So, H. J. et al. (Eds.) (2020). Proceedings of the 28th International Conference on Computers in Education.
Asia-Pacific Society for Computers in Education

Learning Support System for Understanding Pointers Based on Pair of Program Visualizations and Classroom Practices

Koichi YAMASHITA[a*], Keisuke SAKATA[b], Satoru KOGURE[b], Yasuhiro NOGUCHI[b], Tatsuhiro KONISHI[b] & Yukihiro ITOH[c]
[a]*Faculty of Business Administration, Tokoha University, Japan*
[b]*Faculty of Informatics, Shizuoka University, Japan*
[c]*Shizuoka University, Japan*
*yamasita@hm.tokoha-u.ac.jp

Abstract: In this paper, we describe a program visualization (PV) system with capability of visualizing logical data structures and concrete memory space in the execution process of a program based on teachers' intent of instruction, and classroom practices for understanding pointers using this system. Thus far, we have developed a PV system called TEDViT which has two distinctive features: capability of customizing PV of target domain world based on teachers' intent of instruction, and capability of simultaneous visualization in target domain world and memory image which have different levels of abstraction. Here, target domain world is a notional world of data processed by the target programs, visualized as logical data structures. However, TEDViT evenly visualizes a memory space as memory image, thus, providing insufficient support for novice learners to understand program behaviors by observing and comparing target domain world and memory image visualizations. Therefore, in this study, we extended the original TEDViT to enable teachers define highlighting expressions in PV of memory image so that they could designate focus points or provide natural language descriptions to learners. We evaluate the effect of learning support that our extended system provided to novice learners by introducing the system in actual classrooms. The evaluation results based on the score improvements between pre and post-tests suggest that our extended TEDViT and classroom practices would have a certain degree of learning effectiveness.

Keywords: Programming education, program visualization system, program visualization design, pointer learning

1. Introduction

To this end, we have developed a program visualization (PV) system called TEDViT (Teacher's Explaining Design Visualization Tool), and have conducted several classroom practices using this system (Kogure et al., 2014). There are two distinctive features of TEDViT: first, it makes it possible for teachers to customize PVs based on their intent of instruction, and second, it makes it possible for learners to observe and compare two PVs with different abstraction levels of target data structures. However, from our own experience of several classroom practices, we consider that the visualizations provided by TEDViT are not sufficient to help novice learners understand pointers. For instance, for target domain world, teachers can customize PVs and designate focus points by highlighting the changes made in the domain world by the target program by providing comments with balloon, etc. Meanwhile, the visualization of memory image provides almost the same level of information as a debugger since TEDViT can only flatly visualize the contents of memory space, and cannot express teachers' intent of instruction by highlighting the changes. If the learners are software engineers who have an intuitive grasp of the so-called notional machine (Sorva, 2013), they may be able to perceive the change in the memory image and detect points to focus based on their own experience. However, novice learners with insufficient background knowledge might not perceive the changes.

Therefore, in this study, we extended TEDViT to make it possible for teachers to define highlighting expressions for target domain world as well as memory image. The PV customizations for

target domain world are achieved by defining drawing rules in TEDViT. The additions include a rule system to define highlighting expressions for memory image and a function to visualize memory image. We conducted two classroom practices for understanding pointers using our extended TEDViT, incorporating it into actual programming courses for novices. The evaluation results based on pre- and post-tests suggest that the extended TEDViT would have a certain degree of effectiveness in novice learners' understanding of pointers.

2. Previous Works

TEDViT allows teachers to define the policy for drawing a status of the target domain world based on their intent of instruction. Teachers can create or edit a configuration file independently from the target program file. TEDViT interprets such a visualization policy by scanning the configuration file and then visualizes the target domain world accordingly. The learners can then observe the program behavior in the target world visualized in accordance with the teacher's intent of instruction. At the moment, TEDViT supports visualizations of C programs only.

