

Jeliot 3, an Extensible Tool for Program Visualization

Andrés Moreno, Niko Myller and Roman Bednarik
Department of Computer Science
University of Joensuu
Joensuu FIN-80110 Finland
+358 13 251 7977

firstname.lastname@cs.joensuu.fi

ABSTRACT

Jeliot 3 is a program visualization tool that can be used in introductory courses of programming. It animates a large set of Java programs and can interact with the BlueJ IDE. Jeliot 3 has been tested and shown to be useful for novice students with difficulties in programming. In this paper, we discuss how Jeliot 3 has been designed to allow for extensibility with modular design. This allows both original and other developers to add different visualization paradigms to Jeliot 3, or to connect Jeliot 3 with other environments. We present how Jeliot 3 has been integrated with others system in order to ease its use by teachers and students.

Keywords

Jeliot, Program Visualization, IDE, BlueJ

1. INTRODUCTION

Jeliot 3 [8] (see also [6]) is a Program Animation tool aimed to support novice programmers and CS students in introductory programming courses. It displays the execution of Java Object Oriented (OO) programs by animating the source code evaluation. Most of the Java language is currently supported and animated by Jeliot 3, starting from simple assignments, through method calls, to the object allocation.

Jeliot 3 is the successor of Jeliot 2000 system. Two main reasons led the development of Jeliot 3 after Jeliot 2000: to incorporate OO support and to redesign the software architecture for better modularity and extensibility. The visualization paradigm used in Jeliot 3, a representation of how the virtual machine works, remained the same as it was designed in Jeliot 2000. Thus, the benefits found in Jeliot 2000 can be translated to Jeliot 3. Ben-Bassat et al [1] found that Jeliot 2000 was helpful for mediocre students. In a classroom experiment, Jeliot 2000 helped students to create viable mental models of the actual Java program execution and provided them with a vocabulary to describe the execution.

Jeliot 3 is currently being used in some Finnish and Israeli high schools and universities. Moreover, we have had reports of usage in USA, Germany, Denmark, England, and South Africa. In this paper, we introduce Jeliot 3 as an extensible PV tool that 1) allows CS educators to effectively provide students with dynamic animations of computer programs, and 2) gives students a possibility to implement their own programs in a general purpose programming language and visualize their programs execution. We invite all CS educators and students to download and use Jeliot 3 and by reporting their experiences to contribute to next stages of the tool development.

2. JELIOT 3

Program comprehension, the ability to understand programs written by others, is a central skill to programming. While expert programmers create accurate mental models during comprehending programs, novices need to be supported in this task. Program visualization can be one of the ways how to provide students and novices to programming with viable mental models of program execution. Several PV tools have been developed to address this potential; however, the results of their evaluations are often mixed.

Animation of a program in Jeliot 3 takes places in the so called *theater*, that is a larger panel on the right of the code, divided into the following four discrete areas (see Fig. 1): a) *method area*, where method frames containing variables are stacked, b) *constant area*, where constants and static variables are displayed, c) *object area*, where objects and arrays are allocated and referred by the variables in the method area, and d) *expression evaluation area*, where expressions are evaluated.

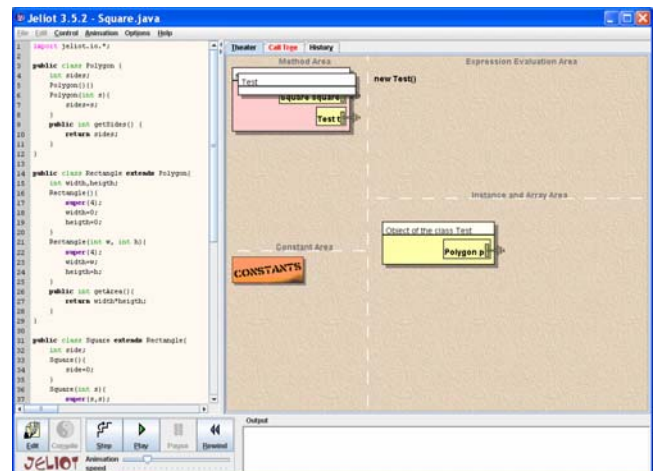


Figure 1 : Jeliot 3 user interface

Considering the actual implementation of Jeliot in a programming course, it can be employed both as a demonstration tool that teachers use to show programming concepts to students, and as a development tool that students use to complete assigned exercises. As a demonstration tool, Jeliot 3 can be used as it is. The only requirement is a suitably chosen example to show and explain the concept clearly to the students. A Java Web Start launcher has been developed for online courses. The launcher will start Jeliot from the web site. It can be modified to load an example from the web at the same time it starts [5]

If a student wants to use Jeliot 3 as a development tool, they can use it in several different ways. Firstly, they can use Jeliot 3 as it is. Jeliot 3 provides a basic code editor with some advanced features, such *color syntax highlighting* and *search & replace*. Although possible, the use-case of Jeliot 3 as a Java program editor is not the primary purpose the tool was developed for. Especially, more advanced programmers can find the abilities of the tool insufficient. If a student wants to use a more advanced editor, Jeliot 3 is available as a plug-in to “Editing Java Easily” [4], an editor developed at the University of Karlsruhe. Finally, Jeliot 3 has an extension that can be incorporated into BlueJ [3], a novice-oriented IDE. The extension adds a menu selection to the interface with which students and teachers can animate their projects or single method calls directly from BlueJ.

