PESEN - A Visual Programming Environment to Support Initial Programming Learning

António Mendes, Neli Jordanova, Maria Marcelino

Abstract:. *Student major difficulties in programming learning can not be easily overcome even if we use non-traditional approaches, like supporting learning with interactive animation programming tools. Special designed tools are needed for some students. In this paper we present PESEN, a learning tool designed to help novice programming students with major difficulties in the development of basic algorithmic and programming skills. With PESEN students can create, simulate and test small programs for proposed problems based on the movement of simple geometrical figures using a small set of basic commands.*

Key words: *algorithm animation, software visualization, educational technology, programming teaching and learning.*

INTRODUCTION

Over the years, several studies have reported that students evidence difficulties in programming learning, specially in initial stages [1-2]. Also, several approaches and tools have been developed to support those stages [2-4]. Since some beginners find very difficult to grasp the principles of programming it is useful for them to have access to these special instructional programming systems, instead of the commercially available integrated development environments, which are meant for experienced professionals. Our group developed a few such tools to support our students learning [5].

SICAS is an interactive visual animation programming tool that we have used regularly in our classes [6]. It basically allows a student to design, observe, analyze and simulate computer algorithms, using flowcharts. The main objective is to allow students to detect errors, correct them and specially learn from them.

The utilization of SICAS showed us, however, that for some students, those with major difficulties, an approach like this is not enough. Confronted with a request to create an algorithm some students do not know what to do, that is to say what instructions to choose (even if from a small set of available ones, as it is the case in SICAS) and how they should be organized. Usually students understand the problem and sometimes they can make some kind of high level description of a possible solution. However, they have difficulties creating a solution that can be expressed in instructions; no matter we use code, pseudo-code or flowcharts. The language seems irrelevant accordingly to students' own comments and observations. Typical examples are situations where they grasp that there must be a cycle in the solution, but they do not know how to articulate it with other instructions or simply how to control it. Some students cannot even identify that they need a cycle to solve the problem. Programming simulation tools like SICAS are not useful in this situation, since it is necessary that the students make a first solution attempt, so that they can test and improve it.

We decided to develop a tool specially tailored to students that show deeper problems in initial learning stages. We wanted to create an environment where students can practice simple programming constructs in a familiar environment. Also from our experience we have realized that students are especially keen on graphical interfaces and objects. As they are more concrete and give immediate feedback the programming of graphics seems to be easier to understand. This lead to the design and implementation of an educational interactive computer-tool to help the learning of basic mechanisms of programming, using simple graphical objects, named PESEN that will be described in the next section.

PESEN 1.0

PESEN (Programming Environment Simple Enough for Novices) main concept is the programming of elementary geometrical shapes through simple commands. The tool provides a simple environment where the most important classic control structures of programming, like If and While, can be learned in an easily setting. PESEN can also help students learn the basic concepts of sequential, conditional and repetition execution usually present in computer programs. Furthermore indirectly it gives the notion of some basic data structures like arrays. Its basic aim is to support our first year students with difficulties to improve their programming level, so that they can start using SICAS and later the Java programming language.

PESEN has two functioning modes:

- **Student's Mode** and
- **Teacher's Mode**.

To have access to the environment the user must be registered, so that the tasks proposed to him/her take in account his/her previous history in the environment. PESEN is supported on a database where we store information about the tasks and users.

In **teacher's mode** a teacher has an environment, like the one shown in Figure 1, for introducing new tasks (problems) for the students to perform (solve). Every registered teacher can contribute with new tasks.

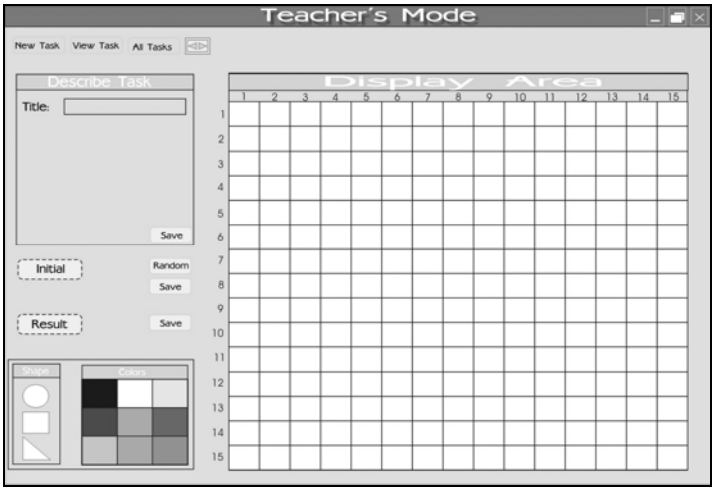

Figure 1 – Teacher environment.

