

ProLEARN, A PLATFORM TO SUPPORT PROGRAMMING LEARNING

António José Mendes

Centro de Informática e Sistemas da Universidade de Coimbra – Portugal
toze@dei.uc.pt

INTRODUCTION

Programming learning is complex for many novice students at university level. The most important problem for many is their low ability to develop an algorithm that solves a given problem. The application of basic concepts or the design of simple algorithms can be difficult obstacles. These difficulties are felt independently of the programming language or paradigm used. Some authors identified reasons for these difficulties [1], such as:

- The need for good competences on problem solving;
- Students are used to courses that depend mostly on theoretical knowledge and memorization, but basic programming learning needs a more practical approach, based mostly on problem solving activities;
- Traditional teaching, often based on lectures and specific programming language syntaxes, often fails to motivate students to get involved in meaningful programming activities;
- Programs have a dynamic nature, but most learning materials have a static format which makes difficult to analyse program's dynamic behaviour;
- Students' learning needs are often very different within the same course. Different learning styles and previous experience difficult a common approach to the whole group. Group sizes often difficult individualized support to students.


Animation/visualization software systems have been used trying to take advantage of the potential of human visual system. Those systems are rooted in the conviction that programs can be better understood when represented graphically if compared with textual descriptions and representations. For example, BlueJ [2] is an integrated system including an object-oriented language and an object-oriented development environment. It uses UML- like class diagrams to present a graphical overview of a project structure. It allows the interactive creation of objects from any given class present in the project.

Another approach is to use micro worlds populated by representations of concrete entities (robots and turtles for example). The student can control those entities' movements and behavior through programming instructions [3].

Many other tools have been proposed, most of them having a limited scope, since they address only a particular type of algorithms and programs. They serve mainly demonstration purposes, as they can only animate a set of pre-defined programs, but not student's programs.

However, some studies have shown that the learning results obtained with the utilization of visualization tools are not as good as expected [4]. Three approaches have been proposed to make visualizations more helpful, engaging visualization, explanatory visualization and adaptive visualization [5]. Engaging visualization stresses the importance of student involvement in learning. This means that students must have an active role, instead of just seeing teacher prepared animations. Explanatory visualization proponents argue that many times students fail to understand what they are seeing. They defend that visual representations should be augmented with natural language explanations that can help student understanding. Adaptive visualization consists on adapting the level of detail of visual representations to the difficulties the underlying concepts pose to each student.

Our own work has followed the two first approaches mentioned. The tools we developed support engaging activities (they animate student developed programs) and have some explanatory features. We believe



that it is necessary to include adaptive functionalities, but also collaborative support. However, we also think that learning tools should be adapted to student needs and not only to what teachers think is the best approach. Knowledge about each student learning style is fundamental to define the best approaches in each case [6]. These are the main directions of our current project (ProLearn). To add new dimensions to our animation based simulation tools, namely collaborative features and more individualized support that can make them more effective for each particular student.

In the next sections we will describe briefly our group previous work and outline our current project (ProLearn)

ANIMATION BASED SIMULATION TOOLS

Animation based simulation has been proposed to reduce student's difficulties. It can make program's dynamics concrete and visual and support practical work at the student own learning rhythm. Animated views can help many students in three central learning activities: Understand programs; Evaluate existing programs; Develop new programs [7]. In our view, this last activity is the most important for learning, but also the most difficult for many students, as they often fail when asked to develop a new program, even if it is similar to one they just studied.

We believe learning is more effective when students assume an active role. So, it is important that the tools allow students to see how their solutions work and compare with how they thought they would work. This process should lead to error detection, correction and, hence, learning. These activities are very important in programming learning, since students can reach a higher competence and confidence level after being able to have the program running correctly. This is very important, since after a first wrong attempt many students just give up or try to find a teacher or a colleague that shows them a solution.

