Web-Based Graphic Environment to Support Programming in the Beginning Learning Process

Carlos J. Costa[1], Manuela Aparicio[1,2,3], and Carlos Cordeiro[4]

[1] ADETTI-IUL/ISCTE-IUL, Lisboa, Portugal
[2] IADE, Lisboa, Portugal
[3] UNL, Lisboa, Portugal
[4] Springfield Collegiate Institute, Manitoba, Canada
{carlos.costa,manuela.aparicio}@iscte.pt,
ccordeiro@sunrisesd.ca

Abstract. The present paper focus on the computer programming for a beginning level of learning. Students´ learning challenges were identified through literature review. We propose a solution that enables students to interact with an editor that gives an output as a response. We also analyze the impact of the use of this framework and identify that some support tools. Preliminary evaluation shows that some of the support tools are more effective than others.

Keywords: Computer science education, engineering education, computer programming education.

1 Introduction

Learning is a process of acquiring knowledge. In the area of computer programming, also known as computer coding is often regarded as a difficult task. In order to improve programming learning, Richard E. Pattis developed an alternative method that enabled an easy way for introducing students in computer programming[6] named Karel. It introduces a language repertoire to imperative commands whose actions can be visually displayed. Several environments have been developed to introduce younger students to the concepts of programming through robotics (whether real or virtual). These include Papert's Turtle Graphics[34], Pattis' Karel [6] and Lego Mindstorms [9],[10]. The use of robots is inherently attractive to many young students (even more attractive than 2D graphics). However, the constraints imposed here by the real (or virtual) world have to do with the available kits that give these environments an essentially compositional focus, and restrict the opportunities for exploration and extension by students, such as the robot construction itself and technical domain of hardware assemblage. There have been many efforts over time to develop initial learning environments to beginners, focused on either through direct manipulation of objects or through data and structure visualization. Following in this paper we describe the main problems faced by students. We describe a solution evaluation phased and present results on that evaluation.

M. Herrlich, R. Malaka, and M. Masuch (Eds.): ICEC 2012, LNCS 7522, pp. 413–416, 2012.

2 Programming Learning Problems

Students face diverse problems when they are learning programming [3] mainly because programming is dynamic and abstract. Several authors have identified which are the main difficulties in learning programming ([5]; [1]; [8]; [4]). Teachers deduce more difficulties than the perceived problems by the students. Programming novices often tend to focus on specific aspects rather than general ones. These results are verified by several studies ([4]; [5]). To start, student and teacher perceptions on programming's knowledge base are rather different, as teachers understand better the student´s limitations than themselves. Learning programming contains several activities, such as learning a language features, variables, program design and program comprehension. A study conducted by Rist [11] demonstrates that syntax understanding is not the main difficulty. Students may know the syntax and semantics of individual statements, but they do not know how to combine those elements in order to produce valid programs [12]. The cognitive domain plays an important role in the process of learning programming [7]: data recall (forget to declare initialize variables), understanding of the meaning of the problem, difficulty in translating that into a logic program, difficulty in algorithm design and applying a concept in a new situation, analysis of the program structure, and synthesis difficulty in integrating different modules. Furthermore, there are a myriad other reasons programming is difficult to learn [2],[14] like originality, many skills, program design and comprehension, choice of language, timing and course structure, and varied individual student abilities.

3 Proposed Solution

We propose a solution in which students are able to visualize and interact within one screen, in a dual interface, having a programming instructions, each has a graphical icon, at the same time they can also see a dragon moving according to the commands. A manual was developed with samples as supporting documentation, consisting of a list of procedures [13]. The manual describes basic procedures, conditional constructs, looks and variables, with simplified examples. On the right side of the screen, students visualize a dragon moving accordingly to the given instructions on the left side of the screen. If a student wants the dragon to move to the right, he/she must push the green right arrow. Students have a small number of available instructions to accomplish a specific task, as required by the teacher. The left side of the screen also displays programming command lines they appear automatically as the buttons are pushed. This is an important feature in order to allow the student to become familiarized with lines of code. The system also has different patterns or labyrinths that correspond to various levels of difficulty, within the exercises.

4 Evaluation

In order to evaluate opinions, this system was used by a group of first year computer science students. Students were asked to fulfill a questionnaire, using a 7 point Likert scale; the number of valid answers is 78. Concerning types of support, the manual was considered to be the least preferred type of support. Examples and the manual were most preferred by the students. This indicates that the students were not familiar with the subject.

