Teemu Rajala | Mikko-Jussi Laakso | Erkki Kaila | Tapio Salakoski

VILLE – Multilanguage Tool for Teaching Novice Programming

Turku Centre *for* Computer Science

TUCS Technical Report
No 827, June 2007

VILLE – Multilanguage Tool for Teaching Novice Programming

Teemu Rajala
University of Turku, Department of Information Technology
Mikko-Jussi Laakso
University of Turku, Department of Information Technology
Erkki Kaila
University of Turku, Department of Information Technology
Tapio Salakoski
University of Turku, Department of Information Technology

TUCS Technical Report
No 827, Month 2007

Abstract

Visualization tools have proven to be useful for enhancing novice programming learning. However, those tools are typically tide to a certain programming language and focus to low level aspect of programming such as changing value of variables during code execution. In this paper, we present a new program visualization tool which provides a language independency view of learning programming and supporting learning process by integrating role and call stack information. Moreover, two different languages can be viewed in parallel and role information can be inserted to support learning process in more abstract level. In addition, VILLE can emphasize that the syntax and definitions of basic programming concepts are very similar in every imperative programming language.

Keywords: Program Visualization, Novice Programming, Language Independence, Multilanguage, Pseudo language

TUCS Laboratory
Learning and Reasoning Laboratory

1. Introduction

Teaching programming have been challenging and interesting field in computer science for many decades. Many computer based systems have been developed for novice programmers to aid their learning process at early steps of learning to program. Those systems use different kinds of visualization and animation techniques to give more concrete model and to show what actually takes place during program code execution. Usually, the goal of such systems is to enhance students understanding of program execution [10]. Moreover, Lister et al [15] and McCracken et al [17] have showed that novice programmers have difficulties to read and understand a given program code and create their own programs.

In general, the most of visualization and animation based system are heavily program language dependant and those systems can only visualize or animate the execution of program code in just one programming language. However, the syntax and definitions of basic programming concepts are very similar in all imperative programming language. Those basic concepts include the control structures (consecution, selection and loops), statements, expressions, tables, method and their definition.

From student's point of view it is not that important to learn how loops are defined and executed in Java –programming language. Far more important aspect is to conquer what are the basic principles behind loop-structure regardless of selected programming language.

More over, Grandell et al [7] have argued that for novice programmer courses the programming language's syntax should be as simple as possible and this way more focus and time can be placed on studying more important and relevant aspect of basic programming. Defining a pseudo language can be one really good alternative and it should be used for teaching programming at early steps. While using pseudo language the algorithm or program code can be represented in higher abstraction level as Boada et al [4] and Stern et al [24] have reported. Pseudo language is often perceived as language without possibility to execute or interpret as Garner [6] have noted. More over, another higher abstraction tool for teaching is the concept of roles of variables which Sajaniemi [21] have defined a taxonomy which is based on variable's behaviour during the execution of a program. With this concept a student can interpret program code in more abstract way regardless of programming language or even programming paradigm. Sajaniemi and Kuittinen [22] have noticed that using role information in basic programming courses aided the learning process of the students, because the role concepts can enhance student's understanding about the program.

These are examples of aspects which should be taken into consideration during the definition and design of visualization or animation based application targeted for basic programming courses to aid the learning process of individual student.

VILLE is a visual tool, which can be used both in lecture and for self-learning. The tool supports Java, C++ and pseudo –programming languages. The pseudo programming language can be self defined and its' execution can be traced as well as the other languages. Programming examples can be presented in parallel view side by side in two different languages and by doing that the language independency view of basic programming concepts can be emphasized. VILLE can also trace step by step (or row by row as we define it) the execution of program code and its' effects to values of variable and program outputs. More over, every executed code line is connected and explained with textual description of its' meaning and actions. Role information is also integrated to make visualization more effective and easy to interpret. In addition, VILLE's predefined or self-defined examples can be published in the web for 24/7 availability for students a possibility to engage the learning session at any time and place.

The structure of this article is following. Section 2 presents related work, previous studies and related systems. VILLE-application and its' features is shown in section 3. Section 4 presents the discussion and section 5 describes the future work. At the end, section 6 presents the conclusions in brief.

