Improving Students Learning Programming Skills with ProGames - Programming through Games system

Raquel Hijón-Neira[1], Ángel Velázquez-Iturbide[1], Celeste Pizarro-Romero[2], Luís Carriço[3]

[1] Departamento de Lenguajes y Sistemas Informáticos I, [2] Departamento de Estadística e Investigación Operativa, Universidad Rey Juan Carlos
[3] Departamento de Informática, Universidade de Lisboa

{raquel.hijon, angel.velazquez, celeste.pizarro}@urjc.es, lmc@di.fc.ul.pt

Abstract. We present a system for learning programming skills, ProGames, through a leveled set of visually-attractive and interactive programming exercises in Greenfoot, categorized by students likes to offer them solutions to problems they really enjoy or like most. The system has been evaluated during the course 2012-13 in 3 Computer Science Degrees and our results show very positive acceptance by the students.

Keywords: Programming teaching, Interactions Analysis, e-Learning, Moodle, Greenfoot, Visualization**.**

1 Introduction

The difficulty of learning abstract concepts, generally unknown to students, and the lack of systems proposing the unification between technical programming aspects and methods currently applied to foster student motivation, provoke in them lack of interest, failure and abandonment. That is why many concepts and methods of programming constructs are currently rethought. Professor Mark Guzdial [1, 2, 3] of Georgia Institute of Technology is considered a pioneer in teaching programming and sciences learning. He proposed a series of innovative activities aimed to enhancing the teaching-learning process in the engineering field. The main objective of ProGames, the system presented in this paper, is facilitating learning and foster motivation into the programming area. To this purpose, we propose a comprehensive set of programming games that are in fact exercises arranged into categories that students individually can select and complete according to their particular tastes. Therefore, there will be a wide

range of solutions to problems in environments attractive to students where they will likely feel comfortably capable. Once a given student has been categorized, he will start his own path of learning proceeding progressively through the levels of the chosen category. Moving to the next level is possible only after successfully completing all the tasks in the current one. The process can be repeated as many times as needed, until success is attained.

To provide visual interactive solutions to the programming exercises, the Greenfoot [4] development environment has been used because: i) its target audience is the educative community, and, ii) it includes tools for developing great visually-rich interactive applications. We have proposed 4 categories or scenarios for the problems. Each category consists of 7 levels of contents. Therefore, a set of exercises is proposed for each category and its 7 levels. All in all, a total of 192 different exercises are proposed. Students will download the set of exercises for each level, to be executed in Greenfoot Programming Environment. Thus, they can visually execute the interactive exercise solution, and understand it intuitively, providing a solution to a real case. Later on, students must study deeply the programming code, to understand it and become familiar with the different programming techniques used.

2 Existing systems

One of the earliest pieces of software developed to teach programming is Logo [5], a high level language designed for teaching beginners to program. Another is Greenfoot [4], a restricted Java environment with custom high level libraries. In addition, much attention has lately focused on using a combination of visual programming with a 'storytelling' aspect, to make the experience of programming more immersive for children; namely Alice [6] and Scratch [7]. Letting children create their own stories in an open-ended way means that they can connect with their interests, which can be a key factor for motivation [8]. Using visual programming lets younger children become more easily involved in the process of programming both by removing syntax problems, and also through basic HCI mechanisms, such as recognition instead of recall.

A problem with simplified programming systems is that students often feel that they are not doing real programming, which can have a negative effect on both confidence and interest in programming. Related to this, is requiring students to make a commitment to learning the language and environment of the system. This is often somewhat wasteful since it can take a long time, and once they finish the course they will likely never use the system again – after all, these are not 'real' programming languages. There are a number of games, including LightBot [9], Robozzle [10], Carnage Heart [11] and VISPROCON [12] that make use of programming as the core game play mechanic. Although these games have a reduced programming fidelity compared to those mentioned above (e.g. Scratch and Alice), they do offer a lot of advantages. Games generally do not rely on teachers, which for the aforementioned reasons can be advantageous.

