Jeliot 3

Niko Myller, Andrés Moreno, Roman Bednarik, Erkki Sutinen
Department of Computer Science and Statistics, University of Joensuu
P.O. Box 111, FI-80101 Joensuu, FINLAND
Tel. +358 13 251 7929, Fax. +358 13 251 7955

Ronit Ben-Bassat, Mordechai Ben-Ari
Department of Science Teaching, Weizmann Institute of Science
Rehovot 76100, ISRAEL
Tel: +972 8 934 2940, Fax: +972 8 934 4115

Jeliot 3 is a program animation system intended for teaching and learning introductory programming. All steps of the execution of a program (expression evaluation, control flow, method calls, parameter passing, variables and object allocation) are animated on a “stage,” while the source code is displayed in a separate pane next to the stage. The animation is fully automatic: it does not require any preparation by a student or instructor so that any modification of the source program is immediately animated. Students can execute the animations continuously or step-by-step. Jeliot 3 has been employed in numerous programming courses worldwide and empirically evaluated in several classroom and laboratory studies.

This courseware has been added to the NEEDS database. It can be downloaded from the Jeliot web site at http://cs.joensuu.fi/jeliot/downloads.php. The software is written in Java and is portable. The size of the setup file is 1.5 megabytes; a version that is bundled with the Java Runtime Environment is available (15 megabytes).

The developers of Jeliot are listed above and have been contacted regarding this submission. Niko Myller, Andrés Moreno, Mordechai Ben-Ari and Erkki Sutinen hold the copyright of Jeliot 3 and have licensed it as open source under the GNU General Public License. Other contributors to this project have been contacted and are listed here: Gil Ebel, Weizmann Institute of Science, Roland Küstermann, University of Karlsruhe, Antoine Pineau, Polytech Nice-Sophia and GL Trade, Noa Ragonis, Weizmann Institute of Science, Timo Rongas, Lappeenranta University of Technology, Jorma Tarhio, Helsinki University of Technology, Pekka Uronen, University of Helsinki.

The software uses following open source components: DynamicJava developed by Stephane Hillion. JEditTextArea developed by Slava Pestov, Artur Biesiadowski, Clancy Malcolm, Jonathan Revusky, Juha Lindfors and Mike Dillon. Kunststoff Look&Feel developed by INCORS. AVInteraction developed by Guido Rößling and Gina Häußge. Licensing terms for these components can be found from the Appendix A. Authors of these components have not been contacted regarding this submission.

We are authorized to submit the software for the Premier competition as the designers and developers of Jeliot 3. All the licensers authorize NEEDS to become a non-exclusive distributor of this courseware as it is submitted based on the licensing terms.
Contact Information for Deans and Department Heads

University of Joensuu

Dean, Faculty of Science
Prof. Jussi Parkkinen
Faculty of Science
University of Joensuu
P.O. Box 111
FI-80101 Joensuu
Finland
Tel. +358 13 251 7932, Fax. +358 13 251 7955
Email. jussi.parkkinen@cs.joensuu.fi

Head of Department, Department of Computer Science and Statistics
Prof. Erkki Sutinen
Department of Computer Science and Statistics
University of Joensuu
P.O. Box 111
FI-80101 Joensuu
Finland
Tel. +358 13 251 7934, Fax. +358 13 251 7955
Email. erkki.sutinen@cs.joensuu.fi

Weizmann Institute of Science

Dean, Feinberg Graduate School
Prof. Yosef Yarden
Feinberg Graduate School
Weizmann Institute of Science
Rehovot 76100 Israel
Tel. +972 8 934 2924, Fax: +972 8 934 4114
Email. Yosef.Yarden@weizmann.ac.il

Head of Department, Department of Science Teaching
Prof. Avi Hofstein
Department of Science Teaching
Weizmann Institute of Science
Rehovot 76100 Israel
Tel. +972 8 934 3894, Fax: +972 8 934 4115
Email. nthofstn@wisemail.weizmann.ac.il
**Introduction and Background**

Programming is a complex problem-solving domain because a small amount of source code in programming languages gives rise to complex runtime structures that are hidden within the computer. Novice programmers have little knowledge about program execution and may see it as an abstract and nondeterministic process that is hard to understand. Jeliot was developed to support learning of novices by visualizing program execution, thus facilitating the development of viable mental models.

