Scratch: Applications in Computer Science 1

Inés Friss de Kereki
ORT Uruguay University, kereki_i@ort.edu.uy

Abstract - **Programming is a complex intellectual activity. We observed through the years that it is difficult for some students to understand the logic of a program and to familiarize themselves with the control structures. In order to help smooth this initial relationship with programming, we tried Scratch (a programming language designed for young people, developed by MIT's Media Lab). We analyze the use of Scratch in two Computer Science 1 courses: one in university degree and other in vocational studies. We use this tool in the very first weeks of those courses with the purpose of improving students' programming experiences and motivation, and also to detect its influence, if any, in scores and dropout rates in comparison with normal courses. We developed detailed lab guides, exercises, tests and questions forms. We contrast the results with normal courses and found that students who used Scratch expressed higher motivation but there was no statistically evidence of differences neither in dropout rates nor obtained scores. In this paper we present the detailed courses, the experimentation and the results. We offer some conclusions and reflections over the particular value of including this kind of tool. We include suggestions with the intention of improving Scratch.**

Index Terms - Computer Science 1 (CS1), Object Oriented Programming, Teaching.

INTRODUCTION

Programming is a complex activity. The importance of a properly constructed first course in programming cannot be overstated [1]. We observed through the years that it is difficult for some students to understand the process of programming and to familiarize themselves properly with the control structures. In order to help smooth this initial relationship with programming, we tried some tools in Computer Science 1 (CS1) and chose to experiment with Scratch [2] in the first semester.

The organization of this paper is as follows. First, we present a collection of different approaches and tools for CS1. After that, we explain our proposal for the courses (university and vocational). We describe Scratch and its use in the courses and we present the experimentation and results. Finally, we offer some suggestions to improve Scratch and draw conclusions.

TOOLS FOR COMPUTER SCIENCE 1

Robins, Rountree and Rountree [3] indicate that learning programming implies the acquisition of new complete knowledge and its related strategies besides practical skills. The most important weaknesses that learners show are related to activities for solving problems, designing solutions and expressing those designs as programs.

There are different approaches for teaching programming. Some of them emphasize individual work versus collaborative work [4], other approaches are based on simulation tools [5], the use of special environments [6], [7], [8], the use of games [9], [10], [11], pair programming [12], [13], role games [14], test-first or test-driven approach [15], [16], and, or, CRC cards [17]. Bruce [18] states that there is not enough information to evaluate how effective pedagogic tools, such as pedagogic environments, special libraries providing utility classes or micro worlds, are in teaching initial courses.

Also, some particular environments, like Alice [8], Jeroo [19] and GreenFoot [20], were specially considered and tested in small groups in our university. The preliminary evaluation of those tests showed that the tools were not so easy to use and the learning curve -the relationship between the duration of the experience and the resulting progress- seemed to be too high. It involved too much time and the results appeared to be not significant.

Malan and Leitner [21] propose the use of Scratch as a first language in introductory courses, as a viable gateway to Java. They used it in a summertime version of a CS1 course. In their work, they applied two lectures and two problems set on Scratch (one week) and after that, transitioned to Java. Their goal was not to improve scores but also to improve first-time programmers' experiences. Conclusions are that "Scratch excite students at a critical time (i.e., in their first foray into computer science), it also familiarizes the inexperienced among them with fundamentals of programming without the distraction of syntax". They also proposed variations of the problem sets. In the article, they did not compare with another course, also there was no comparison or analysis related to scores or dropout rates.