2. Related work

With program visualization students are able to reduce the effort of the thinking process which aim is to construct the mental model of program execution process, its' actions and events which takes place in it [25]. In addition, the existence of graphical elements is not solely enough to describe important aspects of visualization in adequate level of detail. Petre [20] have noticed:"The question is not 'Is a picture worth a thousand words?', but 'Does a given picture convey the same thousand words to all viewers?' More over, a secondary cues or notations are required to aid the interpretation of graphical visualizations. [20]. Ben-Ari [1] claimed that graphical and textual descriptions have to be heavily synchronized, because a major problem of novice programmers is to choose important aspects and issues which are relevant and which are not relevant to a problem. Also, Naps [18] presents that visualization is proven to be useful only if we can engage the learner into learning session. For example, visualization and animation based system have been used successfully in teaching data structures and algorithms [8] [13] [14].

Jeliot 3 (Figure 1) is developed in University of Joensuu and it's a tool which can trace the execution of Java program language. As the execution goes forward step by step the evaluation of expressions are visualized with graphical symbols. Jeliot3 is mainly designed to support the learning process of novice programmers.

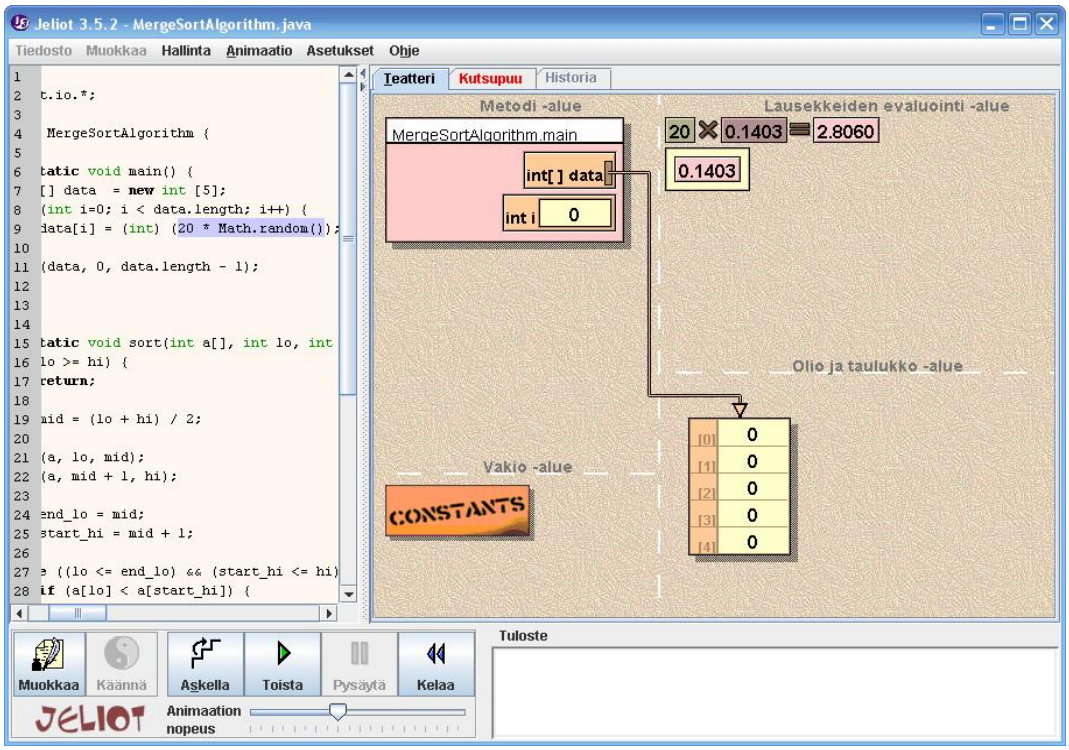

Figure 1: User interface of Jeliot 3

Kannusmäki et al. [11] evaluated Jeliot 3 with qualitative methods and pointed out, that only less talented students were willing to use jeliot3. The results showed also that Jeliot3 aided students' skills of perceiving if-statements and loops, understanding objects and tracing errors from program code.

In addition, BlueJ is an example about a static program visualization tool, which is developed at University of Monash in Australia. This tool is also meant to be used to teach teaching basic programming concepts. [12].

Over the past few decades many visualization and animation based applications have been developed such as JavaVis [19] which uses object- and sequence diagrams as visualizations, WYSIWYC (What You See Is What You Code)-model and direct manipulation of structure based on ALVIS LIVE! [9] and Raptor [5] which is programming environment that uses dataflow diagrams for its visualization. JHAVE [8], BALSA II [2], ZEUS [3], XTANGO [23] and TRAKLA2 [16] are algorithm animation systems which focus on visualizing data structures and algorithms such as sorting. In conclusion, Jeliot3 is same type of tool as VILLE is but all the other applications' purpose, visualization technique or abstraction level differs essentially from VILLE.