Each task is presented simply as a set of instructor's directions that include three parts:

- Text description,
- Initial state and
- Final state.

These tasks the students have to solve are based on the movement of simple geometrical figures, circles, squares and triangles. The figure parameters that can be changed are just:

- Name,
- Colour and
- Position.

To define a task the teacher has to establish its initial and desired final states, manually or through random positioned and coloured figures. The teacher can choose the type and the number of figures and their colour. For that purpose the system presents him/her several buttons and panels:

Control Buttons: these buttons allow creating a **New Task**, or **View Task**, **All Tasks**, **Change Account Details** and **Exit**.

Describe Task panel: Consists of text fields for adding a title and a description of the problem.

Initial button: When the teacher clicks on this button, it becomes disabled until he/she sets and saves the initial state. This can be done by placing some figures manually and/or randomly in the display area and defining their properties, colour and position.

Result button: This button has similar functionality, allowing the definition of the task desired final state.

Random button: It allows the teacher to establish the initial state of a problem with random positioned and coloured figures. It forces the appearance of the window shown in Figure 3. The teacher can choose the type and number of figures. Furthermore she/he can set the colour of these random shapes by choosing from a combo box or typing the colour's value. The user is prompted also for entering the name and number of the objects.

Manual placement of the figures for the Result and Initial state: The teacher can click over a particular colour and shape, and drag it where she/he wants in the display area. The system automatically names every object and saves a record of its position, and colour in the context of the current task.

Each task is associated with a number of geometrical figures. The system stores the figure parameters in the database (in the initial and result state) and the task they belong to and uses this information to present the task in student's mode.

In **student's mode** the tool provides an environment for the development of simple programs that control the figures' movements, from their initial state to the final state. This work is facilitated by a grid-area (see Figure 2).

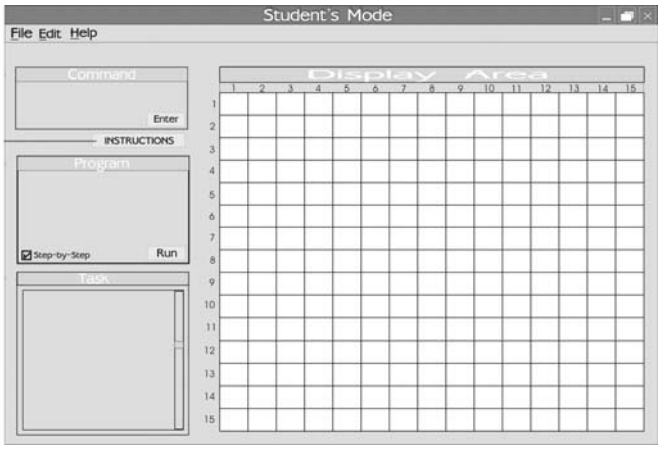

Figure 2 – Student environment.

From the student's perspective, the application interface is very simple, because the system is intended to support programming learning in a very simple and intuitive way. The programming within the environment can be described as "click & type programming" as the student specifies a program using these techniques, providing maximum support to the novice programmer, since a program can be constructed entirely through menu interaction.

The student can control the figures through isolated commands and then create small programs using the set of available instructions. The instructions are selected from a dialog window and then their parameters must be specified. This is intended to avoid syntax errors. There are only three different types of geometrical figures which can be placed on the display area, and the student can instruct them to move around the area according to the particular task. PESEN programs can include simple statements (like

MoveUp and MoveDown) and two control structures (**If** statement and **Repeat While** loop). The whole list of available commands is:

- SetColor (ShapeName, Color)
- MoveUp (ShapeName)
- MoveDown (ShapeName)
- MoveRight (ShapeName)
- MoveLeft (ShapeName)
- Repeat While(Condition) [1. …N]
- If (Condition).