Our team has been involved in the development of algorithm and program simulation tools for several years. In particular we developed two tools, SICAS and OOP-Anim that are briefly described in the next sections.

SICAS

SICAS environment [8] was designed to support learning of basic procedural programming concepts, such as selection and repetition. It is language independent and oriented to the design and implementation of algorithms. Using SICAS students are encouraged to develop their capacities through problem solving. They can create solutions to problems, simulate them and see if they work as expected. The simulator can be used to detect and correct errors, but also to look for alternative ways to solve the problem and to compare them. SICAS supports a constructivist approach to learning, as each student assumes an active role, learning at his/her own pace and progressively constructing his/her own knowledge.

In SICAS, algorithm design is supported by an iconic environment where the student builds a flowchart that represents her/his solution. Algorithms developed with SICAS can include attributions, input/output, repetition and selection instructions. They can use numeric and string variables. Functions can also be defined and used. These elements can be introduced in a flowchart by clicking (in the toolbar icons) and pointing (in the design area). When that happens, a dialog box automatically opens asking the student to specify the element details (e.g. the condition in a selection). This option helps students to avoid common novice programmer syntax errors. Lines connecting components are automatically inserted, avoiding inconsistencies in the flowchart. The environment also includes the ability to delete, modify or copy any component.

Any algorithm created with SICAS is automatically translated to pseudo-code, C and JAVA code. These alternatives show that a well designed algorithm can be easily translated into several programming languages and that the most important factor in algorithm design is its conception, not the programming language in which it will be coded.

After creating an algorithm, it is possible to see its animated simulation. The student can control the speed at which the simulation progresses (step-by-step, slow or fast), pause the simulation and go back to repeat some part of the execution. This allows a deeper analysis of available data and/or a discussion with the teacher or other learners.

OOP-Anim

Object oriented programming (OOP) paradigm has grown significantly in the last years. Consequently many universities have adopted this paradigm in their basic programming courses. The development and popularization of OOP languages, like Java, has also contributed to this development. Although knowledge about basic concepts, such as selection and repetition, is still fundamental, it is also important to have tools that can support students when learning the basic concepts of the OOP paradigm.

OOP-Anim is a tool that can animate and simulate small Java programs [9]. Its objectives and pedagogical approach are the same presented for SICAS. It supports student learning, giving them a tool where they can create solutions to proposed problems, simulate their execution, detect errors, correct them and, hence, learn. In a first learning stage this environment can also be used to study examples of programming solutions given by the teacher. The idea here is to familiarize the student with OOP concepts, like class, object or inheritance.

PROLEARN PROJECT

As mentioned before, our previous work consisted essentially in the development of animation based simulation tools. However, our experience shows that although these tools are useful in many situations, in other cases they need to be complemented with other features and tools that can give a more complete support to students, especially those with deeper learning difficulties. Our current project tries to address this and some other aspects, as we describe briefly in the next sections.

Student guidance - ProGuide

One of the main problems with animation based simulation tools is that they depend on the capacity the student has to create a first solution to a given problem. Only after a first solution exists, even if wrong, students can use the tool to simulate and correct it. However, in early learning stages many students find difficult even to create a first solution attempt.


We are trying to address this problem through the development of a new tool, ProGuide, that works together with SICAS, interacting with students during basic algorithm development, guiding them when necessary. It is a dialogue-based tool that helps novice programmers to solve problems using text based communication. When students are creating an algorithm, ProGuide monitors their actions (or lack of action) and interacts with them, providing some guidance whenever necessary.

ProGuide has three main modules. The first manages natural language knowledge to be used in dialogues with students. The second module has information about partial steps and strategies to develop a particular algorithm. The third module follows student's actions, so that the tool can decide when to initiate a dialogue and the type of interaction that is more adequate depending on the resolution stage. All these modules work together to create a useful dialogue with students, trying to encourage them and providing hints so that they can reach a good solution to their particular task. We believe that this tool can help novice programming students, especially those that have more learning difficulties.