3. VILLE

VILLE is a program visualization tool, which can be used to create and edit various programming examples, and to observe events in the examples during their execution. Its main purpose is to support the learning process of novice programmers. Teacher can add programming course's programming examples to VILLE and then visualize their execution e.g. in lectures or over the web.

When planning VILLE's features, we considered e.g. Naps et al. [18] 11 points on visualization tool effectiveness.

3.1. Key features

Example collection. VILLE contains predefined set of programming examples, which are divided in different categories based on their subject. User can create new categories and examples or edit the predefined ones. By creating and editing examples, teacher can illustrate subjects that he thinks are essential in learning to program. Teacher can also make changes to the programming examples during lectures to visualize the effects of the modifications to students.

Language-independence. One of the most important aspects of VILLE is the possibility to show programming examples in several different programming languages. When user observes program execution in different languages, he sees how similar their basic functionalities are. For novice programmers it is more important to learn how different programming concepts really work than just the syntax of some programming language.

Defining and adding new languages. At present VILLE supports Java, pseudo code and C++ programming languages. Pseudo codes definition can be altered to suit the teacher's needs. It is also possible to define and add a new programming language to extend the language support.

Visualization row by row. Progress of the program execution is visualized by highlighting rows in the code. In addition to highlighting the program row under execution, VILLE also highlights previously executed row with a different colour. This makes the following of the program execution easier.

Flexible control of the visualization both forwards and backwards. User can move one step at a time, both forwards and backwards in the execution of the program. Examples can also be run automatically with adjustable speed. Moving backwards in the program execution isn't usually possible in similar applications (e.g. Jeliot3). Additionally, VILLE has an execution slider with which user can use to move to any place in program execution.

Breakpoints. The user can set breakpoints into any program code line and move between them both forwards and backwards. This functionality enables debug-based control and observation of the program execution. More over, backward tracing between breakpoints is not a standard feature in program code debuggers.

Code line explanation. Every code line has an explanation, in which all the program events on the line are clearly explained. Furthermore all possible outputs and variable states are shown. Code line explanation is a missing feature in many similar applications.

Role information. Variables' role information is integrated into the code line explanation. According to Sajaniemi and Kuittinen [22] this helps programming learning and enhances understanding of the program.

The parallel view shows the program code execution simultaneously in two different programming languages. This way the user can see how the execution progresses similarly regardless of syntactical differences between the languages.

Call stack. The moving of the program execution between different method calls is visualized with a call stack. When the execution shifts to a method, new window is opened on the call stack. The window remains on the call stack until the method is finished. When the execution leaves from the method, possible return value is shown on top of the call stack. Alternatively the visualization of the execution can be viewed in parallel view.

Publish examples. With the export feature VILLE's examples can be saved to an example collection. The example collection contains a version of VILLE, which has no example creation and modification functions. Teacher can use the export feature to publish programming course's programming examples in the web for the students to use.

3.2. The User Interface of VILLE

VILLE's user interface consists of three different views. When the application starts, main view is loaded. In the main view user can browse the examples and create new categories for the examples. Program code visualization and example modification have their own views, where user can navigate through the main view.

3.2.1. Main view

On the left side of the main view (Figure 2) lies programming example tree and buttons for controlling the application. With the buttons below the programming examples user can modify the examples (creation of new categories and examples, editing of examples and deletion of examples). The buttons above the examples can be used to change the language of the application between Finnish and English, exporting the examples to an

example collection and to move to the visualization of the chosen programming example. On the right side of the view lies the description and code listing of a chosen example.

