Animation and Automatic Evaluation to Support Programming Teaching

Paula Correia Tavares[1], Pedro Rangel Henriques[2] and Elsa Ferreira Gomes[1,3]
[1]*Departamento de Informática,Instituto Superior de Engenharia do Porto, Rua Bernadino de Almeida, Porto, Portugal*
[2]*Departamento de Informática, Universidade do Minho, Campus de Gualtar, Braga, Portugal*
[3]*GECAD- Knowledge Engineering Decision Support, Instituto Superior de Engenharia do Porto, Porto, Portugal*

1 RESEARCH PROBLEM

Learning programming is a complex task that poses significant challenges.

Students face different kinds of difficulties at complex levels that traditional teaching/learning methods are nor able to cope with resulting in a high rate of failures.

Two very distinguished concepts that are incredibly misunderstood by the students are: learning programming and learning the syntax of a programming language.

Programming is, first of all, to outline strategies in order to solve problems, regardless of the language used. In fact, this task involves several steps that go from the understanding of the work proposal to the test of the program, passing through the algorithm development and codification.

Although we believe the codification is not the main difficulty, previous studies had conclude that (Gomes, 2010), the adopted programming paradigm and the language used have a huge impact in the learning process and consequently in the task performance.

Learning how to program is an iterative process. The solution to a complex problem can be obtained in a successive steps solving simpler problems and enriching the previous solutions.

It is only possible to learn how to program by programming. Following this approach, students can understand and acquire problem-solving strategies. Therefore, it is obvious that an active behavior by the student, instead of a passive behavior, leads to an improvement in his ability to solve the proposed problems.

However, teachers realize that in most cases, when students are requested to solve a particular problem, they are not able to start the task, neither on paper nor in the computer. Even when they break this initial inertia, they often become discouraged and give up easily as soon as they face the first hurdle. Under these circumstances, the students' main difficulties are:

- Understanding the problem due to their unfamiliarity with the subject or due to the inability to understand the problem statement;
- Lacking logic thinking to write the correct algorithm to solve a given problems;
- Learning the language syntax and semantics.

The above difficulties identified in learning programming led to the creation of languages and development environments that smooth the designing of algorithms, or the writing and the analyzing of programs.

New teaching/learning approaches must be devised. The resort to computer supported education specially tailored to programming activities shall be explored

For this reason, several authors have researched the pedagogical effectiveness of program visualization and animation, and developed some tools. Animation can help students on the analysis and understanding of given programs, and can also guide on the development of new ones.

Besides that, it is very important to give students the opportunity to practice solving programming exercises by themselves. Receiving feedback is essential for knowledge acquisition. New tools arose (especially in the area of programming contests) to allow for the submission of solutions (programs developed by the students) to the problem statements presented by the teacher and to assess them, returning immediately information about the submitted answer. These tools can be incorporated into teaching activities, allowing students to test their work getting immediate feedback. Automatic evaluation systems significantly improve students performance.

In this article are shown two approaches to the teaching of programming, animation and automatic assessment are reviewed, and a new pedagogical practice resulting from combination of both is

Correia Tavares P., Rangel Henriques P. and Ferreira Gomes E..
Animation and Automatic Evaluation to Support Programming Teaching.

proposed.

2 OUTLINE OF OBJECTIVES

The goal of the Ph.D. work is summarized below:

- To provide means for an easier understanding of programs
- To make students increase their ability to practice regularly programming, since the first day, obtaining immediate feedback.

We believe that this can increase their motivation, engagement and consequently improving the academic success.

3 STATE OF THE ART

The animation tools provide a visual metaphor that significantly help the understanding of complex concepts. Therefore, these tools allow the students to find the dynamics of hard to understand but extremely important processes. In this way, the student is stimulated to progress in his activity (Hundhausen et al., 2002).

In this sector, beyond discussing the animation strategy, we will also discuss the importance of feedback in the teaching-learning process. In this perspective, we will analyze the impact of tools for automatic evaluation of programs when integrated in teaching process.