Project Euler is a series of challenging mathematical/computer programming problems that require more than just mathematical insights to solve. The trick is to craft a ladder of increasingly difficult levels, each one building on the last [13]. Project Euler is popular and engaging, in part because it is set up like a video game, with 340 fun, very carefully ordered problems.
With the right programming environment, students with even fairly rudimentary programming skills can produce software that is graphically impressive and supports complex user interactivity [14]. It is generally accepted in the literature that a programming assignment that involves building a real application is more rewarding than one that involves an isolated and unrealistic processing task [15].

The target audience of users of Greenfoot includes students from about 14 years of age, and scales up into introductory university education [4]. Users of Greenfoot may be complete programming novices within that age group, or they may have had prior programming experience in Scratch, Alice, or comparable systems. Concepts learned in Alice and Scratch transfer well into Greenfoot. The programming language used in Greenfoot is standard Java. The use of a textual language in general (as opposed to the Scratch/Alice model of drag-and-drop instruction blocks) is considerably harder for young learners to master. But we are using it at university level. Using Java in Greenfoot is considerably easier than using Java in a standard professional environment (be it a text editor/command line environment or a standard professional IDE).

3 The ProGames system organization

To determine the student's preferences, we have created an ad-hoc website to offer a personal interest test based in the manual Kuder-C [16, 17]. This manual is applicable to teenagers and adults, so it perfectly adjusts the ProGames target audience. It offers an evaluation of interest of the subject on 10 different fields of interest with 168 questions on: Open Air, Mechanical, Calculus, Scientific, Persuasive, Artistic, Literary, Musical, Assistance and Administrative. The results obtained by a subject point out his likings or preferences for given type of activities. We have simplified it to the 4 more representative categories and only 20 questions. The chosen categories are: 1- Open Air: indicates preference for activities generally performed outdoors. 2- Artistic: indicates preference for creative works, generally dealing with (pleasing) works, such as drawing, coloring and attractive materials. 3- Assistance: indicates preference for activities that imply helping other people. 4- Calculus: indicates preference for numeric tasks and solving of mathematical problems.

The test would also give the affinity percentage to the other three categories. So the students, if desired, can complete his learning with the proposed exercises on the other categories, according with their preferences.

Each categories have seven levels of exercises that increase in difficulty: 1-basic programming constructs 2-structured instructions, 3- sub programming, 4- introduction to recursion, 5- arrays and algorithms with arrays, 6- files and records and 7- complex data types. He would firstly pick the category he has more affinity to, and start progressively with the adapted contents (lessons) for that category. In each

of the seven lessons a set of exercises is proposed. The main idea is that the student downloads the exercises and executes them in the Greenfoot Programming Environment. Thus, he would visually check the exercise solution and, intuitively, understand it; so he has also a real use case for a given problem. Following to these, student must deeply study the exercise code, to understand it and learn the programming skills of each lesson.

The ProGames system has 192 exercises (49 for *open air*, 48 *Artistic*, 47 *Assistance* and 48 *Calculus*). These visually rich scenarios and the questions for their testing for learning programming have taken over 9 months of a full time Java programmer.

4 Example of Interactive materials: Recursion with the Towers of Hanoi

The process for each exercise is that student would interact with the game, execute it several times, for different inputs and see what happens; once this phase is over, he would go to the code, and try to understand how the interactive game has been made. Next to that he would be asked a question about it, in order to check if he has understood it or not. In case of failure, he would have to repeat the process until the concepts become clear and answers are right.

Recursion is traditionally a very challenging concept for new programmers to master. Into the interactive Greenfoot world we are able to demonstrate recursion with the game "Towers of Hanoi". The game explains the movements required to transfer be made to move an entered number of discs from the first tower to the last, using as an intermediate tower the middle one. The player can move only a disc at a time, and a disc cannot be placed on top of a smaller one. Students can play the game varying the input data, which is the number of discs, just be clicking on the Act button (Figure 1).

