Chapter 1 Introduction

Developing efficient, bug free software is much more difficult and time consuming than before because the complexity of software has grown significantly than before. We need some tools to help this problem. Debugger is one of solutions for this problem. As users debugging programs with debugger, they need some experiences and have to repeat steps such as set breakpoints, run the program to hit the breakpoints, and watch the value of variables until the cause of defect is found.

Debugging is the dirty little secret of computer science. Even there are faster computers, networking, simpler GUI, IDE (Integration Development Environment) etc., we still face some challenges: debugging a complex program is not easy. Programmers need to gather enough information from variables, data structures to, pinpoint the causes of program errors and then try to fix them. When the size of program is small, it is possible for programmers to discover the bugs without much help from tools. But, as the sizes of programs grow, programmers become incapable of processing large amount of information. They need tools to help them understand the logic of complex programs. Unfortunately, current debugger tools are inadequate for their need.
Traditional debuggers only provide textual display for programmers to display the debugging information. It is not user-friendly and is difficult to get the whole picture of complex data structures. Although programmers can still get the essential information from textual information, it could be much more helpful if a graphic representation can be constructed from the textual information.

There are some systems which can show data as 2D graphs. For example, the instructional system Amethyst [14] displays the call stack and data values and LDBX [21] extends DBX debugger to show data structure as graphs. These systems provide more useful information than traditional textual debuggers but it is still not enough. It is difficult to arrange all large number of elements in screen when these elements need to be drawn on 2D layout. If there is a 3D environment, it is possible to place all elements into the virtual world and provide more natural computer human interaction with haptic equipments (i.e., virtual reality gears).

DIVINE [17] is a 3D debugging visualization tool we are developing. In the past, it provides a visualization of link-list structure in 3D environment and interactions to manipulate the elements in the screen. The visualization system DIVINE communicates with HILCADT [11], a debugger for Java language. HILCADT provides a high-level debugging language that is more expressive and flexible than the primitive language of debuggers. It can be used to explore the temporal relation of
variable’s value in a period of time and for further analysis, and can display the debugging information on its panel or in 3D environment with DIVINE. DIVINE can send requests to HILCADT to get the states of a Java program when it is paused by breakpoint. Later, DIVINE can visualize the information as a rooted tree on the screen.

Current DIVINE can display the information with the only one way and that is not easy to let the users to understand complicated data structure. In this thesis, we strengthen DIVINE’s by supporting the concept of metaphors and enhancing more human natural interactions between users and the system. A metaphor is an imaginative way of describing something by referring to something else which is the same in a particular way. In other words, metaphor is a way of presenting a set of data. For example, to data structure like graph, we are familiar with the metaphor which use circle to represent a node and an edge to represent the pointers between nodes. It is helpful to find the problem if programmers can visualize the information using appropriate metaphors. In some cases, different metaphors may reveal different aspects of the data. Well-designed, useful interactions are helpful to use DIVINE more easily and useful to understand data structure when amount of visualizing data is growing fast. The goal of this thesis is to study and implement the 3D metaphors of data structures in computer programs.
An overview of visualization and debugger techniques and a basic concept of 3D scene are described in Chapter 2. Chapter 3 describes the architecture and the implementations of DIVINE system. An example module of a metaphor is described in Chapter 4. Chapter 5 is the conclusion.