3.1 Animation

Several authors have been concerned about the use of graphic interfaces that enable a way of communication between the user and the computer not restricted to a textual form (Hansen et al., 1999) (Stasko and Kehoe, 1996) (Hundhausen and Douglas, 2000).

Aiming to enhance the learning process, many educators and computer scientists have been dedicated to the construction of animation, visualization and simulation systems (computational programs). The great motivation is to appeal to the human visual system potential.

The key question is how to apply these methods in order to help students to deal with complex concepts.

Many researches (Brown and Sedgewick, 1985) (Korhonen, 2003), (Kerren and Stasko, 2002) have been working to identify the rules that should be followed while designing and creating visualizations and animations effective for teaching. As it computational programs con be hard to understand when presented in a textual format; however it is expected that a better comprehension could be achieved with an animated graphic format (Pereira, 2002).

An animation is a natural approach of expressing behaviors. Particularly, the animation of an algorithm is a dynamic visualization of their main abstraction. So, its importance lies on the ability to describe the algorithm logical essence.

When inspecting the control and data of a program to understand its behavior, we have, at the first time, two big choices: do it during code execution (debugging), or simulate the execution in another environment (Pereira, 2002). For teaching purposes we believe that the second approach is clearly the most interesting. Thus, means animation system is a tool that allows build animations kind of interactively. In this context, there are several tools that try to introduce basic programming concepts through a familiar and pleasant environment in order to help students learn to program. The following list shows some of these most well known tools: BALSA (Saraiya, 2002), TANGO (Hughes and Buckley, 2004), Jeliot (Silva et al., 2009), Alma (Pereira and Henriques, 1999), SICAS (Mendes et al, 2004), OOP-Anim (Santos et al., 2010) (Esteves and Mendes, 2003), VILLE (Rajala et al., 2007), JIVE (Lessa et al., 2011). All of these tools are concerned with visualization or animation of programs written in traditional programming languages (C, Java, etc.). Besides these, there are also some programming environments less conventional that allow to edit, run, and view programs developed in visual languages designed for the purpose of facilitating the teaching of programming: AMBAP (Xavier et al., 2004), Alice2 (see http://www.alice.org/), Scratch (see http://scratch.mit.edu), etc. As referenced above, several authors became interested in this problem. They develop less complex and appealing environments than the professional environments, with important features for novice programmers. These systems have long been used to allow understanding important aspects in programming through animation pseudo-code, flowcharts, or programs written in specific or general programming languages such as Pascal, C, Java, and others. The most interesting and appealing are those that allow students to introduce and simulate their own algorithms and programs. The animation based at simulation allows introduce the dynamic visualization of the program and help student

comprehension at his own pace. To understand this idea, Figure 1 and Figure 2 show illustrative screenshots of Jeliot System.

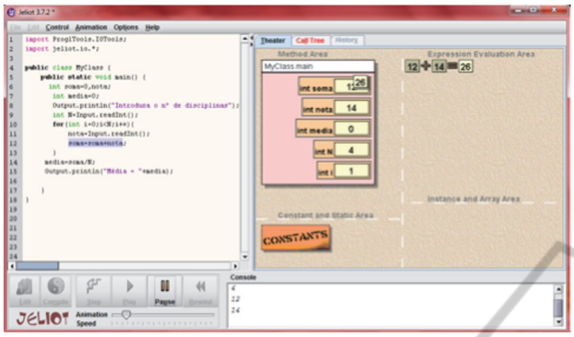

Figure 1: Jeliot interface - Java Class to calculate an average.