Once the game has been played, student is requested to edit the Towers class to see the recursive function used (Figure 2). The variables origin, destination and aux correspond to the first, third and second tower respectively, and the n contains the entered number of discs for that execution.

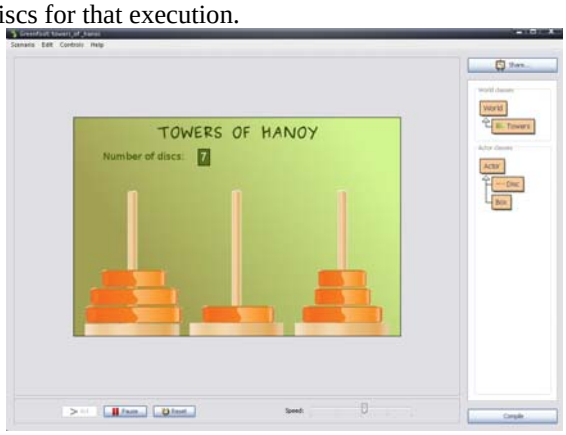

Fig. 1. Recursion with Greenfoot Towers of Hanoi Game

Students can now understand how a recursive function makes call to itself, and what are the parameters on every call. With some more games of this type in the selected category, the student will be ready to be examined by the virtual teacher in Merlin-Know, as explained in the next section, or on any other system.

```
private void moveDiscs(int n, int origin, int destination, int aux){
    if(n>=0){
        moveDiscs(n-1,origin,aux,destination);
        move(n,destination);
        moveDiscs(n-1,aux,destination,origin);
    }
}
```

Fig. 2. Source Code for the Towers of Hanoi Game

5 Acceptance by Students

The sample of students belonged to 3 Degrees of the Rey Juan Carlos University of Madrid: Computer Science Degree, Computer Science – Mathematics Double Degree, and Computer Science – Business Administration Double Degree. The three degrees had the same teachers, course materials and schedule. ProGames has been tested in the first semester (from September 20th to December 20th) of the academic year 2012-2013. The experiment included 44 CS1 (Computer Science) students of the 3 degrees of the course "Introduction to Programming", including: basic programming con-structs and structured instructions, sub-programming, introduction to recursion, ar-rays, files and records. We have combined the quantitative data provided by the ProGames system thanks to Merlin-Know and Merlin-Mo [18] and the subjective information provided by students through to a survey that students sent to teachers, using Google drive [19] technology, a survey on how students interacted with the system, to which degree they liked the various parts of it or how they reached it, is presented on this section.

The use of the ProGames was an extracurricular activity, only taken by 44 of the 117 students belonging to the 3 Degrees, who voluntarily wanted to; the use of the ProGames system was incentivized with the additional recognition of 1 ECTS credit when they passed with a grade of 7/10 all the tests in Merlin-Know (an ad-Hoc module for Moodle to test the progress in learning of students and that offers them feedback) of only one or two categories, 1.5 ECTS credits when have done the same with 3 categories and 2 ECTS credits when the 7/10 grade was obtained for all the tests of the 4 categories.

The data analysis has been done with Many Eyes [20] that allows interactive analysis. Figure 3 presents the analysis of categories, on the x-axis the seven lessons for each category (from 1 to 7), on the y-axis, the 4 categories (from bottom to top: *Assistance*, *Calculus*, *Artistic* and *Open Air*), the bubble size indicates the number of students that chosen it, and color indicates the acceptance grade (legend is on the top right). On the categories analysis, the first three were clearly more picked than the last one, *Open Air* which is the last one that student completed, about ¼ less students completed it (44 to 29 approximately). It clearly seems that the last of students 'choice was Open Air, which can be positive for future computer science profession-