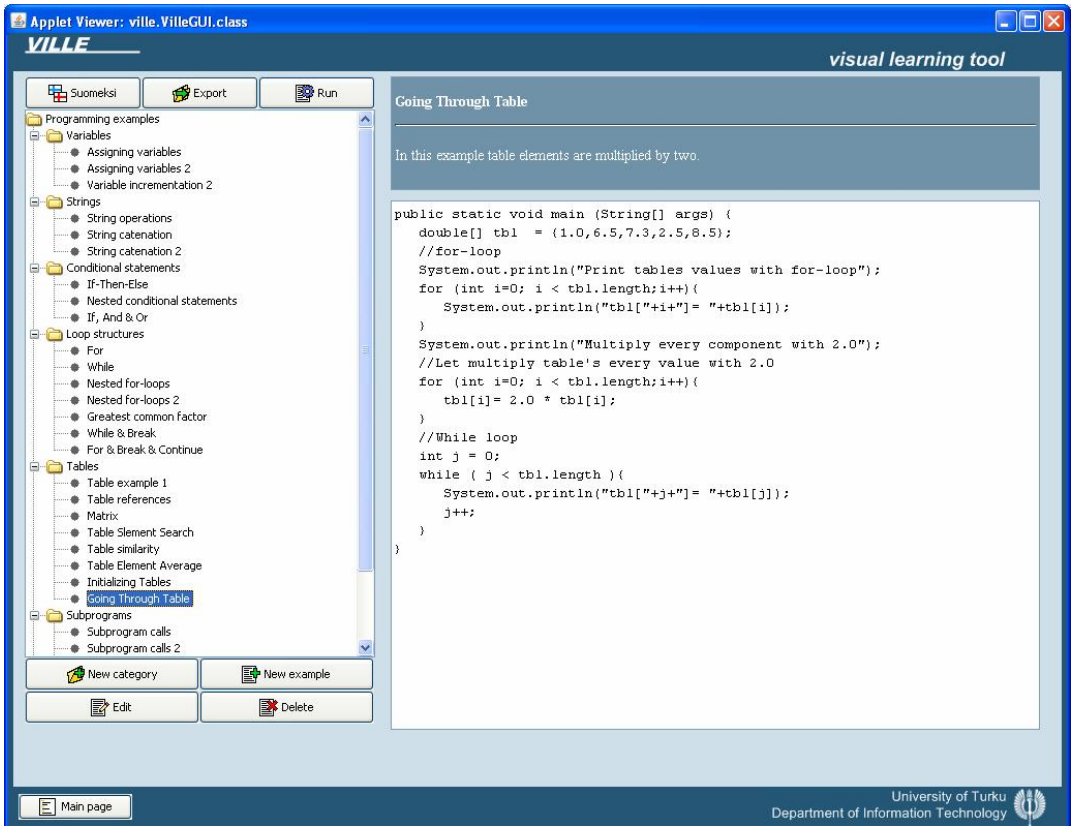

Figure 2: the main view of VILLE

3.2.2. Example Creation and Editing View

In the creation and editing view (Figure 3) user can add Java program code to the left text area. When the translate button is pressed, VILLE creates pseudo code and C++ translations of the Java program code and automatically generates explanations for each program line. Besides this user can write program example a description, which is shown in the main view, when browsing the examples. To save an example, user has to write a name for the example and destination file and choose the category where the example is shown in the main views example tree. Editing of the examples is done in the same view. In this case the example's information is loaded in the editing fields.

The translation of the program code is done with syntax files. There is a syntax file for each programming language and for the Finnish and English explanations. During the

translation the program code line is searched from the Java's syntax file and the translated to other languages using their syntax files' equivalent syntax line. The events of a program code are solved by going through the program in its execution order and saving an execution event for each event in the program in separate control file. The control file is used in the visualization view to control the visualization of the programming example.