Several versions of Jeliot have been developed, used and researched for over 10 years in an international collaboration; this long-term project has given us a strong knowledge base on program visualization (Ben-Ari et al., 2002), and its contribution to programming education and learning (Moreno et al., 2004). The key design principle has been that of learning-by-doing: students should have an animation tool which helps them construct a visual and mental representation of a program as easily as possible. The two representations of a program, namely the code and its animation, should match the mental image of the student so that he or she could concentrate on the comprehension process instead of being misled by disturbing visual clues.

To achieve that degree of transparency and integration in an animation environment requires a sophisticated technical design, in which an interpreter or compiler automatically generates the animation of a program. This means that a student can write almost any program without worrying about how to visualize it. The learning process takes place at the level of coding a program and studying the two representations—the textual and the visual—without creating yet another representation in a visualization language.

**Description of the Software**

Jeliot 3 is a computer environment that has a novice-oriented GUI. The interface consists of menus and buttons for the text editor and VCR-like buttons for the animation display. The use of the tool during a lecture is facilitated by keyboard shortcuts so that lecturers do not need to spend time with the mouse interaction during lectures. Jeliot 3 automatically visualizes the execution of Java programs by illustrating the data and control flow, as well as the object-oriented features of the program. Jeliot 3 can visualize a large subset of novice Java programs and is freely distributed under the GNU General Public License (see [http://cs.joensuu.fi/jeliot/](http://cs.joensuu.fi/jeliot/)), so that anybody can download and use the program and make modifications to it.

The internal technical design of Jeliot 3 makes the system extensible and allows for adding new types of visualization. This is achieved by defining an intermediate language for the interface between the interpreter and the display software. This was used, for example to implement a call tree visualization of the program, where only the method call tree of the program is shown.

The user interface of Jeliot 3 is shown in Figure 1.
The interface consists of four main areas. A code editor on the left hand side shows the program code; during program visualization, the currently executed statement or expression is highlighted. A control panel in the bottom left corner is used to control the animation with VCR-like buttons. On the right hand side of the window, the largest area of the Jeliot’s interface is occupied by a visualization view showing the execution state of the program. Moreover, the animation view is further divided into four areas that show the learner structures in a semantically meaningful form:

- The method area shows the data structures of currently executed method and the local variables (the activation records);
- The expression evaluation area shows expressions as they are evaluated step-by-step (the CPU and its registers); in addition, messages are shown to the student to draw attention to control flow;
- The constant area contains classes together with their static variables, and the constant "box" where the literal constants appear from.
- The instance and array area shows the visualization of the arrays and objects containing their fields (the heap).

Finally, an output console lies in the bottom right corner of the window and shows the output of the executed program. For input, an input console appears as needed.

There are two specialized visualizations on the respective separate tabs of the tabbed pane: the call tree and the execution history which enables the user to go backwards in the animation.

In a typical session with Jeliot, a user writes or loads a program, and then compiles it. After the program is compiled, a visualization view is opened for animation. Jeliot shows the execution either step by step or continuously. The user can control the animation with VCR-like buttons: animation speed, pause and rewind. Reverse execution is not possible but user can view the execution history which is a stepwise recording of the current execution.
Jeliot 3 in Programming Education

Introductory programming is one of the most difficult subjects to learn in computer science curricula and dropout rates are very high (McCracken et al., 2001; Kinnunen and Malmi, 2006). A primary reason for this is that the source code of a program in a programming language is a small static entity that gives rise to extremely complex and dynamic behavior when the program is executed on a computer. This makes it difficult to form viable mental models of the execution of a program (Ben-Ari, 2001). Creating an accurate mental model, however, is the integral part to successful comprehension and programming. Jeliot has been designed to bridge this gap by visualizing and animating the execution of a program in full detail. Its measured contribution to learning is described in the next section.