Adaptability

ProGuide is a first attempt to create tools that are more adaptable to individual student's needs. However, we believe it will be necessary to take further steps in this direction. In this area our work is still in the very beginning, although we have some ideas to explore.

Everybody can learn, but each one has his/her own learning style. Teachers use diverse strategies to cope with students' different learning styles. Some learners give more attention to facts, data and algorithms, while others prefer theories and models. Some learners prefer visual information, while others prefer verbal information. One of our goals is to include in ProLearn tools that can identify each student learning style and take that into consideration when proposing him/her problems, activities and representation languages that may lead to an improvement of their problem solving competencies.



ProLearn can also be more adaptable if it is able to identify the types of logical errors frequently made by each student and use that information to define the type and quantity of stimulus, support and information given to each of them. The collected information will allow the environment to know each student evolution in terms of the complexity of tasks successfully completed and the errors he/she more commonly makes. This information will also be useful to teachers as they will access information about their students, allowing them to take corrective measures when necessary.

Another planned approach consists on the utilization of case-based reasoning techniques to reuse examples of past solved problems. These experiences, or cases, which can represent programming solutions, will be used as a base to retrieve, adapt and present new solutions to students with learning problems. As the student resolves successfully a problem, a new case can be generated representing this learning situation. Cases can also represent failure situations, in which the student has failed to solve the problem. Other relevant knowledge can be added to these cases to explain why the student has failed. Teachers can also provide an important contribution by giving correct solutions to the system in the form of prototypical cases. These cases, along with all the others in the case base, can then be used to make ProLearn more proactive in its interactions with students, for example when they use a simulation tool to solve a problem or when a group is collaborating in some task. ProLearn will monitor student (or group) work and, if necessary, will act giving suggestions, trying to stimulate student work, giving similar problems partially solved, and so on.

Collaborative learning – College and PlanEdit

Computer Supported Collaborative Learning (CSCL) environments allow geographically distributed learners to work together and collaborate to accomplish a goal or task. They can offer important support to students in their programming learning activities. According to [10], collaboration in problem solving provides not only an appropriate activity, but also promotes reflection, a mechanism enhancing the learning process. Students that work in groups need to communicate, argue and give opinions to other group members, encouraging the kind of reflection that leads to learning.

Our work in CSCL area results from a collaboration with our colleagues from the Castilla-La Mancha University in Spain. They have developed two collaborative tools that we are integrating with our animation based simulation tools, as we believe that students can take advantage of using them together [11].

COLLEGE [12] is a Real Time Collaborative Programming system that allows geographically distributed programmers to work concurrently and collaboratively in the same programming task (edition, compilation and execution). This tool includes three shared workspaces that support edition/revision of the source code, compilation of the source code and program execution. The edition is carried out individually, using the Driver-Observer model which is characteristic of Pair Programming. The edition turn is asked by a student and authorized by the others at any moment. They also decide democratically, using the coordination support, when to compile and execute. This coordination is completed with a Decision-Making tool. The collaborative support is materialized by means of an Instant Message Tool (Structured Chat). In addition, the system has awareness functionalities: a Session Panel and allows the use of tele-pointers.

PlanEdit was originally developed as a component of DomoSim-TPC, a tool to support collaborative problem solving in the field of Domotics [13]. However, it was adapted to programming as it is also a problem solving activity. In particular, its argumentative discussion tool can be used together with SICAS and OOP-Anim to support group work in programming problem solving.

When working together on a programming problem, students propose and discuss possible solutions. This work includes several activities, such as analyse proposed solutions, modify them to create alternative solutions, ask for explanations about proposed solutions, comment them and answer to questions and comments put by other students. All these activities should culminate in the group member's agreement about the best problem solution.

PlanEdit structures student's contributions in a tree and each of them has an associated icon that represents its type (question, comment, agreement, disagreement and so on). It includes also the list of participants and the photo of each contribution author. Interaction with existing contributions and the creation of new ones is made through a set of buttons that represent each possible type of contribution.