COMPUTER SCIENCE 1 AT ORT URUGUAY UNIVERSITY

In this work we are considering two versions of the CS1 course: one at Engineering career (university) and the other in vocational studies. The university course is divided in 15 weeks. It has 4 hours of lectures and 2 hours of lab sessions per week. There are 2 compulsory programming

assignments and a test, plus a final exam. The course prepares the learner for constructing simple programs using the object-oriented paradigm. By the end of the semester, the student should be able to analyze simple situations, design possible solutions and implement them with an object-oriented approach. The programming language used is Java. A brief description of the course is given in the following table (Table I):

TABLE I
COURSE DESCRIPTION

Week	Topics
1-3	Variables, pseudo code, control structures
4	Classes, objects
5-7	Relations between classes: Association
8	Inheritance
9-10	Aggregation, Collections
11	Enumerations
12	Sort and Search
13-15	Advanced use of collections

Traditionally, in the first weeks we use pseudo code to present the main concepts of variables and control structures. Over the years, we observed that for some students, it was really difficult to design the solution for a simple problem or to "visualize" and, or, understand an approach for solving the problem. We stimulate the students to manually simulate the execution of the program acting like the computer. In order to reinforce those ideas we included this semester the use of Scratch.

The CS1 course in vocational studies has almost the same characteristics. The main difference is that the selected language is Visual Basic, but the course description is basically identical. One additional difference of the vocational course is the age of the students; they are 16-18 years old. The students of the university courses are usually from 18 to 21 years old.

SCRATCH

"Scratch is a new programming language that makes it easy to create your own interactive stories, animations, games, music, and art. Scratch is designed to help young people (ages 8 and up) develop 21st century learning skills. As they create Scratch projects, young people learn important mathematical and computational ideas, while also gaining a deeper understanding of the process of design" [2]. It is free software.

The main areas in the screen (See Figure 1) are: the blocks palette (left in the Figure 1), the script area (in center) and the stage (on the rigth side). The blocks palette includes the blocks used for programming the scripts. The blocks are grouped by motion (move, turn, point, etc.), looks (to control the background and costume of the elements), sound, pen (to select the characteristics of the pen used to draw), control (if, repeat until, repeat, etc.), sensing (to check the sensors, for instance if a sprite is touching other sprite), numbers (the group includes basic arithmetic operations, random numbers and comparison operators) and variables (to create, to store a value, to change it and to report its value). For creating a script the user must drag the desired blocks into the script area and snap them. After that, to execute the script the user double clicks over the script or press the green flag. In the stage area appears the result of the script.

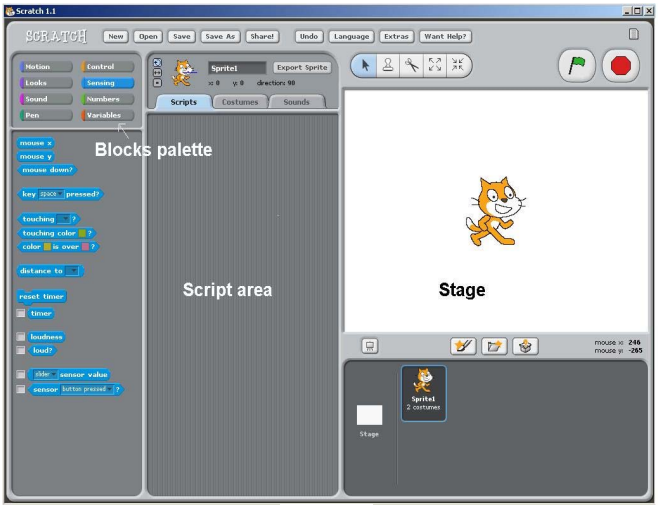

FIGURE 1
SCRATCH

USE OF SCRATCH

Malan and Leitner [21] propose the use of Scratch as a tool to help students to understand the basic ideas of programming. They deployed Scratch via Harvard Summer School's Computer Science S-1: Great Ideas in Computer Science, the summertime version of a course at Harvard College. We applied Scratch for freshmen in ORT Uruguay University. We use it in two extended courses, one for university students and other for vocational students.

We used Scratch in the 2nd semester of 2007, for 3 weeks of our 15 weeks course. In the first week we presented the basic concepts of programming like loops and conditions. The main exercises proposed were similar to, for instance, drawing one square and afterwards, drawing several squares. We developed a basic guide to introduce the main elements of Scratch. The guide included how to move the image, to change colors, two versions of the square program and some analogous exercises. For instance, other exercises were to draw a hexagon, a triangle and a ladder. Those exercises were oriented to formalize and to understand the ideas of process and control. In a different way as Malan and Leitner's work [21], we concentrated in pseudo code and basic concepts; we did not include threads and events. We used only one sprite. We tried to focus on in the main elements. Another difference was the format of exercises proposed. Our exercises were clearly defined and proposed in order of incremental complexity. Malan and Leitner [21] proposals were more open.