3. THE MODULAR DESIGN OF JELIOT 3

When we decided to improve Jeliot 2000, by 2002, the main goal was to add a support for object oriented programs. Unfortunately, the design of Jeliot 2000 imposed many limitations when trying to extend its capabilities [9]. Thus, a complete redesign was needed resulting into Jeliot 3.

Interpretation and animation of Java programs were separated into two modules. On one side, DynamicJava [2], an open source Java interpreter, processes the user program. On the other side, Jeliot 2000’s graphical engine creates a respective animation of the program interpretation. To connect both parts, an intermediate code [7], the MCode, was designed.

MCode is a textual representation of the interpretation of a running program, or a program trace. It not only describes the changes in variables and stacks, as a normal Java debugger would do, it also details the operations that produced those changes. All this information is required, to animate every step in the execution of a program. At this moment, there are two interpreters of the MCode in Jeliot 3: one that produces the complete animation of the program and another one that builds a tree of the method calls during the execution. Program animation designers can build their own animations and integrate them into Jeliot 3 by writing a new MCode interpreter.

MCode aims to be language independent. For example, a C++ interpreter could produce the MCode for a C++ program and send it to Jeliot 3 that would, in turn, generate the animation for such program.

Readers interested in the design of the environment and the intermediate code should refer to [7] and [9] for detailed explanations of Jeliot 3 development.

4. FUTURE WORK AND CONCLUSION

Jeliot 3 is designed to help to overcome the barriers of novice learners. Owing to its modular design and publication under GNU General Public License (GPL), advantages of Jeliot3 over Jeliot

2000, Jeliot 3 has made it possible for different people around the world to develop different extensions and plug-ins exploiting by using Jeliot. Furthermore, Jeliot 3 can be now used to teach object-oriented programming as well.

We want to develop Jeliot 3 further in the directions of adaptive program visualization, in which the user’s abilities, knowledge, goals and intentions are better supported by the tool.

Another future direction is to address a collaborative use of Jeliot in which several users can either face-to-face or online program together and utilize the visualization to better support their work.

Jeliot 3’s homepage [6] provides a set of materials and information including documentation and publications related to Jeliot 3 or its extensions and plug-ins mentioned here. We invite everybody to download, install and try Jeliot 3, and hope to receive feedback from a variety of users, educators, and designers of educational tools.

5. REFERENCES

- [1] Ben-Bassat Levy, R., Ben-Ari, M., Uronen, P. A. The Jeliot 2000 Program Animation System, *Computers & Education*, 40, 1 (Jan. 2003), 1–15.
- [2] Hillion, S. *DynamicJava, Koala Project*, WWW-page, <http://koala.ilog.fr/djava/>, (accessed 27.10.2005).
- [3] Kölling, M., Quig, B., Patterson, A., Rosenberg, J. The BlueJ system and its pedagogy. *Journal of Computer Science Education*, 13, 4 (Dec. 2003), 249 - 268.
- [4] Küstermann, R. *Editing Java Easily*, WWW-page, <http://www.eje-home.de/>, (accessed 27.10.2005).
- [5] Küstermann, R., Ratz, D., Seese, D. Effektive Java-Grundausbildung unter Einsatz eines Learning Management Systems und spezieller Werkzeuge. *Proceedings of the INFOS' 05*, pp. 10, 2005.
- [6] Jeliot Team, *Jeliot 3 homepage*, WWW-page <http://www.cs.joensuu.fi/jeliot/>, (accessed 27.10.2005).
- [7] Moreno, A. *The Design and Implementation of Intermediate Codes for Software Visualization*, Master’s thesis, Department of Computer Science, University of Joensuu, Joensuu, Finland, 2005.
- [8] Moreno, A., Myller, N., Sutinen, E., Ben-Ari, M. Visualizing Programs with Jeliot 3, *Proceedings of Advanced Visual Interfaces, AVI 2004*, 2004, 373–376.
- [9] Myller, N. *The Fundamental Design Issues of Jeliot 3*, Master’s thesis, Department of Computer Science, University of Joensuu, Joensuu, Finland, 2004.