Important features in program visualization

Erkki Kaila, Teemu Rajala, Riku Haavisto, Johannes Holvitie, Mikko-Jussi Laakso, Tapio Salakoski

University of Turku and Turku Centre for Computer Science (TUCS),

Turku, Finland, ertaka@utu.fi

University of Turku and Doctoral Programme for Multidisciplinary Research on Learning Environments (OPMON),

Turku, Finland, temira@utu.fi

University of Turku, Turku, Finland, rahaav@utu.fi

University of Turku, Turku, Finland, jjholv@utu.fi

University of Turku, Turku, Finland, milaak@utu.fi

University of Turku, Turku, Finland, sala@utu.fi

Abstract

Program visualization is a method which can be used to significantly enhance the learning results in introductory programming courses. We have previously conducted a qualitative study among university students about their opinions on usage of a program visualization tool. This time we repeated the study among two instances of high school programming courses. Despite the age and maturity of the students, there were some other differences as well, including the earlier experience of using the tool and the list of features included in the survey. The results show that the students find the tool both motivating and easy to use. Even more importantly, the students felt that using the tool helped them to learn the basic programming concepts. The features found most useful were the flexible controlling of visualizations and the verbal explanations about the executed code lines. Since the results in the university level were quite similar, we need to keep these facts in mind when developing such tools.

# Introduction

Program visualization is a method, where the execution of example programs is presented with graphical and/or textual components. The goal of the visualization tools is to help novice students understand the behavior of programs and algorithms. However, as stated for example in Hundhausen et al. [1], visualizations should be accompanied with tasks of higher level of engagement to achieve the learning results. Instead of passive viewing, the students should participate actively in the visualization process [2]. To accomplish this, we have created a program visualization tool called ViLLE, which allows teachers to attach multiple choice questions or graphical array questions to fixed points of visualized programs.

We have previously shown in various studies that program visualization tool accompanied with engaging exercises can be a significant aid in learning the basic programming. However, we are also interested in which features of ViLLE the students find most useful. The learning outcome is of course the most important issue when developing such tools. However, collecting and analyzing students’ opinions about the usefulness of different features means that we can further develop the tools to improve the learning experience.

We conducted a series of experiments with total of 24 high school students during the academic year of 2007-2008 in Finland. The first experiment [3] had a typical two hour pre-test – treatment – post-test design. The difference between the two groups was the amount of usage of the tool before the experiment, and the results showed that it is very important to familiarize students properly with the tool before the experiment. In the second study [4] we analyzed the long-term effects of a program visualization tool. The results indicate that using the tool enhances the learning of programming concepts: the treatment group used the tool throughout the course and outperformed the control group, who utilized the tool only for 2 hours during the course.

In this paper, we present the results from a quantitative analysis of the survey data about students’ opinions on the usability and usefulness of features in the tool. Both groups answered to a questionnaire after they had used the tool during a two-hour tutorial session. The questions were divided into three categories: the general impressions of the tool, the usefulness of the tool in relation to some basic programming concepts, and the usefulness of different features of the tool. We have previously conducted similar study at university level. In this paper the new results are analyzed and compared to previous results in all three categories.

The paper has the following structure: in section 1.1 the visualization tool used in our experiments is shortly described. Some relevant research related to our work is presented in section 2. In section 3 our previous studies related to this paper are described. The research setup and results are presented in sections 4 and 5. The results of this study are discussed and compared to our earlier studies in section 6, and finally, conclusions are presented in section 7.

## ViLLE

ViLLE (see Figure 1) is a program visualization tool developed at University of Turku, Finland. It visualizes the execution of programs by displaying e.g. variable values, program line explanations and subprograms in their own areas. The program can be executed one step at a time, both forwards and backwards. The teacher can attach questions (multiple choice questions and graphical array questions) to fixed points of visualized programs. ViLLE supports a variety of programming languages, and comes accompanied with editors for defining new syntaxes, questions and exercises. More information can be found in [5].



Figure 1: ViLLE's visualization view

# Related work

Many software visualization systems have been developed over the past few decades. Software visualization is often divided into two types of visualizations: program visualization and algorithm visualization. The development in the field of software visualization is mostly focused on algorithm visualization which focuses on visualizing data structures and algorithms. Notable algorithm visualization tools include for example JHAVE [6], BALSA-II [7], ZEUS [8], XTANGO [9] and TRAKLA2 [10]. In program visualization the execution and the state of a program are presented with graphical and textual information. Currently some of the best known program visualization tools are Jeliot3 [11] and ViLLE.

Bassil and Keller [12] report a survey on software visualization tools. The participants of the survey were asked to rate the usefulness and importance of software visualization in four different aspects: functional, practical, cognitive and code analysis. According to the participants the most essential functional aspects were searching and browsing, use of colors, and easy access from the symbol list to the corresponding source code. Also, hierarchical representations as well as navigation across hierarchies were features strongly desired by the participants.