The second week we proposed more "intellectual" exercises: the cat (one of the icons provided by Scratch) "thinks" numbers and wants to know the sum and the product of those numbers. We introduced variables

formally. After that, we calculate the mean of the numbers and the maximum. In the lectures, the exercises are solved by hand by each student and, after that, were discussed and coded in Scratch in the computer's class. In the lab session, each student programmed by him or herself the solutions and solved the additional exercises proposed. Some exercises were to calculate the sum of the odd numbers thought by the cat, to count how many numbers were higher than 10 and to show the digits of each number.

In our traditional courses –the control groups- the exercises proposed were the same, but they were only solved by hand.

In the third week, we introduced Java or Visual Basic depending on the course (university or vocational). When we presented the language, we made a parallelism between some constructions in Scratch and the selected language. For instance, Scratch's block "when green flag clicked", could be equivalent in Java with "*public static void main (String args[])*"; Scratch's *if* block matchs directly with the *if* in Java, the *repeat n times* corresponds with Java's *for*, and so on.

We included more exercises in Scratch to reinforce the main ideas of loop, conditions and variables. Some exercises in this week were similar to inform, for instance, if the number "37" was thought by the cat, or to count the number of strictly increasing sequences, and the mean of the numbers greater than 100. The absence of boolean variables in Scratch forced to present this concept as an integer flag.

In the fourth week, all the courses were the same, introducing objects and classes. The exercises were solved in Java or Visual Basic depending on the particular course.

To sum up, we based the courses on a different approach than Malan and Leitner [21]. They propose exercises with few requirements and few limits. They present a lot of concepts together and a small number of guides. We formulated detailed and specific exercises. Also, another difference with Malan and Leitner's work [21 Leitner] is that we want to test not only motivation effects but if our particular use of Scratch improves the results of the course and, or, affects in any way the retention rate. This is a long time vision and also a comparable metric with the normal courses.

EXPERIMENTATION AND RESULTS

In both cases (vocational and university courses), we developed two tests: one was applied at the beginning of the course and the other in the 4th week. The first test included some questions intended to know the programming background of the students, if any, and asked to simulate and to solve some exercises in pseudo code. Each student should "execute" the proposed programs and also develop simple programs. The second test included similar questions to check the progress in those areas. The questions were graded with ordinal scales. In each case, we have two groups ("Control" and "Scratch" group); there were about 15 students in each group. The students were randomly assigned to each group.

The independent variable in our study is the Scratch training. The dependent variable is the development of easy computer programs, particularly analyzed in relation to a) to understanding an algorithm, and b) to solve a particular problem. The samples were compared using the Mann-Whitney ($\alpha = 0.05, 0.10$) and the Sign tests. After checking the results, we did not detect any statistically significant difference between the groups neither the initial test nor the final test. Moreover, we did not detect differences between the students with some programming experience and the complete novices.

In addition to the test, we gave a questionnaire to the students in the 6th week intended to detect personal feelings about programming and capture reflections over their own learning process. In university courses, 91.67% (11/12) of the students who used Scratch and completed the questionnaire described learning programming in terms similar to "interesting", "easy" or "positive". In contrast, 60% (6/10) of the control group used the word "difficult", "not easy" or analogous concept; only 40% expressed a positive or encouraging view of learning programming (see Figure 2).

