A Program Visualization Tool for Program Comprehension

Mikio Ohki Yasuo Hosaka
Nippon Institute of Technology
ohki@nit.ac.jp y-hosaka@ezweb.ne.jp

Abstract

At the beginning stage of programming education, comprehending program logics plays a more important role than capturing knowledge of a specific program language. In this paper, the authors describe an outline of a visualization tool that animates the actions of a program by adding simple annotations to the variables defined in a program, and show several ideas to improve the usability of this tool. The authors also discuss the effectiveness of program visualization.

1. Introduction

It is an undeniable fact that each student is born with a different level of inherent programming ability. Teachers who are responsible to elementary programming education in a university make every effort to overcome these differences among individual students. For example, they usually use diagrams or animations to explain the basic programming concepts and algorithms. However these visualization methods have not been proved to effectively improve the depth of a student's understanding level. The SPI (Synthetic Personality Inventory) examination, which has been used by companies to measure an examinee's aptitude as a programmer, gives a suggestion to this question. The SPI examination is designed to measure one's ability both in the language field and non-language field (mainly for image manipulation) and to judge one's total aptitude as a programmer. If the results of the SPI examination show any positive correlation between one's abilities in the non-language field and one's aptitude as a programmer, a student who can easily understand the basic programming concepts and algorithms and is judged to be highly suitable to a programmer should get higher scores in the non-language field examination. On the other hand, those who cannot understand the concepts and algorithms should show lower scores in the non-language field examination. Therefore providing the students who are less able to manipulate images with a tool that presents diagrammatic images or animations to offer assistance in understanding subjects of non-language fields helps those students to understand the basic programming concepts and algorithms.

This paper consists of two sections and is mainly devoted to prove the assumption mentioned above. In one section, we discuss the examination results with regard to the correlation between the depth of program understanding and the effects of program visualization. In the other section, we describe the features and functions of the program visualization tool, PAVI, which we developed to prove the assumption.

2. Relationship between Image Operation Ability and Comprehension Depth of Programming Concept

The SPI examination is designed to measure one's aptitude as a programmer and consists of the questions related to the language field and those related to the non-language field. Questions in the former category include selection of related sentences, rearrangement of sentences, sorting of strings and comparison of sentences, and those in the latter category include sorting of values, comparison of sums, identification of rotated figure patterns, identification of cross sections of figures and identification of rotated shapes.

The authors randomly selected 17 students from freshmen who were learning the specifications and concepts of a programming language (e.g., assignment, arithmetic and logical operations, type conversion, one dimensional array, control statements, loop and nested loops, pointer, etc.) for the first time and conducted an examination to observe whether there was a correlation between the test scores for judging one's program understanding level and the results of the SPI test. Since the examinees were less experienced in programming, influence of accumulated programming experiences was negligible. In the result of the examination, statistically significant correlations were observed as shown in Table 1.

Table 1. The correlation between the results of the SPI test and the program understanding level

SPI terms / Program Concepts	Arrange-ment of Order of Terms in Sentences	Applica-tion of Arithmetic Calculati-on	Identifi-cation of Rotated Shapes	SPI Total Score s
Assignment Statement	0.07	0.05	0.48*	0.12
Logical Operation	0.53**	0.34	0.26	0.43*
Array Definition	0.22	0.10	0.41*	0.25
Loop Control Statement	0.34	0.35	0.39*	0.32
Array Access	0.37	0.42*	0.25	0.43*
Nested Loop	0.31	0.39*	0.30	0.46*

Total score: comprehension level of prog. concepts	0.33	0.30	0.39*	0.40*

(Note 1) *: 5% significant level of Pearson correlation coeff.
(Note 2) **: 1% significant level of Pearson correlation coeff.

Among the SPI examination items, the image manipulation ability shows significant correlation with the understanding level of programming concepts. The result indicates that the students with higher image manipulation abilities can achieve a deeper understanding level of programming concepts (especially for the assignment, array definition, and loop control statements)[1].

3. Overview of PAVI

Base on the positive correlation between the image manipulation ability and the understanding level of programming concepts, we assumed that visualization could assist those students of lower program understanding levels in grasping image manipulations and then understanding the program concepts. Thus, we developed the program behavior visualization interpreter, PAVI (Program Action Visualization Interpreter), to help students easily understand and verify the program behavior.