Kannusmäki et al. [13] report a study where students utilized Jeliot 3 programming visualization tool in a distance programming course. Their goal was to find out how students use the tool and what kind of features students would like to have in such tool. Students liked how the tool helped them understand if-statements, loops and objects, and how it showed step-by-step what happens inside a program. Some negative aspects according to the students were that the error messages were unclear and that the program code editor lacked some functionality.

Hansen et al. [14] report four different empirical studies in where students used hypermedia algorithm visualization system HalVis to learn algorithms. The results show that students who used HalVis learned significantly better than students who used traditional teaching materials or typical algorithm animations. They also discuss the key features of the system and show some excerpts from the survey data gathered in the studies.

# Previous research

The qualitative data in this paper is gathered from two consecutive high school programming courses during 2007-2008. We have earlier published two papers based on the quantitative data gathered from this course. In the first one we reported a study where two groups of students utilized ViLLE during a two-hour programming session. The treatment group was familiarized with the tool’s user interface before the session while the control group didn’t have any earlier knowledge of ViLLE. The treatment group learned significantly better during the session. Based on the result, we concluded that students should be familiarized with a visualization tool before such experiments in order to avoid false conclusions [2]. In the second paper we studied the course-long effects of program visualization [4]. The students in the first instance of the course utilized ViLLE only in the two-hour session described earlier while the students in the last instance used ViLLE throughout the course. The students who used ViLLE throughout the course got significantly better results from the course’s final exam.

In one of our earlier studies we asked university level students’ opinions about ViLLE [15]. Total of 114 students answered to a questionnaire which included questions about ViLLE’s usefulness in learning basic programming concepts, usefulness of its features, and general questions about the tool.

# Research setup

The research was conducted in two instances of a high school introductory programming course during the academic year of 2007 / 2008 in Finland. There were 24 students in total in the course. During the course all students participated in a two-hour tutorial session. The session started with post-test followed by a practice session including ViLLE exercises. At the end of the session a post-test was organized to find out whether the students had learned anything during the session. We have previously reported the quantitative results in Laakso et al. [3].

The post-test was accompanied with a survey which consisted of 16 questions, divided into three categories. All questions were answered in the scale of one to seven, where 1 meant that the student strongly disagreed with the claim, and 7 that the student strongly agreed with it. Answering 4 was considered neutral, i.e. that the student had no opinion whatsoever on the subject.

There were two instances of the course (N1=17, N2=7); however, since there were no statistically significant differences in the averages or standard deviations between the groups, the results were combined for this study (N = 24).

We have previously presented the results from similar study conducted at university level [15]. There are however some differences between the studies:

* This study was conducted among high school students who presumably are less mature and have less previous knowledge on the subject.
* There was a difference in the list of features included in the survey.
* The university students had used ViLLE throughout the course before answering to the survey. For the high school students this was the first time they used the tool properly (although some of them were introduced to the tool’s user interface before the session).

# Results

In this section we present the results of the study. In the next section the results are analyzed and compared to previous results. The first category consisted of five claims concerning general issues of the tool. The results for the first category are presented in Table 1.

Table 1: Results for the general questions.

|  |  |
| --- | --- |
| Claim | Average (Standard deviation) |
| The tool was easy to use | 6.17 (1.09) |
| Tool is suitable for introductory courses (such as the current course) | 5.83 (1.34) |
| Using the tool was fun and motivating | 4.71 (1.73) |
| Answering the questions was motivating | 4.42 (1.59) |
| Using the tool helped me to learn and understand programming related concepts | 5.13 (1.54) |

As seen in table, the claim about the easiness of use got the highest average. All in all, the students seem to think that using the tool is both fun and motivating. Notably, the average for answering the questions is the lowest; this may be due to the fact that answering the questions is the most time- and work-consuming part in using the tool. Ironically, it however is considered the feature that has the best effect on learning.

The second category consisted of seven different concepts of programming. The students were asked to evaluate whether ViLLE helped them to understand the concepts well. The results for the second category of questions are presented in Table 2.

Table 2: Usefulness of ViLLE in learning programming concepts.

|  |  |
| --- | --- |
| Concept | Average (Standard deviation) |
| Variable assignments | 5.17 (1.93) |
| Selection statements | 5.42 (1.69) |
| Pre-conditional loops | 5.54 (1.44) |
| Boolean statements | 4.83 (1.71) |
| Function definitions | 3.96 (1.55) |
| Function calls | 4.13 (1.57) |
| Function parameters | 3.63 (1.41) |

As seen in table, the students found ViLLE quite useful (average > 5) in learning the basic concepts (assignment, selection, repetition). However, the concepts related to functions got lower averages.

