Poster Proposal for SIGCSE 2016

Proposers:

Ryosuke Ishizue, Dept. of Computer Science and Engineering Waseda University Tokyo, ishizue@ruri.waseda.jp

Kazunori Sakamoto, National Institute of Informatics Tokyo, exkazuu@nii.ac.jp

Hironori Washizaki, Dept. of Computer Science and Engineering Waseda University Tokyo, washizaki@waseda.jp

Yoshiaki Fukazawa, Dept. of Computer Science and Engineering Waseda University Tokyo, fukazawa@waseda.jp

Title:

An Interactive Web Application Visualizing Memory Space For Novice C Programmers

Abstract:

The concept of memory management in C programming language is particularly challenging for novice programmers. Consequently, many researchers have proposed program visualization tools to alleviate these difficulties: for example, SeeC is one of the state-of-the-art tools for visualizing the behavior and execution status of C programs. However, three problems (P1–3) remain in SeeC, as well as in other existing visualization tools. P1 (Usability): SeeC requires many steps to revisualize modified source code. P2 (Capability): SeeC does not fully support dynamic memory allocation. P3 (Installability): novice programmers often find installation of SeeC challenging due to its dependency on Clang. We propose a new visualization tool named PlayVisualizerC (PVC) for novice C programmers, which provides three solutions (S1–3) for P1–3. S1: PVC reduces the steps required for revisualization. S2: complete support for dynamic memory allocation. S3: designed to be installed in the user's web browser. From a small-scale experiment and a questionnaire given to 20 students, we found that a set of four programming tasks were solved 1.8 times faster and 24% more correctly using PVC.

Significance and Relevance of the Topic:

Previous studies have proposed various visualization techniques to aid programmers in understanding program execution status [1–6]. Most existing debuggers and integrated development environments (IDE), such as GDB, Visual Studio, and Eclipse, provide only limited features for visualizing program execution status: for example, these applications display simple text and do not present visualizations of the relationships between variables, pointers, and memory. Consequently, it is often difficult for novice programmers (hereafter referred to as novices) to learn how to utilize such tools (we define a novice as a person learning C language who is currently trying to understand the concept of memory management including variables, pointers, and dynamic memory allocation).

C programming language (C language) is popular and is often one of the first languages learned by novices. However, to master C language, users must learn the basic but difficult-to-master concept of memory management, including pointers and dynamic memory allocation, which is challenging for novices [7–9]. To aid inexperienced users, many previous studies have proposed tools for visualizing program execution status [10–13].

However, there are three problems remain in exsiting tools, P1 (Usability), P2 (Capability) and P3 (Installability). Thus, we propose PlayVisualizerC (PVC), an interpreter of C language with features for visualizing program execution status, which is implemented as a web application. It is effective for novices to learn the concept of memory management.

Content:

This poster will consist of three panels: (1) overview of new visualization tool, (2) comparison with previous works, (3) small-scale experiment.

(1) overview of our new visualization tool

Figure 1 shows a system overview of PVC. Figure 2 presents an example of visualization generated by PVC. Users can write source code in the editor and can control step execution by clicking on the execution control buttons. The output window shows the content of the standard output written by the program (*e.g.* printf), while the canvas shows the program's execution status using tables and figures. PVC uses arrows and boxes to represent pointer and stacks references. The table columns in the box show (1) type, (2) name, (3) value, and (4) address of each variable.

(2) comparison with previous works

As a one of the previous works, SeeC is a state-of-the-art tool for visualizing C programs effectively. Table 1 presents a comparison of features between PVC and SeeC.

SeeC has three main problems (P1–3) that also affect other existing visualization tools, listed as follows. P1 (Usability): users cannot modify source code during SeeC's visualization. Thus, SeeC requires many steps to modify and revisualize a C program. P2 (Capability): SeeC does not fully support dynamic memory allocation, only displaying the size of the allocated

memory in bytes, and omits more detailed memory values. P3 (Installability): it is difficult to install SeeC due to its dependency on Clang, a compiling program.

PVC provides three solutions (S1–3) to the problems outlined above. S1: PVC allows users to easily revisualize a program after modifying its source code by clicking on a button in the editing area. S2: PVC can fully visualize variables, pointers, arrays, and dynamically allocated memory, and the relationships between them. S3: PVC is implemented as a web application requiring only a browser for installation, thereby making it easy for novices to install (http://play-visualizer-c.herokuapp.com).

(3) small scale experiment

We used programming tasks and a questionnaire about our new visualization techniques and tools to evaluate our program. Twenty participants who had learned C language to some degree completed this experiment. we found that a set of four programming tasks were solved 1.8 times faster and 24% more correctly using PVC.

Citations:

[1] Milne, Iain, and Glenn Rowe. "Ogre: Three-dimensional program visualization for novice programmers." Education and Information Technologies 9.3 (2004): 219-237.

[2] Furcy, David. "JHAVEPOP: Visualizing linked-list operations in C++ and Java." Journal of Computing Sciences in Colleges 25.1 (2009): 32-41.

