A Survey of Research on the Jeliot Program Animation System

Ronit Ben-Bassat Levy
Weizmann Institute of Science
ronit.ben-bassat@weizmann.ac.il

Mordechai Ben-Ari
Weizmann Institute of Science
mordechai.ben-ari@weizmann.ac.il

Abstract

The difficulties of learning to program are well known. One solution that has been suggested is the use of program animation so that students are presented with a concrete representation of the dynamic execution of a program. The Jeliot program animation system was designed especially for novices learning introductory programming. This article describes the extensive pedagogical research that has been done on the use of Jeliot in various contexts.

Keywords: animation system, phenomenography, theory of planned behavior, pedagogy.

Introduction

Introductory programming is very difficulty to learn and animation systems – software tools that provide a dynamic view of an execution of a program – were developed to deal with these difficulties. The Jeliot program animation system is based upon the concept of "learning-by-doing: a student should have an animation tool which helps him to easily construct a visual representation of a program" (Ben-Ari, Myller, Sutinen, & Tarhio, 2002, p. 31).

Jeliot pioneered the use of *self-animation*, where the animations are produced *automatically* from the source code of the programs without any effort whatsoever on the part of the teacher or the student. This capability is very important for the students as well as teachers: it is well known that educational technology like visualization will have a difficult time being accepted if much effort needs to be invested in order to use it. With Jeliot, this effort is negligible.

Figure 1 shows the Jeliot display: The left panel displays the source code that is being animated, while the animation takes place in the right panel. The VCR-like buttons below the source code control the execution of the program animation.

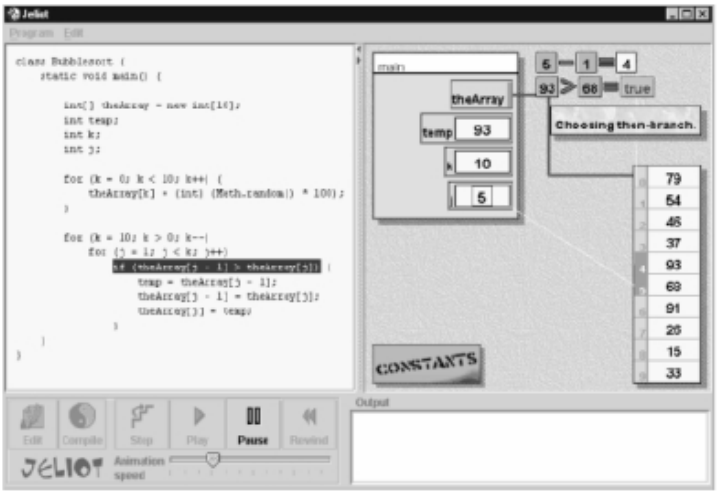

Figure 1. A screenshot of Jeliot 2000

Proceedings of the Chais conference on instructional technologies research 2009: Learning in the technological era
Y. Eshet-Alkalai, A. Caspi, S. Eden, N. Geri, Y. Yair (Eds.), Raanana: The Open University of Israel

Jeliot—together with its documentation, research publications, and learning materials—can be downloaded for free from http://www.cs.joensuu.fi/jeliot/.

The development of the software has been accompanied by comprehensive pedagogical research. This includes investigations of: Does the use of Jeliot improve learning? How is attention in the classroom affected by the use of Jeliot? What factors cause teachers to decide to use Jeliot or not to use it? What is the effect of the use of Jeliot in a specific context like collaborative learning? This paper presents a survey of these and other research projects.

Previous work

A survey of work on software visualization can be found in Stasko, Domingue, Brown, and Price (1998). This work is intimately connected with more general work on multimedia learning: Mayer (1997) consistently found that visualizations need to be accompanied by explanations to be effective. Similarly, Petre and Green (1993) found that graphics are not self-explanatory; therefore, it does not follow that the use of these tools in a classroom will automatically improve learning.

The first research specifically directed at investigating learning of programming using visualizations was done by John Stasko; see for example, Stasko, Badre, and Lewis (1993). They used algorithm animation to teach a complicated algorithm to graduate students in computer science. The results were disappointing: in a post-test, the group that used animation did not perform better than those that did not. Later work by his group showed by if the students are asked to predict the outcome, learning will be improved, although the improvement was not statistically significant.

Hundhausen, Douglas and Stasko (2002) suggest that it is not enough to have a good visualization tool; its effectiveness depends on how it is used. Hundhausen defined the term system roulette for the situation where visualization systems are orphans (unused systems), because their developers did not deal with the real needs of their potential users (Hundhausen, 1997, p. 2).