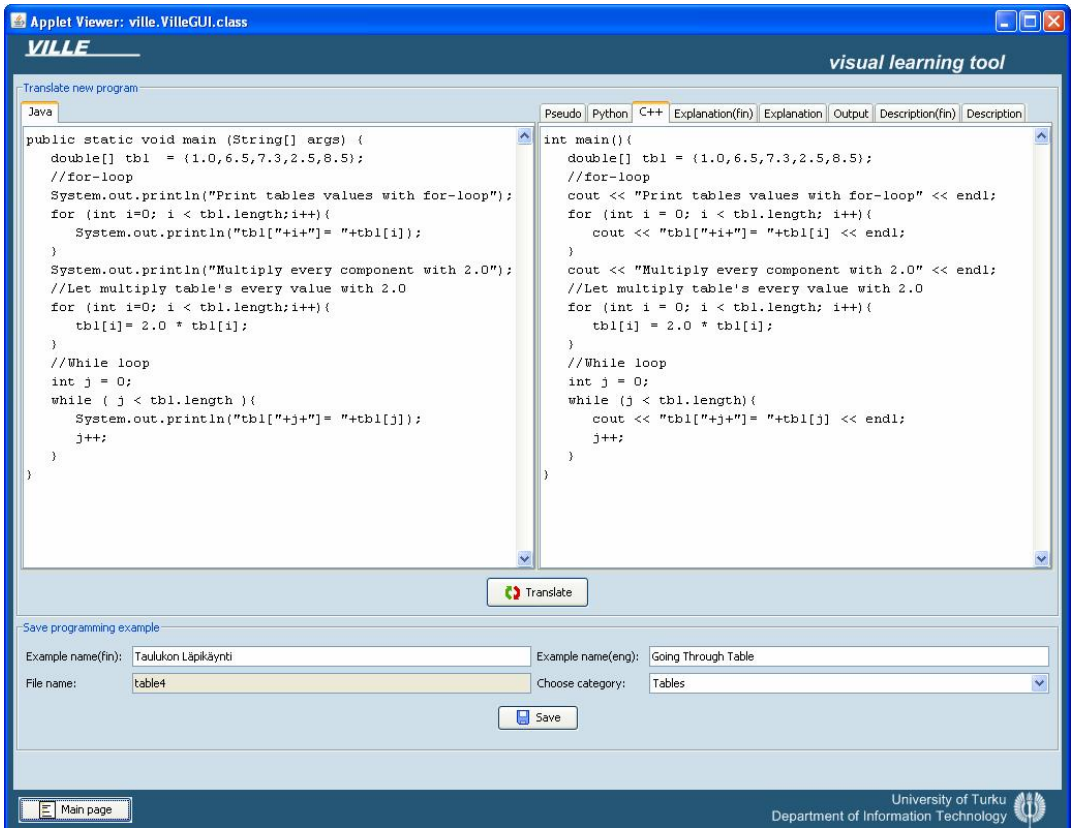

Figure 3: Creation and editing view of programming examples

VILLE supports Java syntax quite well. It understands basic variable types (int, float, double, Boolean), String classes main aspects, conditional statements (if, else-if and else), loop-structures (for and while), one and two dimensional tables and records. With these programming concepts, the basic functionalities of programming can be illustrated quite well.

3.3. Visualization view

In the visualization view (Figure 4) users can follow the execution of the programming examples. On the left side of the view lies control buttons for the visualization and the code listing of the programming example. With the controls user can start automatic

program execution or alternatively move one step at a time both forwards and backwards in the program. Moreover user can add breakpoints to any code line and move between the breakpoints with the program controls similar to debuggers.

Furthermore control area can be used to change the program code language to Java, pseudo code or C++, even during the execution. On the right side of the view lies call stack, on which the method calls are opened on their own frames. At the bottom of the view lie fields that change based on the program states and an execution slider that can used to move around in program execution. In those fields are shown code line explanations, program output, variable states and roles of the variables.