Table 1. System support according to basic programming concepts

	Example (M)	(SD)	Manual (M)	(SD)	Stat t	Sig.
Basic procedures (move, turn left...)	5.56	1.58	5.12	1.45	2.47	.02
conditional constructs	5.04	1.55	4.96	1.47	.47	.64
Loops	4.78	1.65	4.79	1.61	-.08	.94
Functions	5.18	1.31	4.91	1.43	1.86	.07
Variables	5.01	1.32	4.88	1.40	.87	.39

M – Sample Mean, SD – Standard Deviation.

Also, the system was evaluated on what extent it supports writing of a specific type of code. On the other hand, the results of the system support were also evaluated.

Table 2. System support perceived by the students, and system result

	System Support (M)	(SD)	System Result (M)	(SD)	Stat t	Sig.
Complete program	4.73	1.33	5.27	1.40	-3.64	.00
Basic procedures (move, turn left...)	5.34	1.14	5.32	1.30	.09	.93
conditional constructs	4.88	1.40	5.03	1.33	-1.09	.28
loops	4.83	1.32	4.88	1.32	-.41	.68
functions	4.83	1.26	5.08	1.27	-1.72	.09
variables	4.78	1.32	4.94	1.40	-1.19	.24

M – Mean, SD – Standard Deviation.

The system support is based on manuals and exercises. The students were asked to perform the exercises, and the system displayed the dragon moving across the screen. Students found that the system provided good for supporting exercises´ construction, according to the values in Table 2. The students' results provided by the system were more positive regarding the perceived systems' support. But t Stat shows that means are not statistically different, except in the issue of complete program.

5 Conclusions

According to the literature review, there are several challenges related to the learning of computer programming. Some of them may be partially answered by utilizing virtual robots. We proposed a solution (dragon-robot) and than we analyzed the impact of the system usage. We found that that some support tools are more effective than others.

Acknowledgments. Research presented here is partially financed by FCT – Portuguese Research Ministry.

References

1. AlaMyka, K.: Problems in Learning and Teaching Programminga literature study for developing visualizations in the CodewitzMinervapoject. Codewitz Needs Analisys, lIteratur Study (2005)
2. Cummings, S.: Feedback 2.0: An Investigation into using sharable Feedback tags as Programming Feedback. Ph.D. Thesis, Durham University (2010)
3. Donmez, O., Inceoglu, M.M.: A Web Based Tool for Novice Programmers: Interaction in Use. In: Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2008, Part I. LNCS, vol. 5072, pp. 530–540. Springer, Heidelberg (2008)
4. Lahtinen, E., AlaMtka, K., Jarvinen, H.: A Study of the Difficulties of Novice Programmers. In: ITCSE 2005 Proceedings. ACM, Portugal (2005)
5. Milne, I., Rowe, G.: Difficulties in Learning and Teaching Programming— Views of Students and Tutors. Education and Information Technologies 7(1), 55–66 (2002)
6. Pattis, R.: Karel the Robot: A Gentle Introduction to the Art of Programming. John Wiley and Sons, New York (1981)
7. Renumol, V., Jayaprakash, S., Janakiram, D.: Classification of cognitive difficulties of students to learn computer programming. Indian Institute of Technology, India (2009)
8. Sajaniemi, J., Hu, C.: Teaching Programming: Going beyond "Objects First". In: 18th Workshop of the Psychology of Programming Interest Group, pp. 255–265. University of Sussex (September 2006)
9. Papert, S.: Mindstorms: Children, Computers, and Powerful Ideas, New ed. Basic Books (1993)
10. `http://Lego.com`
11. Rist, R.: Teaching Eiffel as a first language. Journal of Object Oriented Programming 9, 3041 (1996)
12. Soloway, E., Spohrer, J.: Studying the Novice Programmer. Lawrence Erlbaum Associates, Hillsdale (1989)
13. Costa, C., Aparicio, M., Cordeiro, C.: A solution to support student learning of programming. In: Proceedings of the Workshop on Open Source and Design of Communication (OSDOC 2012). ACM, New York (2012)
14. Piteira, M., Costa, C.: Computer programming and novice programmers. In: Proceedings of the Workshop on Information Systems and Design of Communication (ISDOC 2012), pp. 51–53. ACM, New York (2012)