The final category consisted of four claims concerning ViLLE’s features. The students were asked to evaluate whether they found the given features in the tool useful. The list was selected by the authors to reflect the features we thought might be most useful. The textual explanation is an automatically constructed description of the line currently executed. Typical example would be something like ‘Assign the value of expression a + b (2 + 9 = 11) to variable c’. The variable state and value area displays all variables with their current values; for object type variables a reference is displayed as a variable value, and the actual objects (namely strings and lists) are displayed in their own area. The possibility to move backwards is a feature often missing in visualization tools. The feature gives students the possibility to repeat previous steps easily and hence deepen the knowledge of e.g. repetition and selection. The questions asked during the execution is a very important feature, as they engage the students into visualizations. As stated before, visualization tools can benefit the learning only when used in higher levels of engagement. The results for the third category of questions can be found in Table 3.

Table 3: Usefulness of ViLLE's features.

|  |  |
| --- | --- |
| Feature | Average (Standard deviation) |
| The textual explanation | 5.67 (1.55) |
| The variable state and value area | 5.25 (1.54) |
| Possibility to move backwards in execution | 6.13 (1.23) |
| The questions asked during the execution | 4.71 (1.68) |

The ability to move backwards in the execution got the highest average. Notably, such feature is often missing in similar tools. The second highest average was for the textual explanations. Variable states and questions were also considered to be important features (average > 4.5).

# Discussion

We have previously presented the results from similar study conducted at university level [15]. There are however some differences between the studies:

* This study was conducted among high school students who presumably are less mature and have less previous knowledge on the subject.
* There was difference in the list of features included in the survey.
* The university students had used ViLLE throughout the course before answering to the survey. For the high school students this was the first time they used the tool properly (although some of them were introduced to the tool’s interface before the session).

The results for the general claims were similar. The tool was considered easy to use (university average 5.49, high school average 6.17), found suitable for teaching programming (UA 5.64, HSA 5.83), and helpful for learning basic programming concepts (UA 5.41, HSA 5.13).

The results for usefulness of ViLLE in understanding basic programming concepts at university level are presented in table 4.

Table 4: Usefulness of ViLLE in understanding programming concepts (university level)

|  |
| --- |
| How useful did you find ViLLE in understanding the following concepts? (scale 1 - 7) |
| Variables and assignments | 5.41 |
| Conditional statements | 5.52 |
| Loops | 5.61 |
| Boolean statements | 5.38 |
| Subprogram definitions | 5.38 |
| Subprogram calls | 5.34 |
| Subprogram parameters | 5.24 |
| Arrays | 4.73 |

There is a notable difference in function related concepts: the students at university level found that ViLLE helped them more in such issues. This may be because they used the tool throughout the course and hence did more exercises about functions. Based on the results from the both surveys we can conclude that ViLLE can be used as a motivator in programming education and it can be effectively utilized both at university and high school level courses.

Table 5 includes the averages and standard deviations (in parentheses) of students’ evaluations of ViLLE’s features at university level.

Table 5: Usefulness of ViLLE's individual features (university level)

|  |
| --- |
| How useful did you find the following features in ViLLE (scale 1 – 7)? |
| Visualization of programs in different languages | 4.93 (1.46) |
| Visualization of subprograms with call stack | 5.35 (1.24) |
| Visualization of variable states | 5.90 (1.17) |
| Explanation of program code line | 5.40 (1.49) |
| Questions about program execution | 5.50 (1.24) |
| Automatic assessment of exercises | 5.80 (1.28) |

The list of features wasn’t similar in the surveys: there are however three features that were included in both surveys. The textual explanation and variable states were considered highly important by both groups. However, university students found the questions about program execution to be more important (UA 5.50, HSA 4.71). There may be several reasons for this: answering questions takes a lot of time and work. Although this presumably leads to better learning, it may not be considered fun all the time. It is possible that university level students better understood the importance of questions as engaging feature. It should be also noted that the participant count was much higher in the university level study, and hence the results from that study should be more reliable.

The feature that high school students found most useful was the ability to move backwards in the execution. This indicates, that the students do repeat the more complex parts of the execution to determine how the programs actually work. The ability to trace the execution backwards also underlines the consecutive nature of program behavior: the current state depends on all previous actions in the program. It is easy to step one step too far in the execution and in traditional debuggers and in most other visualization systems the only possible way to go back to the previous step is to start from the beginning. This possibility to move backwards in the execution should be taken account when such systems are designed and implemented.

Program line explanation and variable states got high averages in both studies. Presenting an explanation about currently executed program code line in natural language seems to be quite important feature for students; notably, an explanation is usually missing from traditional debugging environments. Though the information presented in explanation area is usually presented in other formats (such as variable states and call stack), it seems likely that the students anyway prefer the possibility to read about program actions in their own language. This may be especially important at early stages of learning when a situation occurs where a student can’t grasp the required information from the traditional visualization components. As visualizing variable states is usually considered as one of the most important features in any program visualization tool, it is no surprise that the average given to it was quite high.

