Cognitive Science ad Its Applications for Human-Computer Interaction. Ed Raymond Guindon Lawrence Erlbaum. 201-233.

Garey, M.R. and Johnson, D.S. (1979) Computers and Intractability - A Guide to the Theory of NP-Completeness, Freeman, 1979.

Jarvey, S., Mitsui, H. and Nakamura, H. (1992) Architecture of the xl c++ browser. In proceedings fo the 1992 CAS Conference. Centre for Advanced Studies, IBM Canada Ltd. November, 1992.

Lieberman, H. (1989) A Three-Dimensional Representation for Program Execution,IEEE Workshop on Visual Languages, Proceedings, 111-116.

Mackiney, J.D., Card, S.K and Robertson, G.G. (1990) SIGGRAPH '90 Conference Proceedings. Computer Graphics, 24, 4, 171-176.

Mariani, J.A., and Lougher, R. TripleSpace: an experiment in a 3D graphical interface to a binary relational database, Interacting with Computers, 4(2) 1992 147-162

Messinger, E.B. (1991) A Divide and Conquer Algorithm for the Automatic Layout of Large Directed Graphs. IEEE Transactions on Systems, Man and Cybernetics, 21, 1, 1-12

Purcell, D.G. and Stewart, A.L. (1991) The object detection effect: Configuration enhances perception, Perception and Psychophysics, 50(3) 215-224.

Robertson, G.G., Mackinlay, J.D and Card, S. K. (1991) Cone Trees: Animated 3D Visualizations of Hierarchical Information. CHI'91 Proceedings. 189-194.

Sollenberger, R.L. and Milgram, P. (1991) A comparative Study of Rotational and Stereoscopic Computer Graphic Depth Cues. Proceedings of the Human Factors Society Annual Meeting, 1452-1456.

Tufte, E. (1990) Envisioning information. Graphics Press, Connecticut.

Ware, C., Arthur., and Booth, K.S. Fish Tank Virtual Reality. Proceedings of INTERCHI'93. 37-42.

Wickens, C.D. (1991) Engineering Psychology and Human Performance, (2nd ed.) New York: Harper Collins.

Wilhelms, J and Skinner, R. (1989) An Interactive Approach to Behavioral Control. Graphics Interface, Proceedings, 1-8.

Xiao, Y. and Milgram (1992). Visualization of Large Netwoks in 3-D Space: Issues in Implementation and Experimental Evaluation. Proceedings of the 1992 CAS conference. 247-258.