Rodrigo, M. M. T. et al. (Eds.) (2021). Proceedings of the 29th International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

Development of Mapping Function between Variable Value and Object Properties for Program Behavior Visualization Tool TEDViT

Hiroki SOMA[a], Satoru KOGURE[b], Yasuhiro NOGUCHI[b], Koichi YAMASHITA[c], Raiya YAMAMOTO[d], Tatsuhiro KONISHI[b] & Yukihiro ITOH[e]
[a]*Graduate School of Integrated Science and Technology, Shizuoka University, Japan*
[b]*Faculty of Informatic, Shizuoka University, Japan*
[c]*Faculty of Business Administration, Tokoha University, Japan*
[d]*Faculty of Engineering, Sanyo-Onoda City University, Japan*
[e]*Shizuoka University, Japan*
* soma.hiroki.17@shizuoka.ac.jp

Abstract: TEDViT is a program behavior visualization system that can reflect explanatory intentions of a teacher by visualizing it in a target world. This system can dynamically reflect changes in variables caused by program behavior in terms of numerical values on objects. However, it is difficult to map the variable values to the position, size, and color of the object. In this study, we implemented a function in TEDViT to dynamically map variable values to object properties. We report an experimental evaluation using 10 subjects.

Keywords: Programming learning support, program behavior visualization

1. Introduction

It is crucial to visualize the behavior of an algorithm to understand it. Therefore, several systems have been developed to visualize program behavior (Moreno, Myller, & Sutinen, 2004) (Matsumura, Daisukey, & He, 2009). However, the visualization method used in these systems is fixed by the developer, and the teacher cannot change the visualization method.

Therefore, a behavior visualization tool for C language programs called TEDViT (Yamashita et al., 2015) was developed to solve this problem. It allows the teacher to set the properties of objects associated with variables by specifying the behavior visualization method by rules, and drawing objects according to the intention of the teacher. The teacher can set the properties of the objects associated with the variables by specifying rules to visualize the behavior. The numbers and characters displayed on the object are automatically updated with a change in the value of the variable. This is known as mapping function.

As mentioned above, the objects visualized by TEDViT have properties (drawing color, display position) that can be specified by the teacher as a part of the visualization rules. However, TEDViT has the following problems.

Problem 1: There size of the objects cannot be specified by the teacher.
Problem 2: It is difficult to change the properties of the objects according to the values of the variables.

To solve *Problem 1*, we created an object whose size can be specified and teachers can generate it by rules. To solve *Problem 2*, we developed a mechanism that can dynamically map each variable value to the property of each object such as size, color, and position. In this mechanism, teachers can set visualization rules for mapping variable values to object properties. Therefore, the improved system can allow the teacher to provide a detailed visualization and a wider range of possibilities to reproduce the intended drawing. Moreover, using the new rules, the teacher can reduce the time required to create rules for changing the properties. The purpose of this research is to implement these two mechanisms to solve the abovementioned two problems.

2. Previous Works

The GUI of TEDViT consists of a source code section that displays the source code. It is an implementation view that displays a memory image corresponding to the program behavior, and a conceptual view that visualizes the behavior of the program. Additionally, TEDViT can automatically display the value of each variable in the program at the time of execution in the object. In this case, it is necessary to describe the visualization rules to draw, highlight, and display messages. Examples of visualization rules are shown below.

```
rule,j,state==1,create,obj_i,square,i,x1,y0,blue,white,black
```

In this example, when the statement 1 is executed, an object `obj_i` is created for variable `i` at coordinates (`x1,y0`). This object type is `square`, with a `blue` outline, a `white` background, and a `black` string.

3. Development of Mapping Function between Variable-Values and Object Properties

3.1 Create a New Object for Problem 1

The size of the objects in TEDViT cannot be specified by the teacher. The knapsack problem is an example of an algorithm that utilizes size for visualization. It selects the object with the greatest value within a limited capacity. However, we require a representation method in which the size can be changed arbitrarily to visualize capacity. Therefore, we solved *Problem 1* by creating an object whose size can be specified. Based on the existing rectangular objects, we created three types of objects that can specify only horizontal size, only vertical size, and both, respectively.