An ITiCSE Working Group report summarized the advantages and the disadvantages of using visualization tools (Naps, Rößling, Almstrum et al, 2002). The advantages include: they help students understand topics in CS; they make teaching more enjoyable; they facilitate discussion in class. The disadvantages include: the time it takes to install software tools; the effort required to introduce the tools into the teaching practice; the lack of information about the benefits for learning.

Jeliot improves learning

During the years 1998-2001, we conducted a study that evaluated the effect that Jeliot has on students when it is integrated into a year-long course (Ben-Bassat Levy, Ben-Ari, Uronen, 2003). The study was carried out on 10th grade high-school students studying the introductory course Fundamental of Computer Science. This study was the first to investigate a pedagogical visualization system over a long period in an educational setting, as opposed to a short laboratory experiment.

There were two classes, one of which received an extra hour of instruction using Jeliot, while the control group received an extra hour of instruction using the same content but without Jeliot.

An in-class paper-and-pencil assignment was used as a pretest; the students in both classes were asked to predict the outcome of the execution of a program fragment. A similar in-class paper-and-pencil assignment was given as a post-test. We also used interviews in order to get students to verbalize their thoughts so that we could get a better and a deeper view of the effect of Jeliot on the students.

We found that there was a difference between the grades of the group who used Jeliot and the control group, and that the difference increased the more we used the animation in class. The Jeliot group showed better improvement than the control group. The interviews showed that the average students from the Jeliot group gave the correct answers and in a shorter time than the strong students from both groups! The students were able to do this because they had adopted a vocabulary based upon the graphics displayed by Jeliot and their explanations used these graphics representations. The explanations of the control group were much less effective because they lacked this vocabulary.

We found that the use of Jeliot over a long period improves both understanding and transfer of knowledge. Most importantly, it primarily helps the average students who might otherwise get a low grade, by enabling them to build concrete models of how the computer works inside.

Jeliot improves attention

Earlier (laboratory) studies of software visualization were not able to show consistent improvement of learning, but they did claim (anecdotally) that the use of visualization systems improved affective characteristics like motivation. Ebel & Ben-Ari (2006) attempted to prove this experimentally. They considered *attention* as a prerequisite for effective learning and investigated if the use of Jeliot could improve attention. The study used Jeliot for classroom demonstrations and was carried out in a special education class of students with normal cognitive abilities but with disabilities such as extreme hyperactivity.

The authors video-taped and recorded four lessons and analyzed them, looking for relations between inattentive behavior and the type of activity in class. They measured the time of an activity and calculated the average number of disruptive episodes per unit time.

Ebel and Ben-Ari found that there were *no* disruptive episodes whatsoever when the teacher carried out demonstrations with Jeliot, as opposed to about one such episode every two minutes during other activities. This held through all the lessons analyzed so that it cannot be attributed to novelty alone. It is not to be expected that the continuous use of Jeliot would prevent disruptive behavior, but the results do justify using such educational software occasionally even before it has been shown to improve learning directly.

Jeliot and teachers

In spite of the encouraging results of the studies evaluating the effect that animation systems have on students, the use of them is not widespread (Hundhausen, Douglas, & Stasko (2002)). Since we consider the teachers as the connecting link between animation systems and students, we decided to study the teachers and their reaction to animation systems (Ben-Bassat Levy, Ben-Ari, 2007). We were interested in the ways that high-school teachers of CS experience the use of an animation system as a pedagogical tool, as well as in their attitudes toward the use of Jeliot as a pedagogical tool.

Phenomenography (Marton, 1986, Marton & Booth, 1997) is a qualitative methodology that was used in order to discover the different ways that teachers experience the use of an animation as a pedagogical tool. The preliminary results showed that the decision whether to use the animation system or to reject it depends upon personal attitudes that teachers possess toward employing animation systems in their practice. These results led us to devise a methodology for our investigation that combined phenomenography and the *theory of planned behavior (TPB)* (Ajzen, 2002) from social psychology. TPB was used in order to discover and understand the attitudes that teachers possess towards the use of an animation system. Using this combined methodology helped us predict the behavior of a teacher in this context and, in the future, may help design interventions that will change the teachers' behavior.

The result of a phenomenographic study is a hierarchically ordered set of categories that are the *ways of experiencing* the phenomenon under observation. We focused on the teachers' ways of experiencing the Jeliot animation system and found four ways of experiencing the animation system as a pedagogical tool (Table 1).

We had expected that there would be two main categories, one for teachers who tend to accept the use of Jeliot and another for those who reject. While the By-the-book category is not surprising, the Dissonant category was, because we had expected that every teacher would experience the use of Jeliot either positively or negatively, but not a "combination" of positive enthusiasm together with negative reluctance.