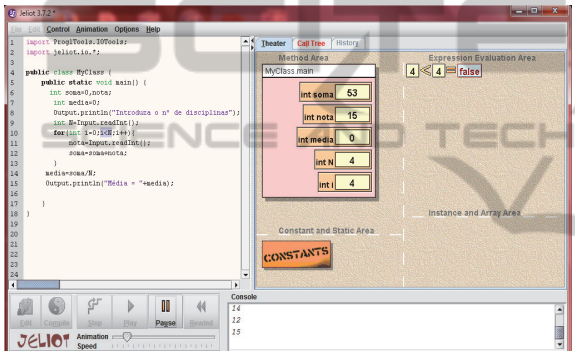

Figure 2: Jeliot interface - Java Class to calculate an average.

The figures show two moments in the animation of a class to calculate an average. Figure 1 illustrates the steps when a new value is being added to the sum (execution of an assign statement); Figure 2 exhibits another step when cycle stop condition is being evaluated. The sequence of these images, corresponding to the execution of each statement produces the animation of the class as desired.

3.2 Automatic Evaluation

It´s very important to give students the opportunity to practice and solve programming exercises by themselves.

The maximum effectiveness of this approach requires the teacher's ability to rate and review each resolution. Instant feedback is very important for the acquisition of knowledge. Independently of the particular learning strategy, it motivates students.

However, in large classes and with few lecture hours, this approach is impractical. Individual feedback may consume too much teacher´s time with risk that students don't benefit from it in due time (Queirós and Leal, 2015).

To solve this problem, there are a large number of online submission systems that support the automatic evaluation for the programming problems (Queirós and Leal, 2012).

Different studies show that these systems enable students to autonomously develop programming skills and significantly improve their performance (Verdú et al., 2011).

Since not all students are motivated in the same way, it is important to provide different learning environments: individual (traditional), collaborative (group work), competitive (competitions), among others. Taking advantage of the human spirit of competition, competitive learning increases commitment and leads to a greater involvement of students in practical activities.

Competitions with automatic evaluation are becoming important for the practice of programming. However, differences in motivation and feelings between losers and winners can exist. These negative effects can be minored through different practices, such focusing on learning and fun rather than in a competition.

New tools have emerged to facilitate and enable their use in teaching activities, allowing students to incorporate tests in its work. These tools increase the level of satisfaction and motivation of students. According to teachers and students, feedback should be as quickly and detailed as possible. These tools do not replace the teacher, but provide help and increase the value of time in the classroom. The proposed problems have different levels of difficulty and feedback is useful for an increased understanding of programming for the students. Teachers should be able to select the problems they intend to present to the students according to their level of difficulty (Verdú et al., 2011). With suitable software tools, correctness of the program can be determined with an indication of the quality according to a set of metrics. It´s not easy to find a unique approach to the problem of assessment of programming works. Different teachers can adopt different strategies, depending on their specific goals and objectives of the course, especially of his own style and preferences (Joy et al., 2005). So, the problem is related to the resources required to manage the evaluation of practical exercises. The students receive accurate feedback at the right time to the benefit of their learning. Most of the tools available for this purpose include the delivery of works and automatic evaluation. This is adequate for an initial learning where knowledge and

understanding are being tested. The final goal is to provide new learning strategies to motivate students and make programming more accessible and an attractive challenge. Boss (Heng et al., 2005), Mooshak (Leal and Silva, 2008) and EduJudge (Verdú et al., 2011) are examples of such tools. Their main goal, besides testing the students' answers against a set of data input and give a rating, is to allow the evaluator motivate students through precise and rapid feedback.

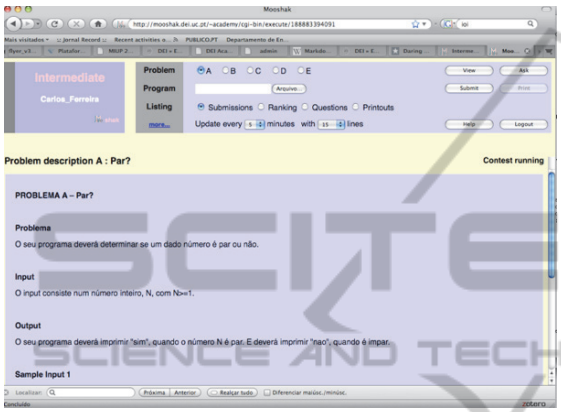

Figure 3: Mooshak System Interface.