al's which work will mostly be done indoors. If exercises are proposed for students on these 3 Computer Science Degrees, on the first three categories, they will probably have better acceptance. Taking into account the lesson analysis for categories, it clearly seems that from lesson 4 onwards, there are a higher students' percentage 'not liking much' the lessons; we think it can be because the difficulty grade increases, since these lessons are: recursion, arrays and algorithms with arrays, files and registers and advanced data types. Actually, first three lessons are more liked by students, particularly lesson 3 in the calculus category, which has had absolutely no dislikes. On the contrary, almost ¼ of students did not like lesson 7 in the Artistic category, whose exercises should be improved on future ProGames versions. All in all, lesson 3 in all categories has been exceptionally well received; it is the one corresponding to subprogramming, which is a very important concept taught on an introductory programming university course.

Fig. 3. Scatterplot of categories comparison by lesson, size and likeness

Figure 4 shows students preferences selected on the Kuder-C test [16, 17] of vocational preferences taken by students for their categorization at the beginning of the experiment.

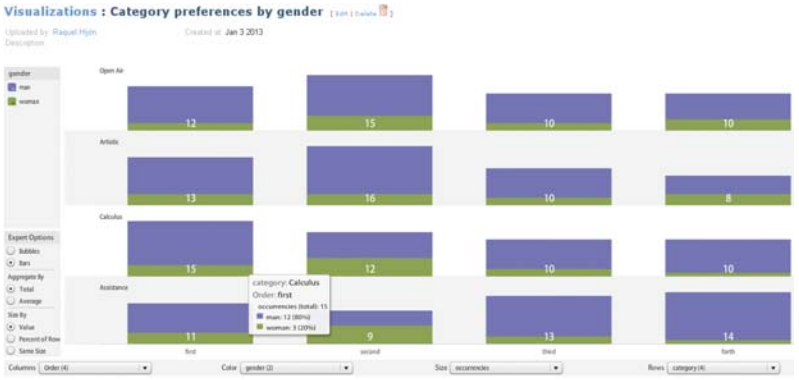

Fig. 4. Scatterplot of Categories chosen by genders

On the x-axis is shown the order of categories' preference they had marked on the initial test based on it, and therefore followed into the ProGames system. On the y-axis the bar size indicates the number of occurrences, and the color differentiates

between men and women. The category chose on first place has been calculus, for both men and women. Therefore, students on Computer Science Degrees show preference for these types of activities, so if we offer them this type of problems, they will accept them better. It is very closely followed by the Artistic an Assistance categories. Considering the gender analysis, the percentage of women liking the Assistance category on the second place is considerably higher than in any other category. Anyway, their first choice is also calculus, as it is also for men

For the interfaces definition task, it is very important to determine how students access to the ProGames system. Thy used PCs (40%), laptops (43%), notebooks (9%), smartphones (5%) and tablets (4%). The reason why the smartphones' percentage is so low is because the Greenfoot API needs to be installed on a computer in order to be able to execute the exercises.

When at the end of the evaluation students were asked, 'in general have you liked working with the ProGames system?" they answered that quite much (48%), very much (25%), a little bit (20%) and not at all (8%).

6 Conclusions and future work

It was a challenge to think on a system for learning programming that fit the students' personal needs and liking, aimed to improving their learning process. Furthermore, it was our goal to offer the 192 leveled sample exercises in an environment interactive, visually attractive and through programming games. It was very important being able to offer the ProGames system to students in a LMS that nicely supported it. Finally, it was important for us being able to test the progress of students through their learning into the system, to analyze their interactions, and also show them their personal progress, performance and comparison with their classmates. Those objectives have been fully satisfied. Firstly, the categorization has been possible thanks to the development of a web system based on Kuder C test of personal interests. Secondly, the interactive and very attractive to students' game-based programming environment for the development of the 192 programming exercises has been Greenfoot. Thirdly, the LMS system proposed hosts the system the collection, Moodle, and the new module developed for it, Merlin-Know, with its virtual teacher that guides and motivates students learning and show them their progress. As a whole, we feel very satisfied with the solution proposed.

