Learning Computer Programming: Start from Scratch!

Afroditi Michailidi
6th EPAL (Vocationa/Technical School) of Heraclion, Crete, Greece
amichail@sch.gr

Abstract. *The Greek Technical High School curriculum includes a senior-year course in Computer Programming which most of the students, with no prior schooling, confuse with basic computer skills.*

To motivate them, we used Scratch, a programming tool that encompasses modularity, object-oriented-programming and multi-threading.

Student response was highly encouraging. All were engaged in the activities and many continued on their own.

Scratch enabled the students to focus on the method rather than the syntax. As a result students earned a better look and a deeper understanding of the basics of computer programming.

Keywords. Scratch, Computer Programming, High School.

1. Introduction

The Greek Technical High School curriculum includes a senior-year course in Computer Programming. Basics are taught using "pencil-and-paper", thus maintaining the strict rigidity of the discipline. Later, students get to use pseudocode, still on paper, with no tangible results available. As an introductory programming language, Pascal is taught later in the semester. None of the students have prior computer programming schooling and most confuse basic computer skills (e.g. surfing the internet) with it. Most book examples involve math equations, thus adding to the frustration - especially of female students. Worse, the curriculum simultaneously includes a Visual Programming course and a Databases course in the same year thus endlessly confusing students between flow charts, pseudocode, Pascal, Visual Basic, SQL etc.

To face these challenges, we used Scratch, an application by the MIT Media Lab, available as Free and Open Source Software. Scratch shows the results of your code immediately. Scratch comes in a variety of languages and has a user-friendly intuitive interface that encourages experimenting. Although tailored to younger children, it is a powerful programming tool that encompasses modularity, object oriented programming and multi-threading. There is also a thriving online community with the motto "Imagine–Program–Share".

We used Scratch to teach conditional statements, both simple and nested. Instead of the usual "if-then-else", students were encouraged to create an interactive game with two figures chasing each other. Moving the figures posed problems that needed solving, including setting conditions and constraints.

In the next sections we introduce the Scratch programming environment, describe in detail the methodology and the working examples used and give a discussion on the usefulness of the approach.

2. The Scratch Project

Scratch is a fairly new educational programming environment, publicly launched in 2007 [1]. It was created by the Lifelong Kindergarten Research Group of the MIT Media Lab Team in an attempt to make programming accessible and easy to learn for everyone.

To this end the user interface is divided into three main panes: on the left is the blocks palette, in the middle the current sprite info and scripting area, and on the right the stage and sprite list.

Sprites are images that can be put on the screen. Users can draw them in a built-in version of painter, choose from a wide variety of supplied images or import their own. On opening a new project a trademark friendly smiling cat is the default sprite as seen in Fig. 1.

Scratch uses an intuitive programming block system to help novices get a head start as also illustrated in Fig. 1. The basic programming blocks are organized and colour coded according to function and much like traditional puzzle

Afroditi Michailidi (2010). Learning Computer Programming: Start from Scratch! M. Kalogiannakis, D. Stavrou & P. Michaelidis (Eds.) *Proceedings of the 7th International Conference on Hands-on Science.* 25-31 July 2010, Rethymno-Crete, pp. 231 – 234
http://www.clab.edc.uoc.gr/HSci2010

pieces have connectors that suggest how they should be put together. The blocks are shaped in such a way that only syntactically valid combinations are possible.

Users "write code" simply by moving programming blocks into the scripting area and putting them together. They can program different sprites by selecting them and can easily copy "code" simply by dragging-and-dropping it onto a different sprite.

Figure 1. Scratch programming blocks

Once a programming stack of blocks is built, it can be executed just by double-clicking on it. Multiple programming blocks can be built at the same time thus providing parallel task execution. The emphasis is on a bottom-up approach and on iterative incremental design.

Another concept high on the minds of the creators was collaboration and sharing. To this end the user interface and Scratch programming blocks have translations in over 40 languages. This approach allows users from different countries to build Scratch projects and then share them, each viewing the Scratch programming blocks in their own language. The Scratch Web Site is widely successful with, on last count, over a million projects uploaded from around the world [2]. The projects published are licensed under a Creative Commons attribution. There is also ScratchEd, a community for educators.

3. Visual vs Structured Programming

The author teaches senior-year students who have chosen the Technical High School computer related specialty of System, Network and Software Support. These students have basic computer skills literacy and have taken basic computer courses in their previous year, mostly focused on digital design, hardware and networking. As mentioned in the introduction the senior-year curriculum includes two obligatory programming classes "Visual Programming" and "Structured (Procedural) Programming" in the same year [3].

Each class has its own course material introducing different concepts in different times. This results in re-iteration and student confusion. In the previous year, the teachers of both classes in coordination tried a restructuring of the course material so that key concepts were introduced at the same time.

Teacher coordination even proceeded to the extent of handing out and working with the same set of example problems. As students get more fluent with programming in the second semester, they get to program the same example both in Visual Basic and in Pascal.

This approach had mixed results. Seemingly at first, students would benefit from two different approaches as they can get a glimpse of the bigger picture. Thus variables in Pascal and variables in Visual Basic are two aspects of the same thing. But mastering the concept of variables alone is hard enough. With no prior programming background the students become confused as parallel teaching progresses on to the syntax and program coding. This leads to general misconceptions and common mistakes (e.g. no end-if in Pascal as opposed to Visual Basic, begin-end necessary for statement blocks in Pascal but not in Visual Basic etc).

.

4. Why choose Scratch?

What students were missing despite the teachers' best effort was the bigger picture. Students need to realize that both structured and visual programming applies the same basic programming principles. An "all programming is equal" approach seemed fit and to that end the common set of problems was devised.