Jeliot supports the Java programming language, which is the language used almost universally in teaching of introductory programming. Jeliot has been used in introductory courses at many types and levels of institutions: high schools, teacher-training colleges and universities. The flexibility is achieved because visualizations in Jeliot are completely automatic. The student or instructor need only write a program in Java as usual and Jeliot will instantly animate its execution. No external tools are therefore needed. This is in contrast to many approaches to visualization which require the instructor to devote a large amount of time to construct visualizations. (In some approaches, a professional web designer or skills in graphics and design are even required.)

Jeliot can be downloaded for free and is easy to install and use. Because of this free access, there are no exact data available on the number of downloads and its use. Furthermore, since it is independent of any specific syllabus, it can be adopted by an individual instructor, without requiring a departmental decision or commitment.

There are two primary modes of using Jeliot: either by the instructor or by the students themselves.

- The instructor can use Jeliot for in-class demonstrations with a screen projector. This saves an enormous amount of time that would be required for drawing execution-time programming structures on a white board or with presentation software. Furthermore, the instructor can respond to or initiate “what-if” questions, or to clarify difficulties that occur in the lab: by simply modifying the source code, the visualization is instantly updated. Jeliot has History option so that the instructors can stop the animation at any stage and repeat previous steps that the students found difficult.

- The second mode of using Jeliot is by the students themselves in inquiry-type exercises. Instructors can build a directory that contains all the needed programs for studying a new topic, and then let the students explore it themselves. Jeliot is extremely simple to use so that the instructor can assign programs to investigate with the assurance that the students can carry out an investigation on their own. The call tree option is used to study the flowing after the execution of recursion. Students enjoy using this option because it helps them check their own recursive programs. Finally, the vocabulary used by Jeliot helps the students talk about their own programs and problems.

Currently, we are looking into the effects of Jeliot in other learning situations such as collaborative learning.

Because visualizations are generated automatically and instantly, the use of Jeliot is compatible with currently existing pedagogical approaches and their associated examples and exercises. For example,
Jeliot has been used with the two competing approaches for teaching introductory programming: “objects-first” and “imperative-first.”

**Empirical Evaluations**

In past years, numerous empirical evaluations of Jeliot 3 have been carried out in order to understand and demonstrate its applicability for teaching and learning of programming. These focused on both long-term effects of using visualization during a complete course to improve student learning, as well as on short-term effects of visualization on visual attention and mental models created during interaction with the tool. In addition, studies of user needs, usability and user satisfaction have been conducted. In the following, we briefly overview some of the investigations that were instrumental to the design and development of the various versions of Jeliot; for detailed reports, please consult the publications listed at the end of this document.

Jeliot 2000, a predecessor of Jeliot 3 with a similar GUI and similar features, was evaluated by Ronit Ben-Bassat Levy in an experiment that is as close to a controlled experiment as one could hope for in an actual educational setting: two classes, one using Jeliot and one as a control group (Ben-Bassat Levy et al., 2003). The experiment was carried out on tenth-grade high school students studying an introductory course on algorithms and programming, and the results were evaluated both quantitatively and qualitatively. The experiment was run for a full year, so that the students could overcome the difficulties inherent in using a new system.

The experiment was carried out by testing learning of each new concept as it was studied during the year. In addition, an assignment at the end of the year and a follow-up assignment during the next school year were used to investigate long-term effects. The quantitative test results were supplemented with individual problem-solving sessions which were taped and analyzed. For details of the experimental setup and results, see (Ben-Bassat Levy et al., 2003). We can summarize the results and conclusions as follows:

- The scores of the animation groups showed a proportionally greater improvement, and their average matriculation exam score was the same as that of the control group, even though the latter contained stronger students.
- Mediocre students were shown to profit more from animation than either strong or weak students, though the grades of the latter two groups do not suffer.
- The animation group used a different and better *vocabulary of terms* than did the non-animation group. Verbalization is an important step to understanding a concept, so for this reason alone, the use of animation can be justified.
- Students from the animation group used a *step-by-step method of explanation*, and some even used *symbols from Jeliot* in order to show the flow of values. Students from the control group expressed themselves in a generalized and verbose manner. This difference in style continued into the next year.