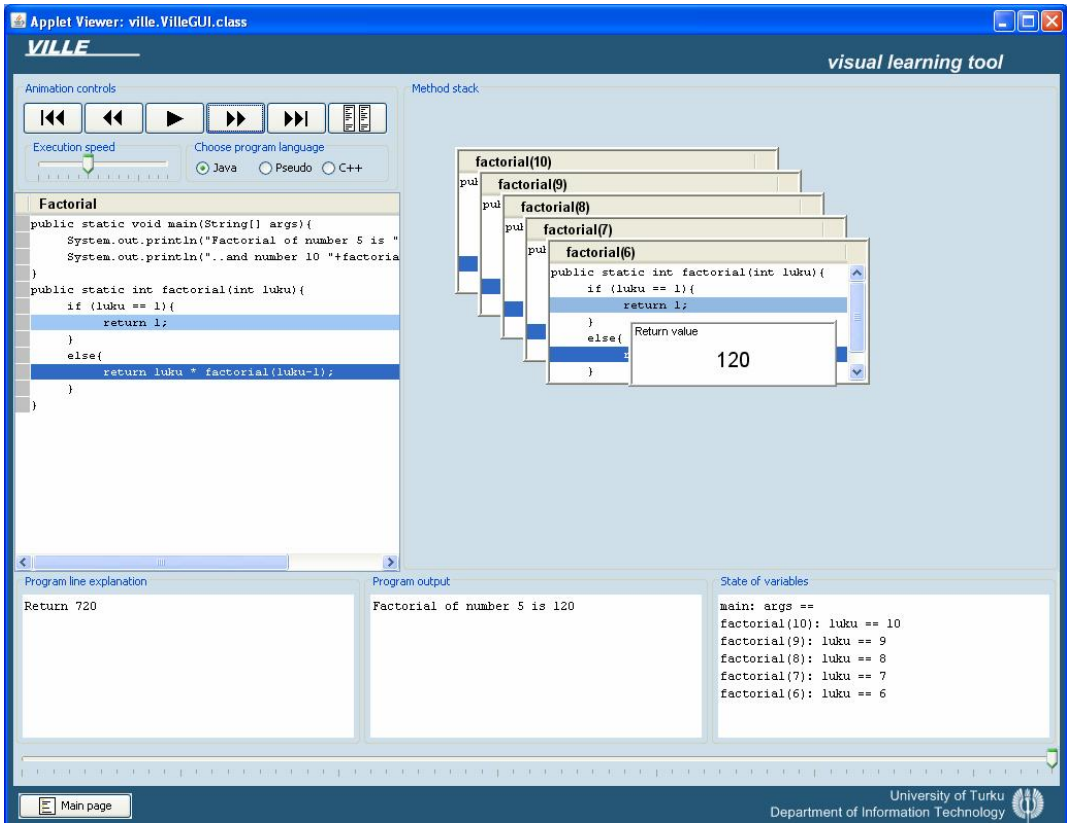

Figure 4: the visualization view of VILLE in call stack mode

The program execution in the visualization view can also be followed in so-called parallel view (Figure 5). Then the programming example opens in two parallel frames. User can select the language in both frames and compare the syntaxes and the program execution between two languages.

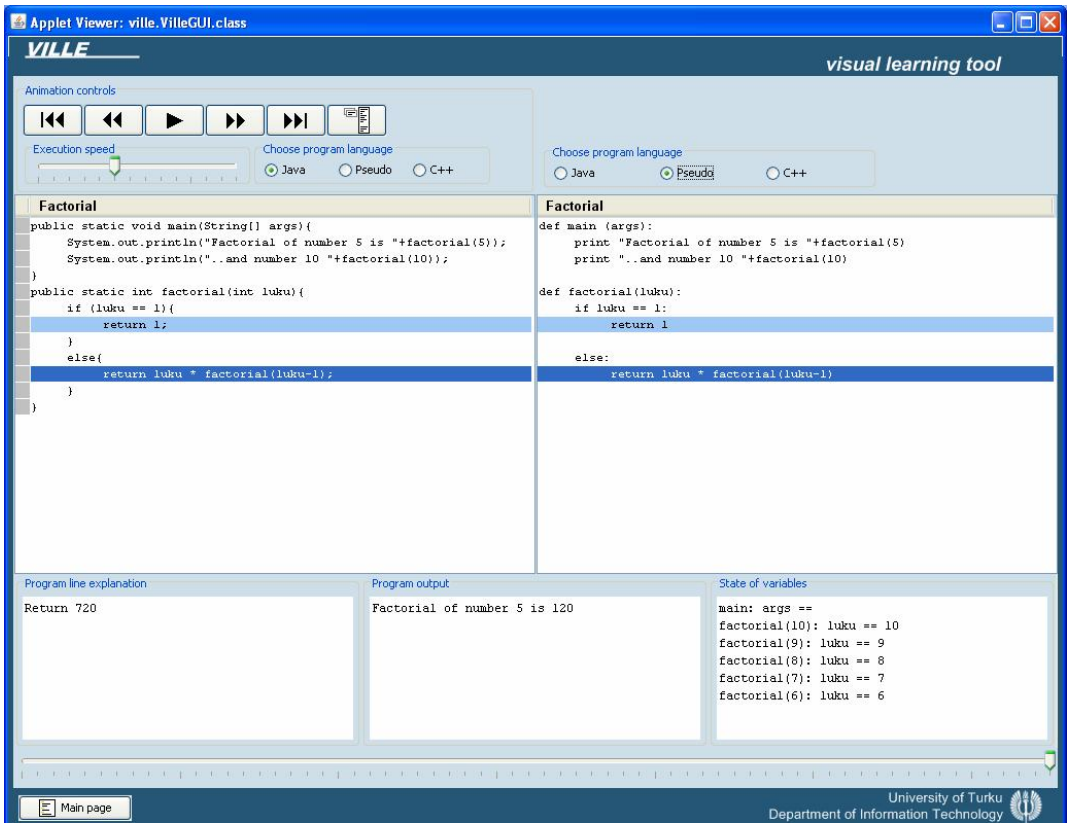

Figure 5: the visualization view of VILLE in parallel mode

4. Discussion

The primary goal of VILLE is to emphasize the programming language independency view of basic programming concepts and enhance the learning process. From the learner point of view is much more important to conquer the basic principles behind basic programming concepts such as loop-control structure regardless of programming language. Use of pseudo language is recommended and reasonable for basic programming courses and with VILLE the teacher can define him own pseudo language and visualize step by step its' execution and effects to variable state and program output. The concept of the roles of variables is gaining ground in computer science and especially in teaching of programming because it gives a programming paradigm independent and higher view of classifying the variables based on their behaviour. In addition, the concept of roles of variables provides also more information about the program code. In VILLE, a description line is generated for every executed code line in which these secondary cues like natural language explanation and the role information are include. These secondary cues according to Petre [20] aid students to understand the relations between different concepts and structures, which is essential part of the process of learn to program.