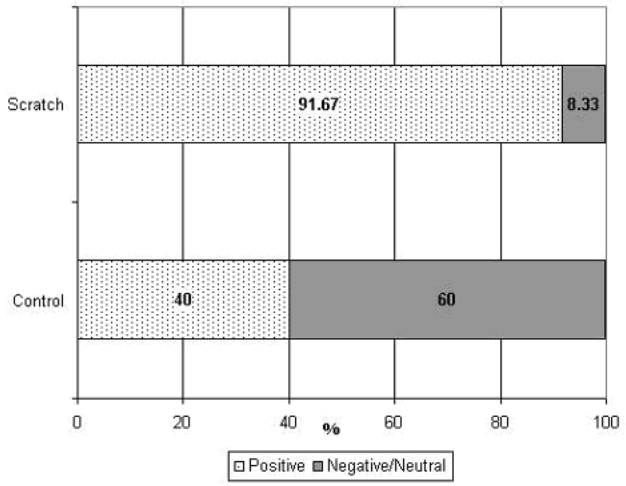

FIGURE 2
UNIVERSITY COURSES: PERCEPTION OF LEARNING PROGRAMMING: POSITIVE VS. NEGATIVE/NEUTRAL.

In vocational courses, the situation was similar. 14 of 16 students of the Scratch course described their learning as "motivating", "funny" or "easy". In the respective control group, only 2 students used those terms. The majority (8 out of 10 who answered the questions) referred as their learning as "difficult" or "normal" (see Figure 3).

In the courses there is a compulsory programming assignment in the 8th week. The task requires the development of a single object oriented program, with 3 classes. In those classes, the students must develop similar algorithms to the previously detailed in the course. We analyzed the results of this task.

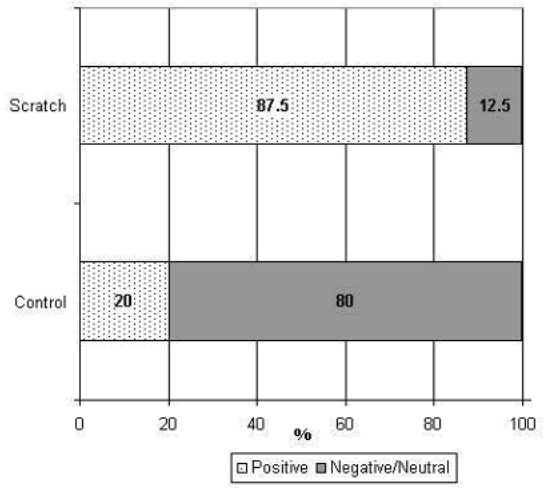

FIGURE 3
VOCATIONAL COURSES: PERCEPTION OF LEARNING PROGRAMMING: POSITIVE VS. NEGATIVE/NEUTRAL.

Considering the task results of university students, the numbers were similar in both groups (Scratch and control): the mean was 7.01 out of 10 points and standard deviation 1.73 in the Scratch course; mean 7.02 and standard deviation 0.69 in the control course (see Table II).

TABLE II
TASK RESULTS (UNIVERSITY)

	Mean	Standard Deviation
Control	7.01	1.73
Scratch	7.02	0.69

The task results of the vocational courses were little better in the control course (see Table III).

TABLE III
TASK RESULTS (VOCATIONAL)

	Mean	Standard Deviation
Control	6.55	2.81
Scratch	5.57	2.79

Also we checked the retention rate. In this work we consider the retention rate as the number of students that attended regularly the course at the 10th week over the number of enrolled students. Related to this rate, in both university groups the results were about 91-94% of students who continue in the course, (see Figure 4). In vocational courses there were also no differences. The retention was similar in both courses (80-83% of students continue in the course, see Figure 5).

