H-SICAS, a Handheld Algorithm Animation And Simulation Tool To Support Initial Programming Learning

Maria Marcelino, Todor Mihaylov, and António Mendes
zemar@dei.uc.pt, xanthus_tim@yahoo.com, toze@dei.uc.pt

Abstract - Mobile devices, like PDAs and smart phones, are small, cheap and light machines capable of being used anytime anywhere. These and other characteristics make them very attractive from an educational point of view. As they have become more and more common and widespread, a crescent interest in developing software applications for them has raised too. Indeed, at present, we find quite easily many applications for them in various areas, including education. However, this is not yet the case for the area of programming learning. Programming learning can be quite difficult for the novice learner and during the years several approaches and tools have been proposed. Some were developed for desktop environments while others for Web-based environments. With the emergence of mobile devices it seems that the next step will include them. This led us to the development of H-SICAS, a handheld algorithm animation and simulation tool. This tool can be used to support initial stages of programming learning, using a procedural approach, and is based on SICAS, a desktop tool developed to achieve similar objectives that has been used with our students with some success. However, this involved several adaptations to be accomplished.

Index Terms – m-learning, programming learning, algorithm animation and simulation, computer science education.

INTRODUCTION

With the appearance and generalized widespread of mobile devices, an interest was raised to investigate its applicability to programming learning. Indeed, they are becoming quite common and we easily find a lot of applications in many areas, like mathematics, chemistry, physics, and for different kinds of learning tasks [1-4]. However, applications to support programming learning using mobile devices are still not common.

SICAS [5] is an algorithm animation and simulation tool, previously developed by our research group that has been used quite successfully till now with our students at the Informatics Engineering Department of the University of Coimbra, in the context of the Informatics Engineering and Communications and Multimedia Degrees [6].

As many of our students have mobile devices, we decided to develop H-SICAS, a version of SICAS to be used in this kind of devices. This would allow students to make a more flexible use of this learning tool, since they can use it in their desktops, laptops and also mobile devices. As its original application, H-SICAS focus on initial programming learning, using a procedural approach. Its target audience includes students who have some difficulties in learning how to program at a basic level, which, in our case, means a significant part of our students.

We will start, in the next section, by making a short overview about some tools that have been proposed to support initial programming learning. Then, in the following section, we will describe SICAS main features. After that we will introduce H-SICAS, the handheld tool we have developed after SICAS. Subsequently we will point out the main issues about H-SICAS adaptation. Finally, we will end the paper with some conclusions.

TOOLS TO SUPPORT INITIAL PROGRAMMING LEARNING

Over the years several approaches and software tools have been developed to support the various stages of programming learning. Among those created to support the initial learning stages, we can mention:

- Simpler Programming Languages or mini-languages, like MiniJava [7],
- Controlled development environments, as BlueJ [8],
- Micro-worlds, from which an example is Karel the Robot [9],
- Tools to test solutions, like ELP [10],
- Tools to algorithm or program animation/simulation, as JELIOT [11] and SICAS.

Many of these tools and solutions are for desktop or Web-based environments. Very few exist however that contemplate handheld devices. As an example we can mention “Learning Objects for Introductory Programming” that was developed after the standalone version [12]. This tool tries to explain abstract programming concepts using visualization techniques. However, our approach is quite different.

According to the authors, several design issues were taken into account during this tool development:

- The limited screen area,
- User interaction with the stylus,
• Text length (it is tiresome to read large texts in a PDA),
• Animations (can be effective if small and simple).

During the tool evaluation authors concluded that users found the PDA more comfortable and providing a more interactive experience than the laptop. Other findings included that it can be used anywhere, it is smaller than a book and thus very convenient, and it gives an opportunity to learn at one own time and pace.

Developers also found that:

• Screen size was not a problem if interfaces and screen layouts were simplified,
• Interactivity was not compromised,
• User experience was more personal.