In student mode, the environment is divided into five areas/panels, each for a specific function:

Display Area: A view window in which the figures are placed in their initial state. In this area, the student observes the result of each instruction.

Instructions: Displays the set of available instructions. After the student selects one of them, it will appear in edit mode in the **Command panel**.

Command panel: The student specifies the instruction's parameters, entering values in text fields and/or choosing them from combo boxes.

Program panel: Provides a view of the program instructions. Once the instruction's arguments are entered, the user confirms them and the instruction appears in this panel. The result of its execution is shown immediately in the **Display Area** if the check box **Step-by-Step** is checked or when the user presses the **Run button**. If the user moves the mouse over any instruction, it appears again in the **Command panel**, being accessible for editing or deleting.

Task panel: A window which displays the description of the current problem.

In the next section we describe a particular case of the application's usage.

When the user completes a task, the system stores this information in the database, so that the student can know which problems she/he has already solved and which remain unsolved.

PESEN is developed using Java and MySQL and is in a final stage of development.

EXAMPLE OF UTILIZATION

First the teacher has to define the problem. She/he is allowed to enter the problem's description and then to save it. In the next figure we present an example of a problem introduced by a teacher. It is named **Circles** and in the **describe task panel** there is the problem formulation, along with the specification of circle number 2 parameters:

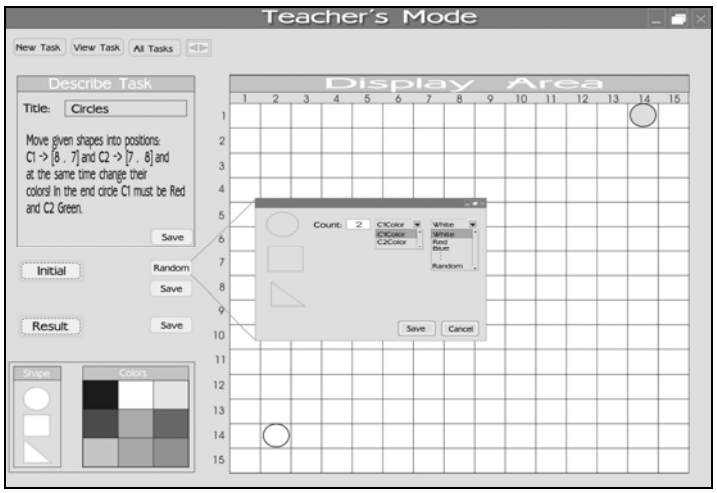

Figure 3 – Example of a problem definition.

In this example, the **initial state** was set through the **Random button** that allows teachers to define randomly positioned and coloured shapes. The teacher needs only enter the number of figures and, if desired, to alter any of their parameters. When finished she/he presses the **Save button**. The next stage is to define the desired final stage. To do this the teacher needs only to place the figures (circles in our example) in the desired final position. The final colour can also be specified.

In Figures 4 and 5 we can see the student's work area.

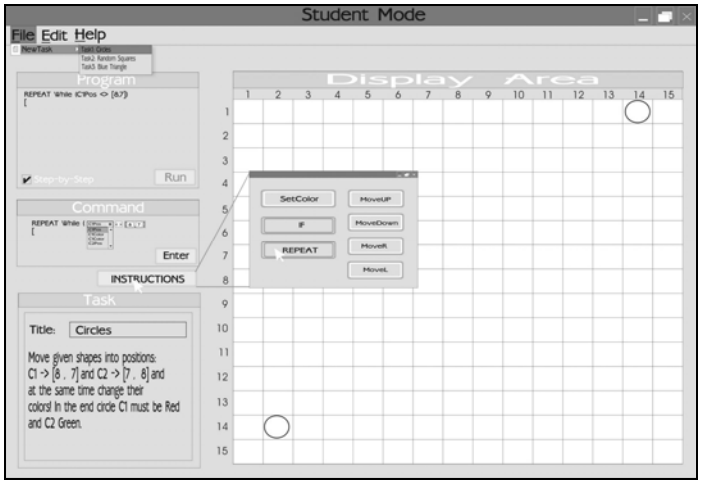

Figure 4 – Student solution to the circles problem (initial stage).