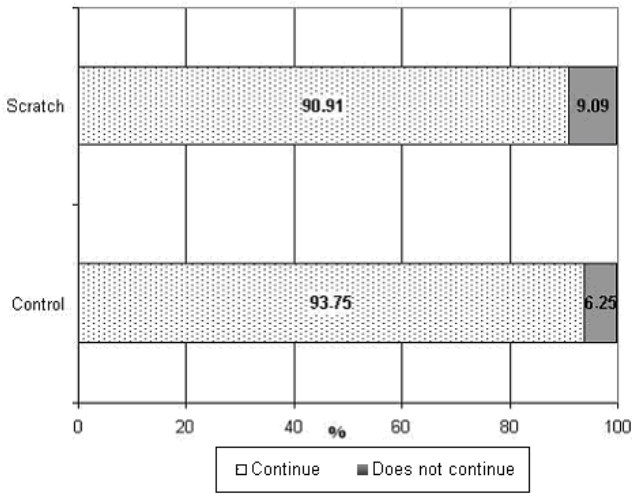

FIGURE 4
UNIVERSITY COURSES: RETENTION RATE

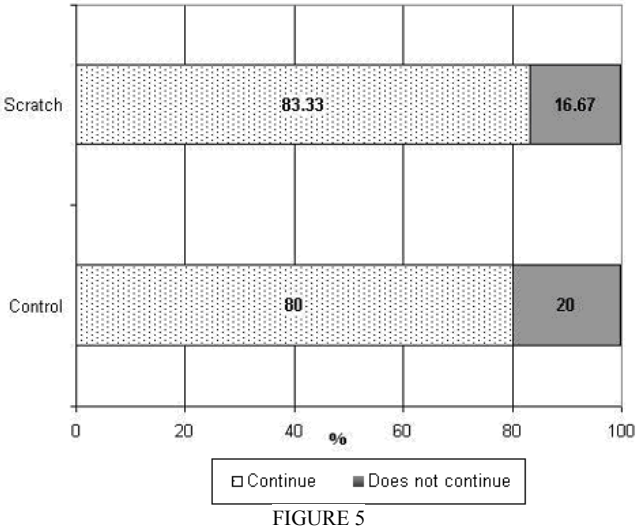

FIGURE 5
VOCATIONAL COURSES: RETENTION RATE

A topic that should be mentioned is the perspective of the teachers. All the teachers who participated in the experimentation (control and Scratch courses) have several years of experience in the CS1 course. A reflection made by the Scratch course's teachers is that they feel that it was easier to perceive those students who might have certain problems programming than in the normal courses. The students who used Scratch appeared to be more open to express orally their difficulties with the exercises. This may allow to detect early on those students who might have difficulties programming and to suggest them extra exercises.

OPPORTUNITIES TO IMPROVE SCRATCH

We found some opportunities to improve Scratch in order to simplify its use in CS1. We used the version available in March 2007. The lack of boolean variables can be supplied by integer variables acting as a flag. This is a

minor problem. The input of user data is a limitation, it would be useful some kind of interactive input. We can use random generator and suppose that the program "thinks" a number instead of asking for one to the user.

With the aim to establish a correspondence between the concept of "message passing" and object oriented programming, we suggest that the broadcasting scheme could be directed to a specific item, not only general. In this way, we could send a message to a particular object or element.

Another topic is that the Spanish translation is confusing in some specific topics. For instance, if we have a variable "suma", the expression "change suma by 30", translated as "*cambiar suma por 30*", is not clear. If *suma* equals 10, that expression changed *suma* to 40 but, in Spanish, one is induced to think that the final value is 30, not 40. It would be better the expression "*modificar suma en 30"*. In the version referred, when the user changed language to Spanish, the buttons (new, open, save, etc.) and tabs remained in English. The dialogs and the help were also in English. In the new version (February 2008), this point was partially adjusted (the user could change the texts, there is a configuration file) but the help is still in English.

CONCLUSIONS

The use of Scratch in the very beginning of the CS1 course promoted a high level of motivation, thus a positive perception of learning programming, but we did not found any measurable improvement of the results obtained by students who used the tool compared with the normal course. Moreover, we did not find any statistically significant differences in retention rates. A possible interpretation is that the students of the Scratch course have two "jumps": first they must familiarize with Scratch and then with Java or Visual Basic. The students of the normal course only have one "jump". The impact of including a different tool like Scratch before Java or Visual Basic should be carefully evaluated, taking into account the absence of "tangible" benefits. It takes some time to install and familiarize oneself with Scratch, and few weeks after that the student must change to another environment in order to develop real programs. The initial improvement of motivation may be affected in some way by this change.

ACKNOWLEDGMENT

We would like to thank to Carlos Berrutti, Adriana Cabella, Marta Castro, Andrés de Sosa, Matías Núñez and Luisa Pereira-Hors from ORT Uruguay University for their collaboration with the experimentation.