3.1 Features of PAVI

(1) Animation exclusively focused on the assignment operations

PAVI is a program visualization tool for elementary training of C language programming, and interprets and visualizes program behavior. It represents and monitors variables, arrays, and pointers as three-dimensional objects, and it animates the actions of an object whenever an assignment operation is detected. The visualization scope is limited to the assignment operations due to the following three reasons. (a) As shown in Table 1, understanding the concept of assignment operation correlates to image manipulation. (b) The side effects of assignment operations make it difficult to understand the procedures in a program. (c) The concept of pointer is one of the major obstacles in elementary programming education.

(2) Using tags to annotate program source code

PAVI uses special annotation statements or lines (i.e., PAVI tags) to specify target variables or pointers to be visualized as three-dimensional objects and to define scopes and styles for visualization, as shown in Figure 2. As a markup language used to specify visualization attributes for C programs, the PAVI tags have the following merits.

(a) Students need not modify existing program source code. Simply adding PAVI tags as annotations is enough to visualize the program behavior.
(b) Since the tags are written as annotation lines, they are compatible with the C language compilers. As a result, any part of program behavior can be visualized and examined at any timing.
(c) Since the user is allowed to specify a specific range of program source code to be visualized as needed, only specified part of a large program can be conveniently visualized.

```
int data[10] =   {  24,10,8, 4,5,15,3,6,12,9 }
      /*$  coord=(-20,  0,  -30)  factor=3  color  =  0xaa0000
          shape= cylinder width=2    $*/    ;
void quickSort(int data[], int l, int r) {
  int i, j, v;
  int swap /*$ coord=(20,0,-1) factor=3 color= 0xffffff  shape=
                cube    width= 3   $*/   ;
  if(r> l)   {
      v = data[r];   i = l - 1;   j = r;
      do  {
          while(data[++i] < v);
          while(data[--j] > v);
          if(i < j)   {
                //$ traceBegin assign = line step 10
                swap = data[i];          //$ parBegin
                    data[i] = data[j];
                    data[j] = swap;   //$ parEnd
              //$ traceEnd
          }
      } while(i < j);
       //$ traceBegin assign = line step 10
       swap = data[i];          //$ parBegin
            data[i] = data[r];
            data[r] = swap;  //$ parEnd
      //$ traceEnd
    quickSort(data, l, i-1);
    quickSort(data, i+1, r);
}  }
```

Figure 2. Sample PAVI Tags

(3) Interactive debugging environment

Since the PAVI environment is interactive, the user can use the mouse to point to and click on the values or types of variables, arrays, and pointers represented as 3D objects. Thus, this feature allows the user to use PAVI as a debugging tool.

3.2 Types and Attributes of Primary PAVI Tags

(1) The tag used to assign attributes to visualized objects

This tag is used to specify the variables, arrays, and pointers to be visualized and to assign attributes to those visualized objects. The following attributes can be specified.

(a) viewPoint: The viewpoint from which the user sees the three-dimensional space.
(b) coordinate: The location of the object that matches the data items in the three-dimensional space.
(c) color: The color of the object.
(d) shape: The shape of the object (cylinder, cube and sphere)
(e) height: width: depth: The upper limit of the object height, width, depth.
(f) factor: The scaling factor of magnitude of the object

(2) The tags used to specify the scope of visualization

These tags are used to specify the range of program source code to be visualized and the type of object actions. The following attributes can be specified.

(a) traceBegin, traceEnd: Assignment and pointer operations executed in the range between these key words are displayed as animated objects. They move sequentially.

(b) parBegin, parEnd: Assignment and pointer operations executed in the range between these key words are displayed as animated objects. They move simultaneously.

(c) trace Style: Specifies the style of trace line along which the object moves from the start point to the end point. The options include a straight line, circle and rectangular.

(3) Other Specifications

In addition, the "include" and "define" specifications can be used like as the C language.

3.3 Operation and Display

Like other algorithm visualization tools, such as BLASA and Zeus, PAVI supports multiple synchronized views. Figure 3 shows the Source Code Viewer, which is used to display the location of the source code step currently interpreted, and Figure 4 shows the Animation Viewer, which is used to animate the 3D actions of assignment operations.