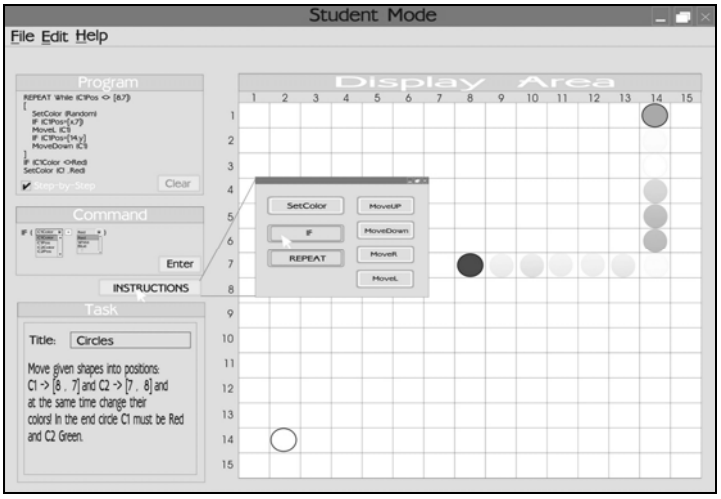

Figure 5 – Student solution to the circles problem (semi-final stage).

First the student chooses from the menu one of the unsolved proposed tasks. The description of the task appears in the **Task panel**. The next step, the important one, is to choose an ordered set of instructions (algorithm) that can transform the initial stage set by the teacher in the desired final state. To do this the student can choose instructions from the dialog window. For example to move C1 circle to position [8, 7] the student can use only Move commands, but we believe she/he will soon realize that the proper utilization of repetition can ease the task and create a much better solution. In this way we hope students can understand the repetition concept and start applying it to solve these very simple programs. In this way they will get used to the concept and may start using it in more complex problems and environments, such as solving problems with common programming languages.

CONCLUSIONS AND FUTURE WORK

Starting to learn programming, especially for students with little or no prior computing experience, is widely known as a difficult experience. This tool, used as an introductory "programming environment" for students with difficulties, allows students to grasp easily and quickly some of the basic programming concepts, because it uses a very simple visual metaphor (based on the control of a few geometrical shapes) and gives the student immediate graphical feedback to her/his actions. This is, in our opinion, one of the most important advantages of the tool.

In the next academic year we are planning to use PESEN with students in practical laboratories during the first classes. Along with this utilisation we are going to evaluate the environment thoroughly, in what concerns interface usability testing and pedagogical effectiveness. We will use direct observation, questionnaires, user interviews and student final grades.

REFERENCES

[1] Milne, I., and Rowe, G., "Difficulties in Learning and Teaching Programming – Views of Students and Tutors". *Education and Information Technologies,* 7 (1), pp. 55-66, 2002.

[2] Madison, S., and Gifford, J., "Modular Programming: Novice Misconceptions", *Journal of Research on Technology in Education.,* 34 (3), pp. 217-229, 2003.

[3] Lavonen, J. M., Meisalo, V. P., Lattu, M. and Sutinen, E., "Concretising the programming task: a case study in a secondary school", *Computers & Education*, 40 (3), pp. 115-135, 2003.

[4] Johnson, D. C., "Algorithmics and programming in the school mathematics curriculum: support is waning – is there still a case to be made?", *Education and Information Technologies,* 5 (3), pp. 201-214, 2001.

[5] Mendes, A. J., Esteves, M., Gomes, A., Marcelino, M., Bravo, C. and Redondo, M., "Using simulation and collaboration in CS1 and CS2", "*The Tenth Annual Conference on Innovation and Technology in Computer Science Education – ITICSE05*", Lisbon, June, 2005 (accepted for publication).

[6] Rebelo, B., Marcelino, M. J. P. and Mendes, A. J., "Evaluation of a System to Support Algorithm Teaching and Learning", *Computers and Advanced Technology in Education Conference – CATE 2005*, Aruba, accepted for publication, 2005.

ABOUT THE AUTHORS

Assist.Prof. António Mendes, PhD, Department of Informatics Engineering, University of Coimbra, Phone: +351 239 790000, E-mail: toze@dei.uc.pt.

Neli Jordanova, BSc, Department of Computer Systems, University of Rousse, E-mail: nrj@mail.bg.

Assist.Prof. Maria Marcelino, PhD, Department of Informatics Engineering, University of Coimbra, Phone: +351 239 790000, E-mail: zemar@dei.uc.pt.