Figure 3 shows a Mooshak screen illustrating the simplicity and ease of its interface. In addition to the statement of the problem to solve, shown in the center window, it´s possible to see the different options on the top window.

4 METHODOLOGY

To achieve this goal we intend to follow a method based on the steps:

- Bibliographic research (theoretical and technological);
- Reasoning about the research evidences and writing;
- Proposal of a new approaches/strategies;
- Experimentation in the classroom.

According to the feedback obtained in the last phase, we will iterate over the four steps above.

5 EXPECTED OUTCOME

In order to increase the motivation and self-confidence of students of Introductory Programming courses, we presented in this paper a proposal for a doctoral thesis. Our goal is to clearly identify the difficulties that actually arise in this process and suggest different approaches, supported on computer resources to minimize these difficulties.

So, the expected outcome of this Ph.D. work is a set of strategies to improve the success of programming courses. These strategies will be based on the use of computers and computer applications that can increase the involvement of students in comprehension and development tasks. The proposal will combine several tools originally created with different objectives.

6 STAGE OF THE RESEARCH

The first year of this Doctoral Program, PDInf, was devoted to literature review in order to write the state-of-the-art chapters.

At present the first proposal design is undergoing. It is based on the principal that students should analyse a good solution (well-solved examples) before starting their own resolution.

To be more concrete we introduce in the sequel a summary of this proposal.

We suggest that for each topic to teach, the teacher prepare two groups of similar exercises.

For the first group of exercises, the teacher discusses the problem statement, outlines the resolution (an algorithm) and presents the program that solves it. Then the student can make its animation and so analyse / understand the solution.

For the second set of exercises, after discussing the proposal, the teacher asks the students to solve and to test the solution produced through an automatic evaluation system.

On a third moment, the teacher discusses with students the feedback received from the evaluator.

This approach assumes that teachers select a powerful animation tool and easy to use; and choose an Automatic Evaluator System (AES) that, besides friendly return feedback and provide a diagnosis. It is also desirable that the Automatic Evaluator Systems comment on the code quality. Currently the idea is to do experiment Jeliot and Mooshak.

REFERENCES

Moore, R., Lopes, J., 1999. Paper templates. In *TEMPLATE'06, 1st International Conference on Template Production*. SCITEPRESS.

Smith, J., 1998. *The book*, The publishing company. London, 2nd edition.

Brown, M., Sedgewick, R., 1985. Techniques for

Algorithm Animation. IEEE SOFTWARE Vol2(1), pp 28-39.

Esteves, M., Mendes, A., 2003. OOP-Anim, a system to support learning of basic object oriented programming concepts. *International Conference on Computer Systems and Technologies - CompSysTech'2003*.

Gomes, A., 2010. Difficulties of learning computer programming. Contributions to the understanding and resolution, Dificuldades de aprendizagem de programação de computadores: contributos para a sua compreensão e resolução. Dissertação submetida à Universidade de Coimbra para obtenção do grau de "Doutor em Engenharia Informática".

Hansen, S., Narayanan, N., Schrimpsher, D., 1999. Helping Learners Visualize and Comprehend Algorithms. *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA'99)*.

Heng, P., Joy, M., Boyatt, R., Griffiths, N., 2005. Evaluation of the BOSS Online Submission and Assessment System.

Hughes, C., Buckley, J., 2004. Evaluating Algorithm Animation for Concurrent Systems: AComprehension-Based Approach. *16th Workshop of the Psychology of Programming Interest Group*. Carlow, Ireland, April. In E. Dunican & T.R.G. Green (Eds). Proc. PPIG 16. pp. 193-205.

Hundhausen, C., Douglas, S., Stasko, J., 2002. A Meta-Study of Algorithm Visualization Effectiveness. *Journal of VisualLanguages and Computing* 13, pp. 259-290.