All those findings were important to us during H-SICAS development as shall be pointed in the next sections.

SICAS STANDALONE

SICAS is a tool that allows the student to build and simulate his own algorithms. Algorithm specification is done through flowcharts (see Figure 1 for an example), but the tool also generates automatically the corresponding pseudo-code (in the language of the user, as some users do prefer this approach) and C and Java codes. This last option can be important for later stages of programming learning, since a student can develop an algorithm first in SICAS and afterwards copy the code to a general IDE for compilation and running.

It has basically two operation modes:

• Teacher mode and
• Student mode.

In the first mode, teachers introduce problems for students to solve, providing, or not, the possible solution(s). These solutions, if given, can be afterwards compared by the students with their own. In student mode, students create solutions (or algorithms) to solve problems provided by teachers.

As SICAS aims to support the initial stage of programming learning, algorithms can include only simple instructions like:

• Assignments,
• Input/output,
• Repetition and
• Selection instructions.

Variables can only be numeric (it encompasses both integer and real numbers), strings and arrays to ease things. Functions can also be defined by the user and used.

Instructions or flowchart elements are introduced by clicking and pointing in the flowchart pane and can be deleted, copied or modified easily at any moment. In order to avoid common novice programmer errors, the information needed for each instruction was reduced to a minimum and connecting lines between flowchart elements are automatically inserted to avoid eventual flowchart inconsistencies.

In Figure 1 we can see an algorithm developed by a student for the following problem:

Write an algorithm that calculates the average age of a group of 5 persons.

This is a classical problem with a cycle and some arithmetic. However many students show some problems when they first try to solve it. For example, a typical approach is to use a lot of variables and avoid the use of the cycle or repetition instruction as they usually feel more difficulty with it.

The problem statement is shown in the bottom pane, the flowchart occupies the main pane and the defined variables are shown on the right pane, under Variables.

In Figure 1 we can see an algorithm developed by a student for the following problem:

Write an algorithm that calculates the average age of a group of 5 persons.

This is a classical problem with a cycle and some arithmetic. However many students show some problems when they first try to solve it. For example, a typical approach is to use a lot of variables and avoid the use of the cycle or repetition instruction as they usually feel more difficulty with it.

The problem statement is shown in the bottom pane, the flowchart occupies the main pane and the defined variables are shown on the right pane, under Variables.
result of each algorithm instruction is. In general, they say SICAS helps them understand how the algorithm works and to find out the reasons why it does not work as expected (when this happens). As soon as execution ends, the final output can be visualized.

In Figure 2 we can see the result of the execution of the previous algorithm.

Now, in the bottom pane appears the final output and in the right pane the last values taken by the various variables, under Facts.

The possibility to simulate her/his own algorithm gives extra possibilities to the student, not available when she/he creates the algorithm on paper (although we think in some situations programming with paper and pencil can be very educational). Using SICAS the student can verify if the algorithm is correct, if it has errors, if it is sufficient to solve the problem, etc., without giving it to the teacher. On the other hand, the tool helps preventing the student from making some logical mistakes, such as using a variable that is not yet declared or incorrectly nesting two cycles or two selection instructions. Although not all possible errors are detected by the tool, this is already a precious help.

We can consider that SICAS is based on a constructivist model, as feedback is discriminative and active, i.e., it does not give a correct answer when the student makes mistakes but, on the contrary, it tries to leads her/him to create an adequate answer. It is also a configurable tool, since it is possible to add new problems, new solutions, change existent solutions, at any time. It allows alternative representations to a problem, since it is possible to have different solutions for the same problem or have a solution in flowchart, pseudo-code, C and Java codes [5-6]. Other programming languages could also be added quite easily and without changing the essential of the tool.

H-SICAS

H-SICAS means handheld SICAS and it was developed after the standalone version of SICAS.

It is also a tool to create and simulate programming algorithms that are specified through flowcharts. It also automatically generates the corresponding pseudo-code, C and Java codes, as the standalone version.