[3] Esteves, Micaela, and A. J. Mendes. "OOP-Anim, a system to support learning of basic object oriented programming concepts." Proceedings of the CompSysTech. 2003.

[4] Sundararaman, Jaishankar, and Godmar Back. "HDPV: interactive, faithful, in-vivo runtime state visualization for C/C++ and Java." Proceedings of the 4th ACM symposium on Software visualization. ACM, 2008.

[5] De Pauw, Wim, et al. "Visualizing the execution of Java programs." Software Visualization. Springer Berlin Heidelberg, 2002. 151-162.

[6] Baerten, Herbert, and Frank Van Reeth. "Using VRML and JAVA to visualize 3D algorithms in computer graphics education." Computer networks and ISDN systems 30.20 (1998): 1833-1839.

[7] Craig, Michelle, and Andrew Petersen. "Student difficulties with pointer concepts in C." Proceedings of the Australasian Computer Science Week Multiconference. ACM,2016.

[8] Milne, Iain, and Glenn Rowe. "Difficulties in learning and teaching programming-views of students and tutors." Education and Information technologies 7.1 (2002): 55-66.

[9] Lahtinen, Essi, Kirsti Ala-Mutka, and Hannu-Matti J¨arvinen. "A study of the difficulties of novice programmers." ACM SIGCSE Bulletin. Vol. 37. No. 3. ACM, 2005.

[10] Egan, Matthew Heinsen, and Chris McDonald. "Program visualization and explanation for novice C programmers." Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148. Australian Computer Society, Inc., 2014.

[11] Koike, Nobuya and Kentaro Go. "A Proposal of Interaction on Visualization Using Address Space Representation in Runtime Program." (in Japanese), IPSJ Interaction. 2012.

[12] Jimenez-Peris, Richardo and Pareja-Flores, Cristobal and Patino-Martinez, Marta and Velazquez-Iturbide, JAngel. "The locker metaphor to teach dynamic memory." ACM SIGCSE Bulletin. Vol. 29. No. 1. ACM, 1997.

[13] Null, Linda, and Karishma Rao. "CAMERA: Introducing memory concepts via visualization." ACM SIGCSE Bulletin. Vol. 37. No. 1. ACM, 2005.

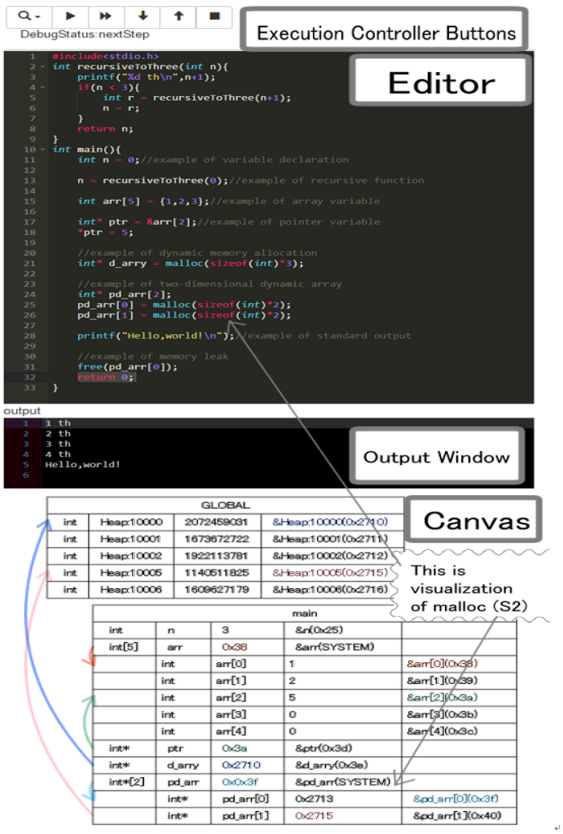

Figure 1: An overview of PVC architecture

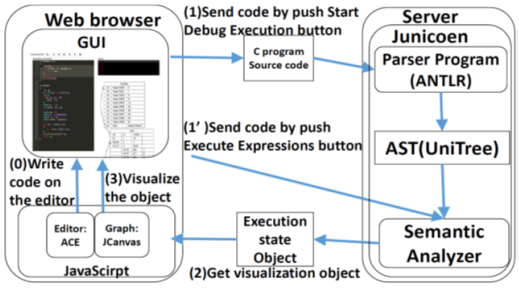

Figure 2: An overview of PVC architecture

Table 1: Comparison of PVC and SeeC functionality

Functions	SeeC	PVC
Values of variables are shown	Y	
Names of variables are shown	Y	Y
Addresses of variables are shown	N	Y
Variables are grouped by the stack	Y	Y
Variables are shown in tabular form	Y	Y
Pointer references are represented by arrows	Y	Y
Arrays are shown in nested tables.	Y	Y
Structs are shown in nested tables	Y	Y
Dynamically allocated memory is visualized	N	Part[1)]
Colorized pointers and referenced variables	N	Y
Movable figures and graphs	N	Y