Hundhausen, C., Douglas, S., 2000. Using Visualizations to Learn Algorithms: Should Students Construct Their Own, or View an Expert's?. *Proceedings 2000 IEEE International Symposium on Visual Languages IEEE Computer Society Press, Los Alamitos*.

Joy, M., Griffiths, N., Boyatt, R., 2005. The BOSS Online Submission and Assessment System. *Journal on Educational Resources in Computing, Volume 5 Issue 3, September.*

Kerren, A., Stasko, J., 2002. Chapter 1 "Algorithm Animation", Volume 2269, pp. 1-15.

Korhonen, A., 2003. Visual Algorithm Simulation. Dissertation for the degree of Doctor of Science in Technology. *At Helsinki University of Technology* (Espoo, Finland), November.

Leal, J., Silva, F., 2008. Using Mooshak as a Competitive Learning Tool.

Lessa, D., Czyz, J., Jayaraman, B., 2011. JIVE: A Pedagogic Tool for Visualizing the Execution of Java Programs. *SIGCSE 2011 Dallas, Texas, USA*.

Mendes, A., Gomes, A., Marcelino, M., 2004. Evaluation and evolution of a Environment Support for Programming Learning, Avaliação e Evolução de um Ambiente de Suporte à Aprendizagem da Programação. *VII Congresso Iberoamericano de Informática Educativa*.

Pereira, M., 2002. Systematization of Programs Animation, Sistematização da Animação de Programas. Dissertação submetida à Universidade do Minho para obtenção do grau de doutor em Informática, ramo Tecnologia da Programação, Dezembro.

Pereira, M., Henrriques, P., 1999. Made Algorithms Animation Systematic, Animação de Algoritmos tornada Sistemática. *In 1º Workshop Computação Gráfica, Multimédia e Ensino*. Leiria.

Queirós, R., Leal, J., 2012. Exercises Evaluation Systems - An Interoperability Survey. *In Proceedings of the 4th International Conference on Computer Supported Education (CSEDU), Volume 1, pp.83-90.* Porto.

Queirós, R., Leal, J., 2015. Ensemble: An Innovative Approach to Practice Computer Programming. In R. Queirós (Ed.), *Innovative Teaching Strategies and New Learning Paradigms in Computer Programming* (pp. 173-201). Hershey, PA: Information Science.

Rajala, T., Jussi, M., Erkki, L., Salakoski, K., 2007. VILLE – A Language-Independent Program Visualization Tool. *Seventh Baltic Sea Conference on Computing Education Research* (Koli Calling 2007), Koli National Park, Finland, November 15-18.

Santos, Á., Gomes, A., Mendes, A., 2010. Integrating New Technologies and Existing Tools to Promote Programming Learning. Algorithms , Vol3, pp.183-196.

Saraiya, P., 2002. Effective Features of Algorithm Visualizations. Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University for the degree of Master of Science In Computer Science, July.

Silva, M., D'Emery, R., Neto, J., Bezerra, Y., 2009. Programming structures: A Experiment with Jeliot,Estruturas de Programação: um Experimento com Jeliot. *IX Jornada de Ensino Pesquisa e Extensão (JEPEX) da UFRPE*.

Stasko, J., Kehoe, C.,1996. Using Animations to Learn about Algorithms: An Ethnographic Case Study. Technical Report GIT-GVU-96-20, September 1996.

Verdú, E., Regueras, L., Verdú, M., Leal, L., Castro, J., Queirós, Q., 2011. A distributed system for learning programming on-line. *Computers & Education* 58, pp. 1–10.

Xavier, G., Garcia, D., Silva, G., Santos, A., 2004. Factors that Influencing Introductory Learning Programming, Estudo dos Fatores que Influenciam a Aprendizagem Introdutória de Programação.

http://www.uefs.br/erbase2004/documentos/weibase.