Here is a summary of a problem-solving session on nested if-statements that demonstrates how the above conclusions were arrived at:

- In the control group, only the stronger students could answer the questions, and only after many attempts. They were not sure of the correctness of their answers and had difficulty explaining them.
- The stronger students of the animation group also had difficulties answering this question. They did not use Jeliot because they believed that they could understand the material without it.
• The weaker students of the animation group refused to work on the problem, claiming that nested if-statements are not legal, or that they did not understand the question.
• The mediocre students of the animation group gave correct answers! They drew a Jeliot display and used it to hand simulate the execution of the program.

Gil Ebel researched the use of Jeliot to teach computer science and programming in a special education high school class, whose students suffered from a variety of emotional problems such as hyperactivity (Ebel and Ben-Ari, 2006). Jeliot was used by the teacher for demonstrating programming concepts, and its use was integrated into the normal classroom activities, such as lecturing and problem solving. By analyzing videotapes of the students, it was shown that the students’ disruptive behavior was reduced from its usual high level to practically nothing! Furthermore, students were found to be attentive to the visual presentations. This research validates claims that visualization in general, and Jeliot in particular, can improve not only cognitive learning, but also affective factors like motivation and attention that are important in classrooms.

Investigations of the differences between expert and novice programmers helped us to understand the effects of expertise on the task performance. In a laboratory study, Bednarik et al. (2006) investigated the visual attention patterns and overall interaction patterns of novice and expert programmers when comprehending a program with the help of Jeliot. When the visualization of the program was active, programmers spent most of their time attending to the visualization of the program, specifically (depending on the program behavior) programmers changed their visual attention strategies and methods, and expression evaluations were visually attended to.

Feedback from computer science students on the usability, the pedagogical and the technological issues of Jeliot has been carried out by registering the tool in the TUP online evaluation service (http://cs.joensuu.fi/~tup). To the date, about 55 evaluations of the tool have been collected and are available online (excerpts from the evaluations are attached to this application).

Adaptations and Applications of Jeliot 3

Releasing Jeliot 3 as an open-source application has fostered many interesting collaborations that have led to innovations. One of those has been with the EJE team (http://ilias.aifb.uni-karlsruhe.de/ruk/) at the University of Karlsruhe, Germany. The EJE team integrated Jeliot 3 into ILIAS (http://www.ilias.de/ios/index.html), an open source Learning Management System used at their institution. This project resulted in the ability of Jeliot 3 to be initiated directly from any lecture webpage (Küstermann, 2005). The EJE team has developed a Java editor, EJE (Editing Java Easily), and Jeliot 3 has been integrated as a visualization plug-in to EJE. The cooperation between EJE and Jeliot 3 teams clearly benefited both tools. Jeliot 3 can make use of an improved environment, with an editor and other features, while EJE can visualize the code that the user is creating.

We have also developed a plug-in for one of the most widely used programming environments in introductory courses, BlueJ. BlueJ is an integrated development environment that is oriented to novice students of Java. It focuses on the graphical creation of classes by means of an UML notation, and on direct manipulation with classes by creating objects and interacting with them in the object bench (Kölling, 2003). The plug-in architecture enables interaction between BlueJ and Jeliot 3: it combines the standard UML notation and interaction with objects, such as object creation and method calling, with the animation created by Jeliot visualization engine to represent the runtime behavior of the program or its parts.
JeCo (Jeliot Collaboratively) (Myller and Nuutinen, 2006) is a fully functional prototype that extends the concept of Woven Stories (Gerdt et al., 2001) into a co-operative programming environment. Woven Stories is a co-authoring tool, where users can concurrently and graphically create text documents by adding nodes to a graph. JeCo supports the novel concept of collaborative program visualization. It allows the user to visualize programs, send these visualizations to other users of JeCo, comment other users’ programs and visualizations and chat with other users. This allows the collaborative use of Jeliot in web-based courses.