With VILLE the teacher can select the programming language to use in his basic programming course and even self-define it to suit his purposes. In addition, the teacher can create an example collection and publish it the internet where students can reach it at any time.

4.1. VILLE vs. Jeliot3

VILLE and Jeliot 3 are same type of applications which can execute step by step a program code, but there are also some differences between those two advanced tools. Jeliot3 support only JAVA programming language while VILLE support JAVA, C++ and pseudo language, and the latter can be also self-defined.

In VILLE, user can compare same program code on two different programming languages in the parallel view and by doing that, we can emphasize the language independency of the basic programming concepts from programming. This possibility aids the process of learning new programming languages or changing from one programming language to other.

Jeliot3 visualized the changed of value of variables with graphical symbols and execution is divided more detail than in VILLE in which the visualization goes as we call it row by row highlighting current line and previous line at the same time. Moreover, VILLE generates automatically a description line for every executed program code line. The description is in natural language and it includes the role information of variables and dynamic information of state of variables.

In Jeliot3 user can insert directly program code and animate it. To do the same in VILLE user have to change from one view to another; first enter the code in creation view and then change to visualization view to trace the execution of program code. Moreover, with Jeliot3 user can step the execution only forward and the next step can take place after the previous phase's visualization is completed. VILLE enables control of execution step by step both forward and backward in the phase defined by the learner. More over, in both systems the visualizations can be viewed as an animation and the execution speed can be altered.

Hence, VILLE visualization is more abstract and higher level than in Jeliot3. In addition, VILLE includes predefined examples with navigation and modification directly from user interface. These examples can be also published in the internet as an example collection. These feature are absent in Jeliot3.

5. Future work

In future, VILLE is going to be under continuous development process. At present, new program examples can be created in modification view and the goal is to integrate modification features directly to visualization view. Moreover, it is planned that VILLE

will support execution time pop-up questions about the trace of execution and changes in program states. Also, Jeliot's theatre view is planned to be integrated to the visualization view of VILLE, which gives much more detailed information about changes of variable states. Also, the roles of variables are now presented in textual form and thus some kind of graphical visualizations are going to be developed for better visualization.

6. Conclusions

VILLE is programming language independent tool for teaching novice programming. Teacher can create an example collection for his own course and even define the pseudo language itself. In the end, VILLE constitute a good amendment to teaching basic programming by offering more abstract concepts to handle basic programming related issues and with it student can easier conquer the barriers in the learning process of learn to program.

In fall, VILLE is going to be evaluated in first programming courses at University of Turku. The research will focus on engagement level of students, effectiveness and viability of VILLE and its' features

References

[1] Ben-Ari, M. 2001. Program Visualization in Theory and Practice. Informatik/Informatique 2, pp. 8–11.

[2] Brown, M.H. 1988. Exploring Algorithms Using Balsa II. IEEE Computer, 21(5), pp. 14–36.

[3] Brown, M.H. 1991. Zeus: A System for Algorithm Animation and Multi-View Editing. In the Proceedings of IEEE Workshop on Visual Languages, pp. 4–9. New York: IEEE Computer Society Press.

[4] Boada I., Soler J., Prados F. & Poch J. 2004. A Teaching/Learning Support Tool for Introductory Programming Courses. In the Proceedings of the Fifth International Conference on Information Technology Based Higher Education and Training. ITHET 2004, pp. 604–609.

[5] Carlisle, M.C., Wilson, T.A., Humphries, J.W. & Hadfield, S.M. 2005. RAPTOR: A Visual Programming Environment for Teaching Algorithmic Problem Solving. In the Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, St. Louis, Missouri, USA, pp. 176–180.

[6] Garner, S. 2006. The Development, Use and Evaluation of a Program Design Tool in the Learning and Teaching of Software Development. Issues in Informing Science and Information Technology, vol. 3, pp. 253–260.

[7] Grandell, L., Peltomäki, M., Back, R.-J. & Salakoski, T. 2006. Why Complicate Things? Introducing Programming in High School Using Python. In the Proceedings of the 8th Austalian Conference on Computing Education, Hobart, Australia, vol. 52, pp. 71–80.

