A Prototypical 3D Graphical Visualizer for Object-Oriented Systems

Hua Qingyi (华庆一)*, Heinz-Dieter Böcker and Dong Cheng (董 诚)

Integrated Publication and Information Systems Institute (IPSI)
German National Research Center for Computer Science (GMD)
Dolivostr.15, 64293 Darmstadt, F.R. Germany

Received March 22, 1995; revised September 5, 1995.

Abstract

This paper describes a framework for visualizing object-oriented systems within a 3D interactive environment. The 3D visualizer represents the structure of a program as Cylinder Net that simultaneously specifies two relationships between objects within 3D virtual space. Additionally, it represents additional relationships on demand when objects are moved into local focus. The 3D visualizer is implemented using a 3D graphics toolkit, TOAST, that implements 3D widgets and 3D graphics to ease the programming task for 3D visualization.

Keywords: Object-oriented system, program visualization, 3D graphics.

1 Introduction

An important feature of object-oriented paradigm is to facilitate code reuse. An object is essentially a reusable, pluggable module. Reusing objects depends on understanding semantics of objects and the structure between objects. The tasks of understanding are usually difficult, because numerous classes are organized in multi-dimensional hierarchies and networks with complex inheritance and various relationships, and small chunks of functionality are usually distributed in multiple classes within an object-oriented system.

Program visualization can be considered as the graphical display of information about a program. The graphics are used to illustrate some aspect of the program or its run-time execution[1]. The graphics can provide much more information than the stream of text, making data and algorithms easier to understand. Moreover, the graphics can be interactive and animated, enabling a user to query for relevant information, and to view the flow of information continuously. Tools for visualizing object-oriented systems can be classified as two categories: those for static code structure (e.g. class, inheritance browsers and code inspectors[2,3]) and those for dynamic behavior (e.g. message tracers and state debuggers[4-8]).

Most of visual tools are done using 2D bitmap technologies. 2D diagrams are used to represent classes or objects and the relationships between them: a box icon

* Author's current address: Department of Computer Science, Northwest University, Xi'an 710069, P.R. China.
is typically used to represent a class or an object, and arcs are used to specify various relationships. However, two-dimensional output media such as paper or bitmap display can generally represent two independent relations\cite{7}. To reduce the complexity of representation, 2D visualization paradigms are thus applied that use a multi-window environment, in which classes or objects and different relationships are separated into subdiagrams and are shown in different windows. This will limit their expressive capability and increase mental effort for understanding. Some limitations are: the multi-diagram paradigms are not consistent with the diagrams used in analysis and design, a user needs extra mental effort to shift from one representation to another; relationships between subdiagrams cannot be created between windows, because traditional desktop technologies do not provide suitable methods to reference contents of windows across windows; and those paradigms force different mental models upon the user through each subdiagram, the unifying mental model, however, has to be reconstructed with constraints by the user to thoroughly understand the structure and behavior of an object-oriented system.

Emerging technologies for 3D visualization and interactive animation offer a potential solution to the improvement of the visualization for object-oriented systems. Some pioneering systems have revealed some advantages of 3D displays over 2D displays, for example, in the area of scientific visualization and information visualization.

This paper describes a 3D visualization framework for object-oriented programs. We show how those limitations mentioned above can be reduced by applying the framework. Our prime goal is to reveal the static states of programs written in C++ through a 3D visualizer. The visualizer represents the structure of a program as a Cylinder Net, and lets the user directly manipulate it to search and query the information. The cylinder net enables the user to see the two most important relationships (class interface and class hierarchy) simultaneously, and shows other relationships (e.g. parameter using) as the user brings it into local focus. Thus, the 3D visualizing paradigm represents the structure of an object-oriented program in a 3D virtual space. We have implemented a 3D graphical toolkit, TOAST (TOwards Application-Specific Toolkit), which involves a set of 3D interactive graphical objects and 3D widgets as building blocks of the visualizer. In our environment, a 3D navigator is used for the implementation of viewpoint (camera) navigation.

2 3D Visualization Paradigm

Booch points out that class diagrams are usually applied for the logical view of static semantics of object-oriented programs\cite{8}. The diagrams are used to show class interfaces, hierarchies, and other basic relationships. The two essential elements of a class diagram are classes and their basic relationships. In the class diagram 2D icons are used to represent the coarse or detailed structures of classes, and different kinds of links with labels are used to illustrate different relationships among classes. It is easy to see attributes and methods in all classes, inheritances, and uses among all classes as a whole. As mentioned above, the class diagram cannot smoothly be