Figure 1 shows a learning environment generated by TEDViT. The environment comprises three fields: the data structures processed by the program in (A) are visualized in (B) and (C). TEDViT reproduces a series of memory images of variables in (B) for each step of the program's execution, and a series of statuses of the target domain world in (C) that visualizes logical data structures. When a learner clicks the "Next" or "Prev" button, the highlight in (A) moves to the next or previous statement in the program code, the memory image in (B) is updated according to the values of the variables after executing the highlighted statement, and the corresponding status of the target domain world is visualized in (C). TEDViT simulates statement execution step by step so that the learner can understand the program's behavior by observing the changes in the target domain world in (C). The learner can also understand the concrete memory image in each execution step and the concrete expression of the data structures by observing and comparing the target domain world in (C) with the memory image in (B).

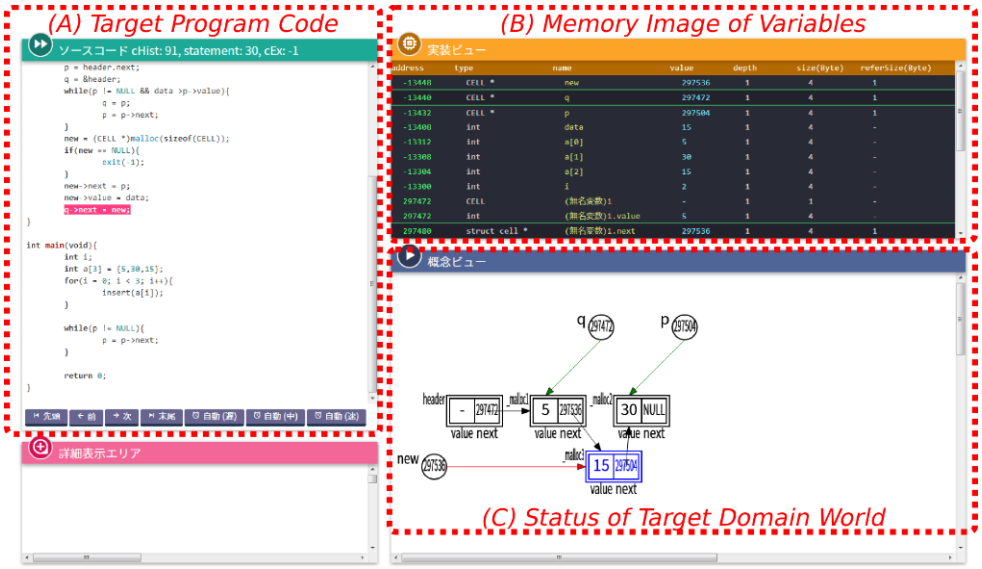

Figure 1. Learning Environment Generated by TEDViT.

However, compared with the visualizations of the target domain world in (C), which include highlighting expressions and descriptive comments defined by teachers, the flat visualizations of the memory image in (B), which lack customizability, make it difficult to perceive changes in values and understand what is happening in memory space. For this reason, we have almost never conducted classroom practices introducing TEDViT with learning scenario including observations of PVs in (B). Although an evaluation result suggested that observations of PVs in (B) would have some effectiveness in a practiced class targeting software engineers (Yamashita et al., 2017), we consider that it was because the subjects had some expectations about the changes in memory image from their experiences

and had observed PVs in (B) based on these expectations. However, for novice learners, TEDViT provides inadequate visualizations in (B) to support their understanding of program behaviors.

3. Extending Visualizations of Memory Image Field

We extended the drawing function and rule system of TEDViT to visualize drawing objects in memory image field based on teachers' intent of instruction. The insufficient support for novices in the original TEDViT had arisen from difficulties in perceiving changes in values and intuitively grasping the focus points in PVs in the memory image. Therefore, we included additional drawing objects in the extended TEDViT to visualize highlighting expressions. In addition, we analyzed several PVs from various teaching materials and textbooks, and decided to introduce the following extensions to TEDViT.