However since the "Structured Programming" class is the one examined nationally for achieving University admittance, it is crucial to follow its own strict schedule including all the examples in the book.

These book examples are filled of math problems (finding grade averages, summing up salaries, calculating fares of car tolls etc). They make interesting programming tests in the academic sense of the word but are not nearly as good enough in motivating teenage students to care about the outcome of their programs.

Worse yet, trying to learn well two new variations of the same thing is much harder than trying to learn only one. Students were bogged

down by the syntax errors as they tried to switch between two or more programming languages.

In contrast, the Scratch programming environment highlights the bigger picture by not focusing on syntax at all. It also makes for highly interactive and personalized programming examples. On top of that its build-in object oriented programming is intuitive in a way that Visual Basic controls and code are not. Plus it encourages step by step programming and allows the user to see results of their work immediately without compiling or switching between user and programmer interfaces..

For all the above reasons, the author decided to use Scratch in the "Visual Programming" class.

5. Methodology and examples used

Scratch was used to teach conditional statements. Rather than give the same old problem of characterizing a student based on their grades (if grade>10 then PASS else FAIL) students were asked to program an interactive game with two characters: a cat-and-mouse chase.

The students were first shown how the game should work once all the programming had finished. They were then asked to figure out, write down and program the necessary constraints for the game to work as expected.

Taking advantage of Scratch's inherent ability for incremental design, rather than the teacher handing out all the constraints at the start, the approach followed was a step by step one with the students coming up with the necessary conditions each time.

Students were first introduced to the Scratch interface. After a quick tour of the controls they were asked to decide which programming block would move the cat to the right and test it out. They were then asked to move the cat continuously to the right, again choosing the appropriate programming block.

The first constraint was set as our cat "hit the wall" of the display and had to be made to turn around and move in the other direction. This set the first conditional statement (*if on edge, bounce*).

Next we wanted our cat to seem like really walking, not just moving. That led students to discover how to switch sprite costumes.

Now was the time to bring on the second sprite. Students were asked to place an icon of a mouse on the screen and make it move up and down this time.

The last crucial step was defining when the cat catches the mouse. Students again had to construct a conditional statement, this time using the *touching* block.

Having mastered the basics, the students were then asked to redesign their project so that the mouse icon movement could be controlled by the user. This led them to discover the *when … key pressed* block and incorporate its implementation in their project.

The next day the game was enhanced by adding variables and more conditions. Specifically the students were asked to place bits of cheese along the screen and keep score of the number of cheese pieces that the mouse managed to eat (subsequently making them disappear from the screen). As an extra penalty, if the cat catches the mouse the score counter should be set to zero.

6. Discussion

The above examples were taught in two consecutive days for a total of four teaching hours (two two-hour periods). They were taught on two separate occasions to classrooms of 12 and 18 students respectively.

Up to that point the students had experience in designing simple Visual Basic forms and writing simple code on Command Buttons- e.g. image1.Visible=False on Command1_Click().

Apart from the requirements, students were not given specific guidelines as to which command to use. Neither were they given a tutorial on the use of Scratch. They were however given ample time to experiment both during programming and after completing their project.

All of the students got actively involved, tried to solve the problem given and make the game work.

Although they had never come in contact with Scratch before, the students all managed to complete both the assignments. Indeed most of them improvised and added more blocks and more sprites.

Additions included having two cats or two mice running in different directions; having numerous animal sprites running up and down the screen; upload their own photos instead of cat and mouse; making the cat speak or the mouse scream upon colliding; using sound effects for

the cheese-eating and mouse-catching; changing the background.

One went as far as using *wait … secs* and developed an introductory animated story which he then proudly exhibited to the rest of the class.

None of these additions were prompted explicitly by the teacher; the guideline given was "seek and you shall find".

Students of the first class told students of the second class about these activities. As a result the second class was expecting "that cool definitely not VB thing".

About half of the students asked to have a copy of the program for use at home and were surprised that they had to pay no licence fee for it. This led to a discussion on Free and Open Source Software and its communities. They were also all directed to the Scratch Web Site for further reference.

Both classes wanted to do more projects in Scratch so we dedicated an extra two-hour period to design a game of Pong (racket moved by the user; ball bouncing off in random direction each time it hit the racket, score counting and end game if ball is dropped).

7. Conclusions

Scratch definitely was a success with students but did it enable them to better grasp the fundamentals of programming?

Upon reverting to the usual curriculum we continued with several if-then-else problems with most of the students grasping the required constraints correctly without much difficulty (note here that they were also of interest to students e.g. finding body mass index given weight and height).

Although developed with a younger age group target in mind, its inviting appearance and its diverse possibilities made programming with the Scratch programming environment a compelling task for the students.

Scratch also motivated the students to think about the steps necessary for solving a given problem rather than worrying about fitting the commands in the right order.

Coupled with examples suited to students' interest and a sound pedagogical approach Scratch can be a powerful tool when it comes to teaching computer programming.

8. Acknowledgements

The author would like to thank the Greek Logo Community for introducing her to the wonderful world of Scratch.

And thanks as always go to my students who never cease to amaze me.

9. References (and Notes)

[1] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y., (2009). Scratch: Programming for All. Communications of the ACM November 2009; 52(11): 60-67.

[2] Scratch Web Site. http://scratch.mit.edu/ [visited 25-June-2010]

[3] Pedagocical Institute: Information and Communication Technology Curriculum. http://www.pi-schools.gr/content/index.php?lesson_id=1 [visited 25-June-2010]