For programming web-based courses, a module for Moodle, a learning management system (LMS), has been developed. The module extends the LMS so that teachers can easily add visualization activities based on Jeliot 3 and their own exercises to their online courses. The implementation is based on the modifications of Jeliot 3 that enable initiating it from any webpage. The module can register in the LMS or other user-modeling servers the result of the programming tasks as well as the questions students have to answer during the animation.

Open-source software also permits the extension of Jeliot into new domains. For example, Fionnuala O'Donnell, from the Trinity College at Dublin, Ireland, developed a simulation framework for teaching distributed systems concepts (O'Donnell, 2004), called FADA (Framework Animations of Distributed Algorithms, see Figure 3 for a screenshot of FADA). It makes use of Jeliot 3 source code to implement the GUI and other parts of the tool. Thus, the authors of FADA could focus on the visualization side of distributed systems, and make use of a novice oriented GUI of Jeliot 3.

At the moment, we are aware of several other projects that have taken the Jeliot 3 platform and technology and extended it to new application domains. For example, an industrial visualization tool based on the technology is being implemented with the support of the authors. Also, extensions to support new programming languages have begun to appear. A prototype for animating C++ using Jeliot 3 has been developed in the Institute of Technology Blanchardstown, Ireland.

**Comments from Users**

Statement from research associate, Roland Küstermann at University of Karlsruhe on using Jeliot in teaching and learning programming at their institution:

"For almost 3 years we have used Jeliot in three main scenarios. Having developed an enhanced blended learning concept, Jeliot supports the students in understanding given examples directly from within the learning context. Additionally, we use Jeliot in the classroom when demonstrating new concepts. Last but not least, Jeliot's visualization engine has been plugged into our integrated development engine for novices called Editing Java Easily. Especially during the first months of studying computer science, students use Jeliot quite often to deepen their proficiency. Both students and teachers are very satisfied with Jeliot since it allows improving the support service without the need of more personal resources, during classes as well as when preparing for exams."

Statement from Dr. Noa Ragonis, head of the Department of Computer Science, School of Education, Academic College Beit Berl, Israel:

"I have been using the Jeliot animation system for 5 years in pre-service courses for students intending to teach computer science and in in-service courses for teachers. There is no question that the saying ‘A picture is worth a thousand words’ characterizes the reaction of everyone who encounters the system. Other comments include: ‘how clear!’, ‘I needed this yesterday when the students had trouble understanding ...’; ‘it is amazing how you see ...’.
In my opinion, the most significant advantage of the Jeliot animation system is that it simultaneously shows the representation of data in memory with the representation of the memory structures associated with the program, in particular, those structures that pertain to control flow between statements, methods and objects.

Experienced teachers, who are well aware of the difficulties of novices learning computer science, immediately see the potential of the Jeliot system to help them overcome learning difficulties and even avoid these difficulties in the first place. This is especially important when teaching object-oriented programming as is done in most courses.

I see the advantages of Jeliot as follows:

- It enables the student to track activities that occur in the computer's memory, such as memory allocation, updating of the values of variables, copying of values from one variable to another, and so on.
- It enables the student to track control flow for if-statements and looping statements, including the conditions for entering and for leaving a loop.
- It enables the student to track programs that include methods, in particular it shows that: when a method call to another method is made, the control is passed to the second method, and the first method is inaccessible until the second one has finished, and then the execution continues from the statement after the method call; and when a method terminates its data cannot be accessed any more.
- Jeliot's ability to display a call tree is important when studying recursive programs.
- Jeliot fully supports animation of object-oriented programming. Understanding the allocation and initialization of an object as the result of a constructor is very difficult, but these are clearly visualized by Jeliot. This is of vital importance in the case of programs consisting of multiple classes, as the control flow between objects of the classes is complex.
- Strings are an important data type and are generally introduced to students at an early stage. Jeliot visualizes strings as objects and facilitates understanding the non-intuitive aspects of computations with strings.