REFERENCES

[1] Pendergast, M. "Teaching Introductory Programming to IS Students: Java Problems and Pitfalls", *Journal of Information Technology Education*, Vol 5, 2006,491-515

[2] Scratch. http://scratch.mit.edu/. last accessed February 18th, 2008

[3] Robins, A., Rountree, J. and Rountree, N., "Learning and teaching programming: a review and discussion", *Computer Science Education,* Vol 13, No 2, 2003, 137-172

[4] Huet, I., Rocha, O., Tavares, J. and Weir, G., "New challenges in teaching introductory programming courses: a case study", *Proc. of the 34th. ASEE/IEEE Frontiers in Education Conference*, Savannah, Georgia, USA, 2004, T2H5-T2H9

[5] Esteves, M. and Mendes, A., "A simulation tool to help learning of object oriented programming basics", *Proc. of the 34th ASEE/IEEE Frontiers in Education Conference*, Savannah, Georgia, USA, 2004, F4C7-F4C12

[6] Bailey, M., "IronCode: Think-Twice, Code-Once Programming", *Proc. of the 36th SIGCSE technical symposium on Computer Science Education*, Missouri, USA, 2005, 181-185

[7] Carlisle, M., Wilson, T., Humphries, J. and Hadfield, S., "RAPTOR: a visual programming environment for teaching algorithmic problem solving", *Proc. of SIGCSE 05*, USA, 2005

[8] Cooper, S., Dann, W., and Pausch, R., "Teaching Objects-first In Introductory Computer Science", *Proc. of the 34th SIGCSE technical symposium on Computer science education*, Nevada, USA, 2003, 191-195

[9] Bayliss, J. and Strout, S., "Games as a "flavor" of CS1", *Proc. of the 37th SIGCSE technical symposium on Computer Science Education,* Texas, USA, 2006

[10] Sung, K., Panitz, M., Rosenberg, R. and Anderson, R., "Assessing Game-Themed Programming Assignments for CS1/CS2 courses", *Proc. of Game Development in Computer Science Education* (*GDCSE 08*), USA, 2008

[11] Bierre, K. and Phelps, A., "The use of MUPPETS in an Introductory Java Programming Course", *SIGITE 04*, USA, 2004

[12] Ahern, T., "Work in Progress: Effect of instructional design and pair programming on student performance in an introductory programming course", *Proc. of the 35th ASEE/IEEE Frontiers in Education Conference,* USA, 2005, F3E11- F3E12

[13] Hanks, B., "Student performance in CS1 with distributed pair programming", *Proc. of the Innovation and Technology in Computer Science Education Conference (ITiCSE 2005),* Portugal, 2005

[14] Andrianoff, S. and Levine, D., "Role playing in an object-oriented world", *Proc. of SIGCSE'02,* USA, 2002

[15] Briggs, T. and Girard, D., "Tools and techniques for test-driven learning in CS1", *Journal of Computing Sciences in Colleges***,** Vol 22, No. 3, January 2007

[16] Janzen, D. and Saiedian, H., "Test-driven learning: intrinsic integration of testing into the CS/SE curriculum", *Proc. of SIGCSE 2006,* USA, 2006

[17] Börstler, J. and Schulte, C., "Teaching Object Oriented Modelling with CRC Cards and Roleplaying Games", *Proc.of WCCE 2005*, Cape Town, South Africa, 2005

[18] Bruce, K., "Controversy on how to teach CS1: a discussion on the SIGCSE-members Mailing List", *Inroads- The SIGCSE Bulletin*, Vol 36, N. 4, 2004, 29-35

[19] Sanders, D. and Dorn, B., "Jeroo: A Tool for Teaching Object-Oriented Programming", *ACM SIGCSE Bulletin*, 35(1), January 2003, 201-204

[20] Project Greenfoot,http://www.greenfoot.org/ (last accessed February 20th, 2008)

[21] Malan, D. and Leitner, H., "Scratch for budding computer scientists", *Proc. of SIGCSE 2007,* USA, 2007