3.2 Implementing Functions to Reference Variable Values

TEDViT cannot dynamically map variable values to the properties an of object. If the teachers want to change the property of an object in response to a change in a variable value, they have to write additional rules for each change. The knapsack problem and sorting discussed in *Problem 1* can be visualized by mapping specific numerical values or variable values to properties. Other visualization tools (Halim, et al., 2012) (algorithm-visualizer, 2021) visualize the process of sorting by changing the bar graph. This is represented by changing the "size" of the object depending on the value of the variable. Therefore, by mapping variable values and numerical values to the properties of existing objects and the objects created in *Problem 1*, it was possible to automatically follow the property values to the variable values. This dramatically reduced the cost of writing rules and solved *Problem 2*. The target properties of the mapping function in this research were "size," "color," and "coordinates."

Size: As a solution to *Problem 1*, we created an object that can specify the size. Therefore, the size can be changed dynamically using this object according to the value of the variable being visualized. In addition, we added an item to the rules to specify a numerical value for the magnification of the change in height and width to ensure that the object can be drawn with the magnification applied.
Colors: The rules in TEDViT allow the user to specify the colors of the background and border of an object, and the text colors of variable values. The first is to convert the variable value into a color parameter. The second is to change the color discrete by a conditional expression, by specifying a conditional expression and a color. Currently, up to two conditional expressions can be specified, and variable values can be used in conditional expressions.
Coordinates: In TEDViT, the position of an object can be specified using the Cartesian coordinate system. By specifying the reference coordinate, the name of the referenced variable, and the magnification factor, a function is implemented to move the coordinate from the reference coordinate by the value of the variable value multiplied by the magnification factor.

4. Evaluation Test

Since *Problem 1* was solved, an evaluation experiment was not conducted. To verify that the time required to write visualization rules can be reduced by solving *Problem 2*, we conducted an evaluation experiment for rule writers. The subjects were two university instructors who teach programming, four university students who have experience as TAs, and four university students who have the same programming ability as the TAs. In the evaluation experiment, after explaining the definitions of the rules to the subjects, we asked them to write existing and new rules to perform the same actions as the example presented, and measured the time required for completion. The task was to visualize the change in "color" and "coordinate" according to the value of the variable, using two questions each for the existing and new ruleset, for a total of four questions. The results of the experiment are shown in Table 1.

Table 1. *Experimental Data*

Problem classification	Colors		Coordinates	
Used ruleset	Existing	New	Existing	New
Existing to New (mm:ss)	40:35	12:00	31:56	9:22
New to Existing (mm:ss)	29:57	25:06	22:32	15:10
Average (mm:ss)	35:16	18:33	27:14	12:16
Ratio [New/Existing](%)	52.6		45.1	

Table 1 shows that the creation time required for the new ruleset was shorter than that for the existing ruleset. Furthermore, multiple people stated in a questionnaire conducted after the experiment that the new rules were easier to write for colors and coordinates. Therefore, it can be inferred that the new ruleset reduced the creation cost of the visualization rules.

5. Summary

We extended the capabilities of TEDViT to allow dynamic mapping of variable values to object properties. Hence, we were able to improve the drawing capability and reduce the cost of creating visualization rules. Conversely, it is necessary to measure the learning effect of mapping variable values and study a new drawing method using the mapping function. However, the description method of the new visualization rules can be improved further.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number JP18K11567 and JP19K12259.

References

AlgorithmVisualizer, https://algorithm-visualizer.org/, (Last viewed on May 22, 2021).

Halim, S., Koh, C., Z., Loh, B. H. V., & Halim, F. (2012) Learning algorithms with unified and interactive Web-based visualization. *Olympiads in Informatics, 6*, 53-68.

Matsumura, K., Daisukey, S., & He., A. (2009). A C language programming education support system based on Software Visualization, *Proceedings of Joint Conferences on Pervasive Computing (JCPC)*, 9-14.

Moreno, A., Myller, N., & Sutinen, E. (2004). Visualizing programs with Jeliot 3, *Proceedings of the working conference on Advanced visual interfaces (AVI)*, 373-376.

Yamashita, K., Fujioka, R., Kogure, S., Noguchi, Y., Konishi, T., & Itoh, Y. (2015). Educational practice of algorithm using learning support system with visualization of program behavior, *Proceedings of ICCE2015*, 632-640.