It works in teacher and student modes too with similar functionalities. In Figure 3 we can see a screenshot showing the previous problem specification in teacher mode and in Figure 4 the same problem opened by a student (in the bottom of Figure 4).

After opening a problem, the student can develop an algorithm that solves it. H-SICAS supports the same types of variables and instructions as SICAS does and it uses similar clicking and pointing techniques to specify an algorithm. In Figure 5 we can see a screenshot showing a situation where a student is creating some variables. In Figure 6 the student is defining a repetition instruction to solve the average age problem.

In Figure 7 and 8 we have the full algorithm flowchart for this problem, before and after being run respectively.

The problem statement now is not shown in the bottom pane, but it could be by clicking in the toolbar small window button. The flowchart occupies the main pane of the screen and the defined variables are shown on the right pane, under Var.

After finishing creating a solution it can be simulated. This can be done also in all the 4 speed modes available in the standalone version with a similar behavior. As soon as execution ends, the final output can be visualized too. Now, in the bottom pane appears the final output, if shown, and in
the right pane the last values taken by the various variables, under Var again.

**FIGURE 4**
H-SICAS – STARTING A PROBLEM IN STUDENT MODE.

**FIGURE 5**
H-SICAS – CREATING PROBLEM VARIABLES IN STUDENT MODE.

**FIGURE 6**
H-SICAS – SPECIFYING A REPETITION IN STUDENT MODE.

**FIGURE 7**
H-SICAS – SPECIFYING A PROBLEM IN STUDENT MODE.
H-SICAS ADAPTATION ISSUES

In adapting the standalone version of SICAS to a handheld device, the bigger problem was the pane where the flowchart should be edited and simulated. Indeed, this is the more complex pane we have in the tool from the user point of view, as it can take a lot of objects, scrolling, etc. Scroll helped us to gain space, due to the handheld small screen dimensions. Also we opted by having a smaller fixed toolbar and two hide/show toolbars, for creating the flowchart instructions and simulating the algorithm, instead of only one fixed toolbar with all the tools, as in the standalone version. The first one gives access to the other two by clicking in the first and second buttons respectively. Also the problem, input/output and example panes can be hidden or shown (by clicking in the fixed toolbar third button).

We eliminated the Facts pane to simplify the layout too. Now we always use the Var pane to show variables and their values, both in creation and simulation modes.

Flowchart objects’ dialog boxes were redesigned without problems as well as those for editing, opening and saving problems and for opening and saving algorithms.

Menus were easily adapted too, although the tool has a menu with some complexity to a handheld device.

Pseudo-code, Java and C panes were not problematic either, since they were implemented as text panes and the amount of text is usually not large.

CONCLUSIONS

In this paper we described H-SICAS main features, a handheld tool to support initial programming learning using a procedural approach. It was developed after a standalone tool with similar purposes and the development involved mainly redefining some screen issues, like having hiding/showing toolbars and panes and reducing the number of panes. Surprisingly, the tool screen is perfectly manageable as users experienced it. That was one of the key design questions for us. Although we need more testing, namely with students, the results from the first usability studies that involved only teachers were positive.

In the future H-SICAS will be integrated with a Web-based system that will include several tools and where teachers can contribute with problems that students can solve, working easily at distance or downloading them using their handhelds. As it can be hard for a single teacher to provide problems for such a system, the idea is to build a sharing platform that can be fed by several teachers and used by many students. For that some small adaptations must be implemented in H-SICAS, namely the synchronization with the system, as we want that users will be able not only to download problems, but also to submit their solutions for validation.

REFERENCES


AUTHOR INFORMATION

Maria Marcelino, Assistant Professor, University of Coimbra, zemar@dei.uc.pt.

Todor Mihaylov, Master Student, University of Rousse, xanthus_tim@yahoo.com.

António Mendes, Assistant Professor, University of Coimbra, toze@dei.uc.pt.