Ext1. Function to highlight the value of a specific pointer variable and the area pointed to by the pointer in memory image field.
Ext2. Function to set character color, background color, font size, and framed box of specific value in memory image field.
Ext3. Function to draw a framed box around the area corresponding to a specific variable in memory image field.
Ext4. Function to draw a text box of descriptive comment in memory image field.

Figure 2 shows a memory image visualization generated using the abovementioned extensions. Ext1 is a function to draw a framed box around the value of a specified pointer variable, an arrow toward the area pointed to by the pointer, and a framed box around that area. A single drawing rule is required to define Ext1, specifying the target pointer variable and the number of area blocks pointed to by that target pointer. Ext2 is a function to highlight a specified single value in memory image field. The single value can be that of a variable, or the address of a variable, variable name, and type of variable. Ext3 is a function to set a framed box around the entire area visualizing information of a specified variable. Ext4 is a function to display a message in memory image field, specifying drawing position as the position relative to the specified single value.

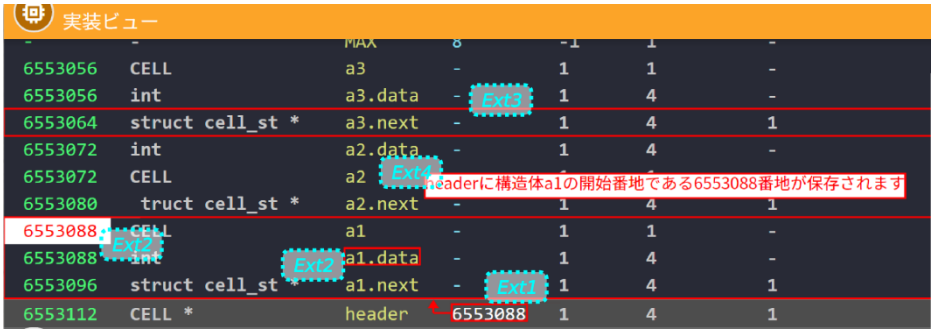

Figure 2. Example of Memory Image Visualization Generated Using Extensions.

The memory image field visualizes memory area corresponding to the processing data of target program in tabular form. The drawing unit is a variable, an array element, or a structure member, each of which is visualized in a single column. The displayed item in each row corresponds to a prescribed attribute, such as the address of the data, reference name in the target program, and its value. It is easy to specify a particular column and row, hence how to draw positions or highlight targets can be specified in drawing rules without any ambiguity. As mentioned earlier, the drawing rule in the extended TEDViT consists of a condition and an object part. While the former can be described in the same way as was done in the original TEDViT, the latter can be specified using the abovementioned extensions. We consider that drawing rules in the extended rule system can be defined without any confusion with some experience in rule definitions.

4. Evaluation

To evaluate the effectiveness of learning scenario for novice learners including observations of PVs in memory images and the learning support of the environment generated by the extended TEDViT, we conducted classroom practices introducing the system in two actual programming courses for different students of the same university. We evaluated system effectiveness based on pre- and post-test scores. Table 1 provides a brief summary of the classroom practices.

Table 1. *Summary of Classroom Practices*

Class	#1	#2
Subject (N)	90	40
Subjects' major	Computer Science	Mathematical and System Engineering
Subjects' grade	Sophomore	Freshman
Course	Algorithms and Data Structures I	Fundamental programming

Class #1 was incorporated into the course "Algorithms and Data Structures I" for sophomores majoring in Computer Science. The subjects had one-year experience in Java programming but had no experience in C. The aim of the class was to understand pointers in connection with a memory model as a C specific feature not found in Java, and had the following three goals.

G1-1. To understand address and pointer operators (i.e., reference and dereference operators).
G1-2. To understand subscript operators and relationships between subscript and pointer operators.
G1-3. To understand the implementation and working of pointers to structure variable.

In the Class, the teacher who regularly taught the course lectured on fundamental syntax of C language and pointers for 90 minutes, and then conducted a pre-test for 10 minutes. The pre-test consisted of three questions corresponding to the three goals, presenting the subjects with three programs similar to the ones discussed by the teacher in the lecture and asking about related concepts.