All these constructs are abstract and hidden from a student who just runs programs, so the contribution of the concrete representations of the Jeliot visualization is significant. The ease of modifying programs to explore new concepts facilitates learning them without misconceptions.

As an experienced lecturer in computer science and as an experienced trainer of teachers, I welcome the initiative to develop Jeliot as a pedagogical tool of the highest quality. I am an avid user and will continue to do so in the future.”

**Comparison of Jeliot 3 against the Premier Awards**

**Evaluation Criteria**

**Learning Objectives**

Jeliot 3 is designed to be used for teaching and learning of programming, but it does not impose restrictions on the content of the syllabus or the instructors' methods of teaching. The ultimate goal is to learn to program in a higher-level language, which is a ABET accreditation criteria for computer science and information systems. Programming is also a required or recommended subject for students in science, engineering and other fields. As discussed in the Section *Jeliot 3 in Programming Education*, Jeliot can
help in teaching and learning of the basic concepts of programming as well as in comprehension and debugging of programs.

**Interactivity**

In Jeliot 3, users can write their own programs and execute them as they see appropriate. *The automatic generation of complete animations from source code makes interactive creation of new visualizations a trivial task!* There are also ready-made examples that can be used by teachers and students. The animation can be controlled with VCR-like buttons for pausing, stepping and playing. In addition to this, it can be also rewound to the beginning and its speed can be controlled. Users are allowed to provide input to the currently visualized program. Users can also view the previous steps of the animation from the history view or select to only see the call tree of the currently viewed program. Currently, Jeliot also provides automatic question generation capability that can be used to help students to concentrate on certain concepts. In this mode, Jeliot will ask questions related to the program in appropriate moments. We are also working on adaptation of the visualization to the current learner’s style of learning.

**Cognition/Conceptual Change**

As discussed in the Section Empirical Evaluations, Jeliot has been found to improve students’ learning and to help them to build a viable mental model that enables them to solve problems and learn new concepts. In a lecture, it has been observed that Jeliot supports “what-if” questions and thus encourages deep thinking. Furthermore, automatic question generation provides students with questions that direct their learning of new concepts.

**Content**

Jeliot visualizes Java programs completely so that all the steps of the execution are displayed. In addition, a history of the animation and a call tree can be displayed. The visualization pane is divided into four areas that contain semantically different parts of the visualization and the program. In this way, the learner can concentrate on those parts of the visualization that are meaningful to him or her, and can understand the relationships between constructs in the programming language and their visualizations.

**Multimedia use**

The principles of multimedia learning have been applied to design the contents of Jeliot’s animation display so that it would be pedagogically appropriate for the learner. The visualization of Jeliot links the two representations of the program—the program source code and the animation—by highlighting the currently animated line in the source code. Furthermore, all animation views (the full animation, the call tree and the history view) are linked to the source code and its execution. Currently, we are also looking for ways to combine automatic audio narration with the animation.

**Instructional Use/Adaptability**

As stated in Sections Jeliot 3 in Programming Education, Jeliot can be used by both teachers and learners of programming. Jeliot’s graphical user interface is intuitive and simple so that most of the users can work with it without additional help. Jeliot allows the use of shortcuts and accelerators in order to support teachers during lectures and experienced users. It provides automatic questions to direct the learners for certain concepts and these can be used to assess learning as well.