ProGames system has been tested on 3 Computer Science Degrees of Escuela Técnica Superior de Ingeniería Informática of Universidad Rey Juan Carlos during the academic year 2012-2013. The students' interaction analysis and their subjective opinion about the system have been very satisfactory. Almost all exercises on almost all categories were very much enjoyed, successfully accomplished and accepted, which has arisen a very positive opinion on them.

More solid studies are carried out to compare possible differences in increase on knowledge between ProGames and the traditional learning method.

References

1. Lauren Rich, Heather Perry, Mark Guzdial. (2005) "A CS1 Course Designed to Address Interests of Women." Technical Symposium on Computer Science Education, Proceedings of the 35th SIGCSE technical symposium on Computer science education, Norfolk, Virginia, USA, Pages: 190 – 194.
2. Tew, Fowler, and Guzdial. (2005) "Tracking an innovation in introductory CS education from a research university to a Two-Year College." Technical Symposium on Computer Science Education, Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis, Missouri, USA, Pages: 416 - 420.
3. Ericson, Guzdial, and Biggers. (2005) "A Model for Improving Secondary CS Education." Technical Symposium on Computer Science Education, Proceedings of the 36th SIGCSE technical symposium on Computer science education, St. Louis
4. Kölling, M. 2010. The greenfoot programming environment. ACM Trans. Comput. Educ. 10, 4, Article 14 (November 2010), 21 pages.
5. "Logo Foundation," MIT University, [Online]. Available:http://el.media.mit.edu/logofoundation/ index.html. [Accessed 25 03 2013].
6. "Alice.org," Carnegie Mellon University, [Online]. Available: http://www.alice.org/. [Accessed 25 03 2013].
7. "Scratch Home," MIT University, [Online]. Available: http://scratch.mit.edu/. [Accessed 25 03 2013].
8. R. Koster, A Theory of Fun for Game Design, Paraglyph Press, 2004.
9. "light-Bot," Armor Games, [Online]. Available: http://www.kongregate.com/games/Coolio_Niato/light-bot. [Accessed 25 03 2013].
10. "RoboZZle online puzzle game," [Online]. Available: http://robozzle.com/. [Accessed 25 03 2013].
11. "Carnage Heart," [Online]. Available: http://en.wikipedia.org/wiki/Carnage_Heart. [Accessed 25 03 2013].
12. Bromwich, K., Masoodian, M., & Rogers, B. (2012). Crossing the game threshold: A system for teaching basic programming constructs. In 13th International Conference of the NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction, CHINZ 2012, Dunedin, July 2-3 2012, (pp 56-63). Dunedin, New Zealand.]
13. http://m.theatlantic.com/technology/print/2011/06/how-i-failed-failed-and-finally-succeeded-at-learning-how-to-code/239855/)
14. Haden, P. The incredible rainbow spitting chicken: Teaching traditional programming skills through games programming. in Proceedings of the 8th Australian Conference on Computing Education. Australian Computer Society, Darlinghurst, 2006, 81–89.
15. Huang. T. Strategy game programming projects. The Journal of Computing in Small Colleges. 16(4) 205-213, 2001.
16. Kuder, G.F. (1986) Professional preferences, Form C, Vocational, Tea Ediciones.
17. G. F. Kuder. The Stability of Preference Items.Journal of Social Psychology, pages 41–50, 10 1939.
18. Hijón-Neira, R., & Velázquez-Iturbide, J. Á. (2011, June). Merlin-Mo, an interactions analysis system for Moodle. In Proceedings of the 16th annual joint conference on Innovation and technology in Computer Science Education (ITICSE'11) (pp. 340-340). ACM.
19. Google Drive https://drive.google.com
20. Many Eyes IBM Research and the IBM Cognos software group [Online] http://www-958.ibm.com/software/analytics/manyeyes/ [Accessed 25 03 2013]