Q1-1. Asking the meaning of the values obtained by address and pointer operators at certain points in the execution process.
Q1-2. Asking about the reference target when an array was accessed without subscript operators.
Q1-3. Asking the meaning of the value obtained by address operation with a structure variable.

After the pre-test, the teacher allowed students to use our extended TEDViT to observe program behaviors without giving them any feedback on the test. The target was three sample programs which the teacher discussed in the lecture session. The teacher explained how to launch and operate the learning environment generated by the extended TEDViT and gave the students 10 minutes each to observe the three target program behaviors. During this observation session, the teacher did not provide any explanation about the target program behaviors but directed students to observe and compare the three fields of TEDViT according to visualizations on the learning environment. The drawing rules of the extended TEDViT included visualizations in the target domain world and highlighting expressions in memory image, and was defined by the teacher beforehand based on his own intent of instruction.

Following the observation session, which lasted 30 minutes, the teacher conducted a post-test for 10 minutes. The content of the post-test was almost same as the pre-test, except the assigned values of some variables and constant values in some arithmetic expressions. Table 2 shows the average rate of correct answers for pre- and post-tests in the class. The average rate for each question significantly improved after introducing the extended TEDViT, suggesting that this classroom practice had a certain degree of effectiveness.

Table 2. *Average Rates of Correct Answers for Pre- and Post-Tests in Class #1*

Question	Pre-test	Post-test
Q1-1	.562	.844
Q1-2	.236	.692
Q1-3	.382	.788

Class #2 was incorporated into the course "Fundamental Programming" for freshmen majoring in Mathematical and System Engineering, as the second class in two consecutive lessons on pointers.

The subjects had less than a year of programming experience. The aim of the class resembled that of Class #1, and included the following goals.

G2-1. To understand the fundamental concepts of pointers such as address and pointer operators, pointer increment and decrement, and so on.
G2-2. To understand behaviors of "pseudo" call by reference by using pointers in function call.
G2-3. To understand subscript operators and relationships between subscript and pointer operators.

The overall flow of Class #2 was almost same as that of Class #1. A teacher who regularly taught the course lectured the students for 90 minutes on pointers, coding exercises for three example programs corresponding to the three goals, and operation verifications of sample programs. After that, the teacher conducted a pre-test for 10 minutes which consisted of three questions corresponding to the three goals.

Q2-1. Asking the values obtained by expressions including pointer and increment operators at certain points in the execution process.
Q2-2. Asking the return values of function call with and without pointer variables.
Q2-3. Asking the values of array elements assigned using several assignment expressions including subscript, pointer, and increment operators.

After the pre-test, the teacher allowed students to use our extended TEDViT to observe program behaviors without giving them any feedback on the test. The target was the same three example programs discussed in the coding exercises. The teacher explained how to launch and operate the TEDViT learning environment and gave the students 15 minutes each to observe the three target program behaviors. As in Class #1, here too the teacher did not provide any explanation about the program behaviors and defined the drawing rules beforehand based on his own intent of instruction.

Following the observation session, which lasted 45 minutes, the teacher conducted a post-test for 10 minutes. The content of the post-test was almost same as the pre-test, except the assigned values of some variables and constant values in the provided programs were changed from the pre-test. Table 3 shows the average rate of correct answers for pre- and post-tests. The average rate for each question significantly improved after introducing the extended TEDViT, suggesting that this classroom practice had a certain degree of effectiveness. The reason behind the small improvement in average rate for Q2-3 compared to the other questions could be the insufficient understanding of arrays. The subjects in Class #2 did not have much programming experience and had learned arrays just before the lessons on pointers. The similar reason also holds for the relatively low rate of correct answers for Q2-3 in both pre- and post-tests.