Table 1. Ways of experiencing the use of animation system

Category	Way of experience
Internalization	Jeliot is experienced as a useful tool consistent with the teacher's pedagogical style.
By-the-book	Jeliot is experienced as a possibly useful tool, but it may not fit with the teacher's pedagogical style.
Repudiating	Jeliot is experienced as an externally imposed tool of limited usefulness for teaching.
Dissonant	Jeliot is experienced in a conflicting manner: on the one hand with enthusiasm and on the other with a reluctance to actually use it.

The above results were the basis of a questionnaire that we built in order to study the attitudes that teachers possess towards the use of an animation system (Ben-Bassat Levy, Ben-Ari, 2008). The TPB model predicts a behavior according to three main predictors: *attitudes*, *norms* and *perceived behavior control (PBC)*. PBC is the extent a person feels that he/she can control the behavior; this is divided into two aspects: (1) how much control a person believes he/she has on the behavior (control beliefs), and (2) how confident a person feels about his/her ability to behave in a certain way (the influence of the control belief on the person). The most striking result of the TPB investigation was the low scores of indirect perceived behavior control among most teachers.

This result is very surprising since it measures the degree of to which the teacher feels confident and in control when using an animation system. We did not expect this low score given that *CS educators* routinely use software tools such as programming environments in their classrooms, and their CS training certainly requires the intensive use of software tools. We believe that the low level of indirect perceived behavior control is caused, paradoxically, by the strengths associated with pedagogical software: since the software is successful at explaining concepts and facilitating learning, it can appear to be threatening to the *centrality* of the educator in the classroom. CS teachers would be reluctant to admit that they are threatened by computing

technology, since the normative beliefs of people in general are that CS teachers are experts in using the technology.

Our results can help developers of pedagogical software, especially when writing tutorials and teaching training courses, by indicating the type of reactions that educators are likely to have and the ways to address their anxieties. Developers must also devote attention to the integration of the systems into the teachers' practice. Another important conclusion is that if problems of perceived behavior control arise with CS educators, a fortiori they will arise with educators of other subjects. We believe that research on CS educators and tool developers can contribute to the acceptance of educational technology by the further study of this phenomenon and by devising ways to overcome the difficulties.

Jeliot and collaboration

Myller, Bednarik, Ben-Ari and Sutinen (2008) investigated how the use of Jeliot affects collaboration among students. The researchers claim that "increasing the level of engagement between learners and the visualization tool results in a higher positive impact of the visualization on the collaboration process." (p. 1). The authors use the *engagement taxonomy* developed by Naps et al. (2002). This taxonomy describes six levels of engagement between the user and the visualization tool. The levels are: (1) No viewing; (2) Viewing; (3) Responding; (4) Changing; (5) Constructing; (6) Presenting. The authors extended the taxonomy by adding levels: (2.1) Controlled viewing – the visualization is viewed and the students control the visualization, for example by selecting objects to inspect or by changing the speed of the animation; (2.2) Entering input – the student enters input to a program or parameters to a method before or during their execution; (4.1) Modifying – modification of the visualization is carried out before it is viewed, for example, by changing source code or an input set; (7) Reviewing – visualizations are viewed for the purpose of providing comments, suggestions and feedback on the visualization itself or on the program or algorithm.

The researchers used a causal-comparative study in order to understand how the use of visualization tools at different levels of engagement and the collaboration process during the learning of programming are correlated. The research was carried out during a 40-hour introduction course on programming at the University of Joensuu during the year 2005. Each student participated in a two-hour laboratory session where he solved exercises with the help of an instructor. The laboratory sessions were videotaped and analyzed.

The engagement level on which the students select to work and the interaction between the students were positively correlated. The correlation is measured with chi-squared-test on nominal scale. The amount of discussion is increased significantly if the tool supports the level of Entering Input. The opposite result is achieved when the tool supports engagement on the level of Viewing, that is, when the students simply viewed the visualization and were actively engaged, the level of collaboration was less. The significance values for activities are: ($\chi 2$ (8) = 48.5, $p < .01$). The significance values for the discussion contents are: ($\chi 2$ (18) = 85.1, $p < .01$). The extended taxonomy can be used to suggest new features for Jeliot; for example, Myller (2007) added the capability of automatic question generation in order to support the level of Responding.

Jeliot as a conflictive animations generator

Conflictive animations are designed to facilitate reflection by students on their mental models. Students should view animations critically, looking for possible errors. Moreno, Sutinen,

Bednarik and Myller (2007) defined conflictive animations as those that deliberately do *not* produce the correct dynamic view of the code. The animation can start showing a correct view, then show a wrong one and even can come back to the correct display again. Students are supposed to identify when the animation is not correct. The errors can vary from simple errors of evaluation to complex ones such as dealing with advanced object-oriented concepts like inheritance. The idea is to support the learning process by confronting students with their own misconceptions. Teachers can use conflictive animations as pedagogical tools in order to discover and address problems that they may be aware of. Moreno and Myller (2008) are implementing conflictive activities in Jeliot, but this research is currently in progress.