The feature list in the high school survey was not comprehensive: instead of asking about all of ViLLE’s features we picked only four. The main reason for this was to keep the survey short enough so that the students had enough time to consider their answers (the survey was held during the post-test, which itself was quite time-consuming). However, the learning results were good, as seen by our quantitative results [3]; hence, the appreciation of given features has a lot of significance.

When considering the results from the earlier quantitative and qualitative studies and both surveys combined, we can conclude that students find ViLLE both motivating and useful which can also be seen in their learning results.

# Conclusions

Based on the results of the surveys we can point out features that should be promoted in visualization tools:

* The tool should activate and engage students for example with questions during the visualization.
* The tool should have flexible controls. Based on our research especially the possibility to move backwards in the execution is feature much appreciated.
* Variable states should be presented clearly.
* Textual explanation of the events in programs in natural language. This should be well synchronized with the program state info.

The students seem to value the possibility to control the execution flexibly, including the possibility to move backwards. Moreover, a bimodal view to visualized data is considered important: in addition to ‘traditional’ elements, such as variable states, a natural language description is appreciated. The average for the questions asked during the execution was lower than for the other features (and lower than at university level) among high school students. It’s possible that students in high school considered the questions quite similar, which means that answering several of them in row may seem like a boring task. The newer versions of ViLLE have addressed the issue by creating more heterogeneous questions, and – even more importantly – new types of exercises including e.g. coding, sorting and program simulation exercises. To develop the tool in direction where students find it more motivating and fun to use is very important. This kind of development is likely to reflect into learning results as well. The next version of ViLLE is a collaborative educational tool which supports peer collaboration with various types of engaging exercises.

# References

1. Hundhausen, C.D., Douglas, S.A. & Stasko, J.D. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing, 13*, 259-290.
2. Laakso, M.-J. (2010). Promoting Programming Learning. Engagement, Automatic Assessment with Immediate Feedback in Visualizations. TUCS Dissertations no 131.
3. Laakso, M.-J., Rajala, T., Kaila, E. and Salakoski, T. (2008). *The Impact of Prior Experience in Using a Visualization Tool on Learning to Program.*Proceedings of CELDA 2008, Freiburg, Germany, 129-136.
4. Kaila, E., Rajala, T., Laakso, M.-J. & Salakoski, T. (2010). *Long-term Effects of Program Visualization.* In 12th Australasian Computing Education Conference (ACE 2010), January 18- 22, 2010, Brisbane, Australia.
5. Rajala, T., Laakso, M.-J., Kaila, E. & Salakoski, T. (2007). *VILLE - A Language-Independent Program Visualization Tool.* In proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland, November 15-18, 2007. Conferences in Research and Practice in Information Technology, Vol. 88, Australian Computer Society. Raymond Lister and Simon, Eds.
6. Grissom, S., McNally, M. & Naps, T. (2003). Algorithm visualization in CS education: Comparing levels of student engagement. *Proceedings of the ACM Symposium on Software Visualization*, San Diego, California, 87–94.
7. Brown, M.H. (1988). Exploring algorithms using Balsa II. *IEEE Computer, 21*(5)*,* 14-36.
8. Brown, M.H. (1991). Zeus: A system for algorithm animation and multi-view editing. *Proceedings of IEEE Workshop on Visual Languages,* 4-9. New York: IEEE Computer Society Press.
9. Stasko, J. (1992). Animating algorithms with XTANGO. *ACM SIGACT News, 23*(2), 67-71.
10. Malmi, L., Karavirta, V., Korhonen, A., Nikander, J., Seppälä, O. & Silvasti, P. (2004). Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education, 3*(2), 267-288.
11. Moreno, A., Myller, N., Sutinen, E. & Ben-Ari, M. (2004). Visualizing programs with Jeliot 3. *Proceedings of the Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, 373-376.
12. Bassil, S., Keller, R.K. (2001). Software visualization tools: survey and analysis. Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on , vol., no., pp.7-17.
13. Kannusmäki, O., Moreno, A., Myller, N. & Sutinen, E. (2004). What a novice wants: Students using program visualization in distance programming course. *Proceedings of the Third Program Visualization Workshop (PVW'04)*, Warwick, UK.
14. Hansen, S., Narayanan, N.H., Hegarty, M. 2002. Designing Educationally Effective Algorithm Visualizations. Journal of Visual Languages & Computing, Volume 13, Issue 3, 291-317.
15. Kaila, E., Rajala, T., Laakso, M.-J. & Salakoski, T. 2009. *Effects, Experiences and Feedback from Studies of a Program Visualization Tool.*Informatics in Education, 8, 1, 17-34.