[8] Grissom, S., McNally, M. & Naps, T. 2003. Algorithm Visualization in CS Education: Comparing Levels of Student Engagement. In the Proceedings of the ACM Symposium on Software Visualization, San Diego, California, pp. 87–94.

[9] Hundhausen, C.D. & Brown, J.L. 2007. What You See Is What You Code: A 'Live' Algorithm Development and Visualization Environment for Novice Learners. Journal of Visual Languages and Computing, vol.18, no. 1, pp. 22–47.

[10] Hundhausen, C.D., Douglas, S.A. & Stasko, J.D. 2002. A Meta-study of Algorithm Visualization Effectiveness. Journal of Visual Languages and Computing 13, pp. 259–290.

[11] Kannusmäki, O., Moreno, A., Myller, N. & Sutinen, E. 2004. What a Novice Wants: Students Using Program Visualization in Distance Programming Course. In the Proceedings of the Third Program Visualization Workshop (PVW'04), Warwick, UK, pp. 126–133.

[12] Kölling, M., Quig, B., Patterson, A. and Rosenberg, J. 2003. The BlueJ system and its pedagogy. Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology, vol. 13, no. 4.

[13] Laakso, M.-J., Salakoski, T., Grandell, L., Qiu, X., Korhonen, A. & Malmi, L. 2005a. Multi-Perspective Study of Novice Learners Adopting the Visual Algorithm Simulation Exercise System TRAKLA2. Informatics in Education, 4 (1), pp. 49–68.

[14] Laakso, M.-J., Salakoski, T. & Korhonen, A. 2005b. The Feasibility of Automatic Assessment and Feedback. In the Proceedings of Cognition and Exploratory Learning in Digital Age (CELDA 2005). IEEE Technical Committee on Learning Technology and Japanese Society of Information and Systems in Education. Porto, Portugal, pp. 113–122.

[15] Lister, R., Adams, S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B. & Thomas, L. 2004. A Multi-National Study of Reading and Tracing Skills in Novice Programmers. SIGCSE Bulletin, vol. 36, nro. 4, pp. 119–150.

[16] 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), pp. 267–288.

[17] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y., Laxer, C., Thomas, L., Utting, I. & Wilusz, T. 2001. A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-year CS Students. ACM SIGCSE Bulletin, vol. 33, no. 4, pp. 125–140.

[18] Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. & Velázquez-Iturbide, J. Á. 2002. Exploring the Role of Visualization and Engagement in Computer Science Education. In the Working group reports from ITiCSE on Innovation and Technology in Computer Science Education, vol. 35, no. 2, pp. 131–152.

[19] Oechsle, R. & Schmitt, T. 2002. JAVAVIS: Automatic Program Visualization with Object and Sequence Diagrams Using the Java Debug Interface (JDI). In the Diehl, S. (Ed.), Software Visualization. vol. 2269 of Lecture Notes in Computer Science. Springer-Verlag, pp. 176–190.

[20] Petre, M. 1995. Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming. Communications of the ACM, vol. 38, nro. 6, s. 33–44.

[21] Sajaniemi J. 2002. PlanAni - A System for Visualizing Roles of Variables to Novice Programmers. University of Joensuu, Department of Computer Science, Technical Report, Series A, Report A-2002-4.

[22] Sajaniemi, J. & Kuittinen, M. 2003. Program Animation Based on the Roles of Variables. In the Proceedings of the 2003 ACM Symposium on Software Visualization, San Diego, Kalifornia, pp. 7–ff.

[23] Stasko, J. 1992. Animating Algorithms with XTANGO. ACM SIGACT News, vol. 23, no. 2, pp. 67–71.

[24] Stern, L., Søndergaard, H. & Naish, L. 1999. A Strategy for Managing Content Complexity in Algorithm Animation. In the Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology In Computer Science Education, Krakova, Puola, pp.127–130.

[25] Tudoreanu, M.E. 2003. Designing Effective Program Visualization Tools for Reducing User's Cognitive Effort. In the Proceedings of the 2003 ACM Symposium on Software Visualization, San Diego, California, 105–ff.

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www.tucs.fi

University of Turku
- Department of Information Technology
- Department of Mathematics

Åbo Akademi University
- Department of Information Technologies

Turku School of Economics
- Institute of Information Systems Sciences

ISBN 978-952-12-1917-7
ISSN 1239-1891