CONCLUSION

In this paper we presented the ProLearn project currently under development at the University of Coimbra. This project builds on our previous experience in the development and utilization of animation based simulation tools. We are developing new tools and planning some others, trying to create an environment that is adaptable to student's characteristics, so that it supports each student learning more efficiently. We are also including collaborative learning support, allowing students to take advantage of the benefices of this approach. Other features may be planned and developed during project development.

Each of environment's tools and ProLearn itself will have to be carefully evaluated, especially in terms of educational effectiveness. As most of the members of the team are programming teachers in several higher education institutions, ProLearn will be evaluated and used in those institutions. This will certainly give us a lot of useful information to validate our options and, probably, to define changes and new requirements to be included in the environment.

REFERENCES

- [1] Jenkins, T., On the Difficulty of Learning to Program. In *Proceedings of 3rd Annual LTSN-ICS Conference*, pp 53-58, Loughborough University, UK, 2002.
- [2] Kölling, M., Quig, B., Patterson, A. & Rosenberg, J., The BlueJ system and its pedagogy, *Journal of Computer Science Education*, 13(4), Dec 2003.
- [3] Bergin, J., Stehlik, M., Roberts, J. & Pattis, R., Karel ++: A Gentle Introduction to the art of Object-Oriented Programming. John Wiley & Sons, 1997.
- [4] Byrne, M., Catambarone, R & Stasko, J., Evaluating Animations as Student Aids in Learning Computer Algorithms. *Computers & Education*, 33(5), 1999.
- [5] Brusilovsky, P. & Spring, M., Adaptive, Engaging and Explanatory Visualization in a C Programming Course. In *Proceedings of ED-MEDIA - World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Lugano, June 2004.
- [6] Hong, H. & Kinshuk, Adaptation to Student Learning Styles in Web Based Educational Systems. In *Proceedings of ED-MEDIA - World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Lugano, June 2004.
- [7] Stasko, J., Tango: A Framework and System for Algorithm Animation. *IEEE Computer*, 23(9), pp 27-39, 1990.
- [8] Gomes, A. and Mendes, A., SICAS: Interactive system for algorithm development and simulation. In Ortega, M. and Bravo, J. (eds.), *Computers and Education in an Interconnected Society*, Kluwer Academic Publishers, 2001, pp. 159-166.
- [9] Esteves, M. and Mendes, A., A Simulation Tool to Help Learning of Object Oriented Programming Basics. In *Proceedings of the 34th Frontiers in Education Conference*, Savannah, USA, October 2004.
- [10] Guzdial, M., Kolodner, J., Hmelo, C., Narayanan, H., Carlson, D., Rappin, N., Hübscher, R., Turns, J. and Newstetter, W. Computer support for learning through complex problem solving, *Communications of the ACM*, 39, 4 (April 1996).
- [11] Mendes, A., Esteves, M., Gomes, A., Marcelino, M., Bravo, C. and Redondo, M.. Using simulation and collaboration in CS1 and CS2. In *Proceedings of the The Tenth Annual Conference on Innovation and Technology in Computer Science Education*, Costa da Caparica, Portugal, June, 2005
- [12] Bravo, C., Mendes, A., Marcelino, M. and Redondo, M. Integrating collaboration with animation and simulation in computer-supported Programming learning. In *Proceedings of XXXIII Symposium IGIP/IEEE/ASEE: Local Identity Global Awareness, Engineering Education Today*. Fribourg, Switzerland, September-October, 2004.
- [13] Redondo, M., Bravo, C., Ortega, M. and Verdejo, M.F. PlanEdit: An adaptive tool for design learning by problem solving. In *Proceedings of 2nd Adaptive Hypermedia and Adaptive Web-Based Systems (AH2002)*. LNCS 2347, pp. 560-563. Springer-Verlag: Berlin, 2002.