Section Adaptations and Applications of Jeliot 3 explains how teachers have utilized and adapted Jeliot in various institutions. This shows that Jeliot 3 is flexible and can be utilized in various educational situations. We also provide a Jeliot website (http://cs.joensuu.fi/jeliot/) which contains downloadable versions of Jeliot, documentation and a discussion forum for the developers and users of Jeliot.
Summary

Jeliot 3 is a pedagogical tool that has been developed and improved over the years based upon empirical pedagogical research and feedback from instructors and students. It has been shown to improve learning and attention of novice students in the classroom experiment, and to facilitate teaching of instructors. Jeliot is an international project, whose users and developers come from many countries.

The following table summarizes the main information about Jeliot 3.

<table>
<thead>
<tr>
<th>Description</th>
<th>Jeliot is a program visualization tool aimed to support teaching and learning in programming courses.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of material</td>
<td>Computer software</td>
</tr>
<tr>
<td>Target audience</td>
<td>Students of computer science and others (in science and engineering) who study introductory programming (undergraduate)</td>
</tr>
<tr>
<td>WWW address</td>
<td><a href="http://cs.joensuu.fi/~jeliot/">http://cs.joensuu.fi/~jeliot/</a></td>
</tr>
<tr>
<td>Distribution</td>
<td>Free, open-source under the GNU General Public License</td>
</tr>
<tr>
<td>Platform</td>
<td>Platform independent</td>
</tr>
<tr>
<td>Language</td>
<td>Multilingual, depending on user local settings</td>
</tr>
</tbody>
</table>
| Institutions (not a comprehensive list) | University of Joensuu, Finland  
Helsinki University of Technology, Finland  
University of Karlsruhe, Germany  
Academic College Beit Berl, Israel  
Weizmann Institute of Science, Israel  
Thammasat University, Thailand  
Tumaini University, Tanzania  
University of Warwick, UK  
University of Texas at Austin, US  
University of Pittsburgh, US  
John Carroll University, US |

References


Appendix A

DynamicJava:

DynamicJava - Copyright © 1999 Dyade

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL DYADE BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of Dyade shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from Dyade.

jEdit Syntax Highlighting Package:

The jEdit 2.2.1 syntax highlighting package contains code that is Copyright 1998-1999 Slava Pestov, Artur Biesiadowski, Clancy Malcolm, Jonathan Revusky, Juha Lindfors and Mike Dillon.

You may use and modify this package for any purpose. Redistribution is permitted, in both source and binary form, provided that this notice remains intact in all source distributions of this package.

KunstOffLookAndFeel is distributed under LGPL v2.1.

AVInteraction is distributed under Creative Commons, Attribution 3.0 Unported (http://creativecommons.org/licenses/by/3.0/)
Appendix B

Comments from users collected using the TUP evaluation service (http://cs.joensuu.fi/~tup) (verbatim, not edited or correct for grammar).

Question: What do you think about the educational capabilities of the tool?

- Jeliot 3 can really help to teach programming to novices. OO concepts are nicely animated and help to create a suitable mental model of the Object Oriented paradigm.

- Jeliot 3 could be used effectively for algorithm animation. However, learners need to be motivated to use it outside the class.

- i have answer with the other question ... i am sceptic for the real interest by this software !! but why not !!! we will see with the future ...

- it has a lot of potential. Surely a novice programmer's will gain knowledge by using this tool.

- The tool is quite useful for constructive learning process. The step by step animation makes the student to understand easily object and its class behaviour.

- It is a very useful tool for people who is starting to learn programming in Java; it can clear doubts about basic concepts.

- Naturally there are some limitations (for example arrays with components of reference types are not implemented) and incompatibilities, but I think that basic OOP concepts and basic programming structures

- I think that we can use this system for studying concept of object oriented programming, basic programming structure. Jeliot has simple structure, it is easier to use Jeliot at the beginning studying

- very very efficient

- Learning to program is not easy -> so this could make it easier for some students. I prefer myself lego programming for novices...

- It is rather good basis for the java learning platform. Especially if the users are novice level students...
Appendix C

List of Publications and Presentation about Jeliot 3