Conclusions

This article described the pedagogical research on the use of the Jeliot program animation system in a variety of contexts. We believe that extensive research on a single educational software tool is unusual. The long-term sequence of research projects is evidence for the vitality of Jeliot and it also demonstrates that the developers are very much in contact with the users of the software. The Jeliot family has existed now more than 20 years and during this time the number of users has increased significantly, not just in Finland and Israel, but all over the world. During the past six months 1600 people got to downloads Jeliot, 8000 people went to the home page where they could also download Jeliot, 1500 people went to download Jeliot for Bluej (another animation system). Table 2 presents the top ten countries with the highest number of visitors to the Jeliot homepage.

Table 2. The top ten countries

	Country	Number of different visitors
1	United States	1661
2	Mexico	754
3	Germany	743
4	Israel	720
5	United Kingdom	611
6	Sweden	479
7	Brazil	469
8	Finland	419
9	Spain	417
10	Canada	261
Total		6534

We think that Jeliot can be an example for developers of educational tools, because it shows how to develop a tool and to maintain it over many years. This article has shown that continuous pedagogical research is an essential aspect of the longevity of the educational tool: not only are the research results used to improve the tool better, but research facilitates the spread of the use of the tool from an academic environment to practice in schools and leads to successfully integration of the tool into the practice.

References

Ajzen, I. (2002). Perceived behavioral control, self-efficacy, locus of control, and the theory of planned behavior. *Journal of Applied Social Psychology, 32*(4), 665-683.

Ben-Ari, M., Myller, M., Sutinen, E., & Tarhio, J. (2002). Perspectives on program animation with Jeliot. *S. Diehl (Ed.): Software Visualization, LNCS 2269*, (pp. 31–45). Berlin Heidelberg: Springer-Verlag

Ben-Bassat Levy, R., & Ben-Ari, M. (2007). We work so hard and they don't use it: Acceptance of software tools by teachers. *SIGCSE Bulletin 39*(3), 246-250.

Ben-Bassat Levy, R., & Ben-Ari, M. (2008). Perceived behavior control and its influence on the adoption of software tools. *ITiCSE 2008: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, 169-173.

Ben-Bassat Levy, R., Ben-Ari, M. & Uronen, P.A. (2003). The Jeliot 2000 program animation system. *Computers & Education, 40(1),* 1-15.

Ebel, G., & Ben-Ari, M. (2006). Affective effects of program visualization. Second International Computing Education Research Workshop. Canterbury, UK.

Hundhausen, C. D. (1997). A meta-study of software visualization effectiveness. Unpublished comprehensive exam paper, Department of Computer and Information Science, University of Oregon, Eugene. Retrieved September, 29, 2008, from http://eecs.wsu.edu/~veupl/pub/MetaStudy.pdf

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing, 13*(3), 259-290.

Marton, F. (1986). Phenomenography - A research approach to investigating different understandings of reality. *Journal of Thought, 21*, 28-49.

Marton, F., & Booth, S.A. (1997). The idea of phenomenogaphy. In F. Marton, & S.A. Booth, *Learning and Awareness*. , Mahwah, NJ: Lawrence Erlbaum associates, publishers.

Mayer, R. E. (1997). Multimedia learning: are we asking the night questions? *Educational Psychologist, 32*(1), 1–19.

Moreno, A., & Myller, N. (2008). Automatic generation of conflictive animations. (2008). Submitted to the Koli Calling Conference.

Moreno, A. Sutinen, E., Bednarik, R., & Myller, N. (2007). Conflictive animations as engaging learning tools. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), 88,* 203-206.

Myller, N. (2007). Automatic generation of prediction questions during program visualization. *Electronic Notes Theoretical Computer Science, 178*, 43–49.

Myller, N., Bednarik, R., Ben-Ari, M., & Sutinene, E. (2008). Extending the Engagement Taxonomy: Software Visualization and Collaborative Learning. (Submitted).

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., and Velázquez-Iturbide, J. Á. (2002). Exploring the role of visualization and engagement in computer science education. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education* (pp. 131–152). New York, NY: ACM Press.

Petre, M., & Green, T. R. G. (1993). Learning to read graphics: some evidence that 'seeing' an information display is an acquired skill. *Journal of Visual Languages and Computing, 4*, 55–70.

Stasko, J. T. , Badre, A., & Lewis, C. (1993). Do algorithm animations assist learning: an empirical study and analysis. In *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*, 61–66. Amsterdam, The Netherlands.

Stasko, J. T., Domingue, J., Brown, H. M., & Price, B. A. (Eds.) (1998). *Software Visualization: Programming as a Multimedia Experience*. Cambridge, MA: MIT Press.