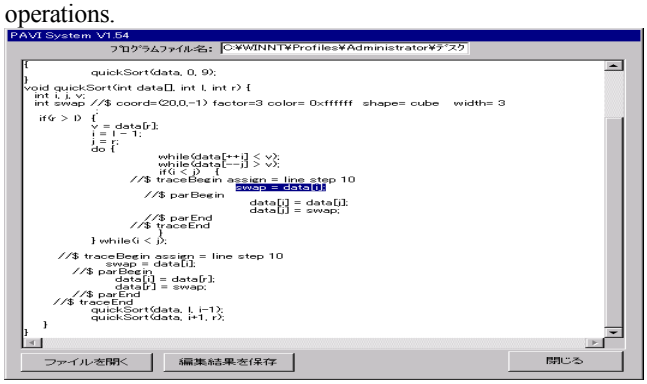

Figure 3. Source code viewer (The currently evaluated line is highlighted) .

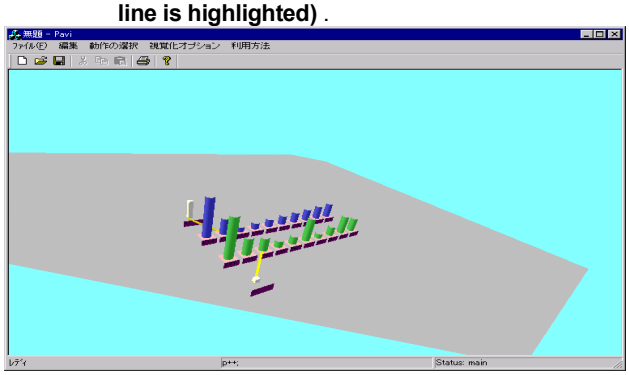

Figure 4. Animation viewer: A program that contains pointer operations is visualized

4. Comparison with Related Researches

Zeus[2] is an interactive visualization tool, which uses multiple viewers. Lens[3] is a tool that allows the user to control the animations within the various space and the shape of objects in detail. Compared with the above tools, PAVI provides simplified control of manipulation and visualization. However, one of the objectives of PAVI is to allow novice programmers to visualize the actions of their programs by simply adding tags to the programs. With regard to this point, adding tags as annotations is a more powerful method compared with the above tools.

5. Conclusion

We conducted an experiment to observe if the use of PAVI could enhance the space identification ability and then improve comprehension level of program concepts and algorithms. In this experiment, we divided the 17 students mentioned in Section 2 into two groups and observed and compared the comprehension level of each member of the two groups, provided that the members in one group used PAVI and those in the other group did not. In the experiment, the students were requested to fill the blank spaces with index operations of an array, break statements, assignment statements or data exchanges operations.

We observed significant differences in the results of algorithm questions between the two groups as shown in Table 2. Achieving higher scores was reasonable for the PAVI group since PAVI assists the students in understanding the sorting process of the array elements.

Table 2. Differences between understanding levels of the algorithm problems achieved with PAVI and without PAVI

Algorithm Problem	With PAVI Standard deviation	Without PAVI Standard deviation	t-value
Array operation	5.63	6.43	3.35**
Break	7.44	8.43	2.58*
Assignment Statement, Data exchange	7.29	4.86	2.54*
Total	30.56	33.00	2.90*

(Note 1) *: 5% significant level of t-test
(Note 2)**: 1% significant level of t-test

6. Refences

[1] M. Ohki, "The Effectiveness of a Program Visualization Tool in the Beginning Programming Education", *Proc. of IPSJ Winter Workshop on Software Engineering*, 2003, pp.85-86

[2] M. Brown, "Zeus: A system for algorithm animation and multi-view editing," *Proc. of IEEE Workshop on Visual Languages*, 1991, pp. 4-9.

[3] Stasko J.T., Mukherjea S., "Toward Visual Debugging: Integrating Algorithm Animation Capabilities within a Source Level Debugger", ACM Transactions on Human-Computer Interaction, Vol. 1, No. 3, September 1994, pp. 215-244