Table 3. *Average Rates of Correct Answers for Pre- and Post-Tests in Class #2*

Question	Pre-test	Post-test
Q2-1	.256	.581
Q2-2	.535	.814
Q2-3	.165	.279

In addition, to evaluate the cost of defining the highlighting expressions in memory image field, we measured the time required to define the drawing rules for two programs p_1 and p_2 consisting of about 30 statements. The number of subjects of this pilot experiment were eight: two teachers who have experience of teaching programming, two students who have experience as programming teaching assistants, and four students who have the same level of programming experience as teaching assistants. First, we explained the drawing rules and rule system of the extended TEDViT to the subjects, and discussed an example program consisting of about 20 statements and drawing rules consisting of highlighting expressions in memory image field for the example program. Then, the subjects observed another PV without any drawing rules and were asked to define the drawing rules that reproduce the provided PV. The time taken by each subject to define the drawing rules was measured. To reduce the order effect, we divided the subjects into two groups, each of which included one teacher, one teaching assistant, and two students, and swapped the order of target program for rule definitions. The number of defined rules by each subject was about 20 to 30, although there were some differences according to the subject. Table 4 details the result of this pilot experiment. Every subject took less time to define rules for the second target program than the first one, suggesting that experience could improve the

productivity of rule definitions. The average time to define highlighting expressions in memory image field for the second target program is 26.05 minutes.

Table 4. *Time Taken to Define PVs by each Subject*

Subject #	1st target	Time	2nd target	Time
1	p_1	40m 25s	p_2	37m 01s
2	p_1	25m 30s	p_2	21m 22s
3	p_1	50m 03s	p_2	30m 59s
4	p_1	34m 19s	p_2	26m 55s
5	p_2	23m 06s	p_1	21m 00s
6	p_2	30m 24s	p_1	20m 38s
7	p_2	31m 15s	p_1	24m 40s
8	p_2	45m 12s	p_1	25m 48s

5. Conclusion

In this paper, we describe a PV system with capability of defining highlighting expressions for the target domain world as well as memory image, and two classroom practices introducing the system in actual classes. We extended the rule system of TEDViT and added the functions for rule interpretation and drawing memory image field. This extension allowed teachers to define highlighting expressions using the extension, and it was expected that learners could perceive changes in values of memory spaces and detect points to focus. To evaluate the effectiveness of the learning support provided by the extended TEDViT and the learning scenario, we conducted classroom practices introducing it in actual classes. As the learning target, we focused on pointers which many novice learners struggle with. The evaluation results based on pre- and post-test scores clarified that the learners cultivated a better understanding of pointers, hence, suggesting that our classroom practices introducing the extended TEDViT would have a certain degree of effectiveness. Furthermore, to evaluate the cost of classroom preparation using our extended TEDViT, we measured the time taken by eight subjects including actual teachers to define drawing rules of some highlighting expressions in memory image field. This evaluation result suggests that teachers can define PVs based on their intent of instruction in actual classes using the extended rule system which includes visualizations of highlighting expressions in memory image field. Therefore, we conclude that the extended TEDViT in this study makes it possible to define PVs based on teachers' intent of instruction, and that it would have a certain degree of effectiveness in helping novice learners understand pointers.

Acknowledgements

This study was supported by JSPS KAKENHI Grant Numbers JP18K11566, JP18K11567, and JP19K12259.

References

Kogure, S., Fujioka, R., Noguchi, Y., Yamashita, K., Konishi, T., & Itoh, Y. (2014). Code reading environment according to visualizing both variable's memory image and target world's status. Proceeding of the 22nd International Conference on Computers in Education (ICCE2014), 343-348.

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education (TOCE), 13*(2), 1-31.

Yamashita, K., Fujioka, R., Kogure, S., Noguchi, Y., Konishi, T., & Itoh, Y. (2017). Classroom practice for understanding pointers using learning support system for visualizing memory image and target domain world. *Research and Practice in Technology Enhanced Learning (RPTEL), 12*(17), 1-16. doi:10.1186/s41039-017-0058-4