Olympiads in Informatics, 2009, Vol. 3, 17–25

Taking Kids into Programming (Contests) with Scratch

Abdulrahman IDLBI
Syrian Olympiad in Informatics, Syrian Computer Society
e-mail: adlogi@gmail.com

Abstract. Since launching the Syrian Olympiad in Informatics (SOI) five years ago, encouraging children to participate in the contest has been a challenging task, a common problem in many places around the world. Students, and many educators as well, see programming as a tough subject to learn. In addition, the style of IOI tasks is generally considered unattractive.

We started overcoming those obstacles through using Scratch, a graphical programming language developed at the MIT Media Lab. Scratch allows kids to start learning programming concentrating on the concepts rather than the syntax, while providing them with the ability to construct diverse projects that are attractive and meaningful to them. Scratch also allows examining children's programming skills against interesting tasks, and gives them the opportunity to move more smoothly into learning a traditional language like C++ and other computer science topics.

Key words: introducing programming to children, Scratch contests, Syrian Olympiad in informatics.

1. Introduction

Through the past few years, studies have warned about the decreasing interest among students to study computing-related fields, as they consider them tough and somehow uninteresting. Similarly, people organizing activities related to computer science contests, with challenges in programming and algorithms, constantly complain about the weak motive among youth to participate in them, especially when compared to the willingness to participate in robotics or IT contests for example. This problem becomes more obvious as the targeted audience gets younger. Taking the previous two examples of more acceptable contests among youth may give us a clue to the potential reasons of the problem: (1) sole programming is not as fantastic as building robots, (2) and children are not exposed to it early in their daily life (nor most of the adults surrounding them) as they are to other IT topics. Having these two points in mind would lead us to a solution.

When the Syrian Olympiad in Informatics started five years ago, the contest was divided into several divisions depending on age. The oldest division had preparation requirements similar to those of the IOI, and the national team for the IOI was chosen from its participants; but as a division got younger it contained less algorithms and programming in favor of more IT tasks (tasks related to knowledge of OS and desktop applications). While children were more familiar with IT skills than programming skills, testing IT skills or promoting them did not lead to a better preparation for the IOI in the older

division, nor did it help in discovering and preparing potential young computer scientists. In addition, the SOI organizers sought only students who were already considered distinguished in school mathematics or IT. Not approaching a wider audience of children meant two things: first, there was no way to discover hidden talents in computer science; second, without more children practicing programming the general public was not be able to recognize its importance.

To overcome the popularity problem facing our contest we changed our strategy. We decided to aim at all children and spread a culture of programming among them, which would provide a better opportunity of selecting future computer scientists. This change needed a powerful tool to be placed in the hands of children, and it was *Scratch*, a graphical programming language developed at the MIT Media Lab. Scratch allows kids to start learning programming concentrating on the concepts rather than the syntax, while enabling them to work on diverse projects that are attractive and meaningful to them. While recently some researchers used Scratch to introduce programming to university students and as a gateway to an advanced language like Java (Malan and Leitner, 2007), we argue that Scratch can be used in a similar way with younger children, both to prepare them to learn a language like C++ and to be used in contests with tasks similar to IOI tasks. Children in the context of this paper are mainly those between 7 and 15 years old.

2. What is Wrong with Programming?

The awareness and understanding of parents and educators have a major role in making any extracurricular activity for children succeed, but they are not the most important factors. The most important one is how fun and attractive children find the topic of the activity, and how closely it relates to them.

When it comes to programming, the languages used in the IOI (and many other typical languages) look like Greek at a first glance, and for many students they remain like that for a long time. Even the common simple start with a "Hello World" program raises several questions that cannot be answered in the first few sessions of a programming course. After that not-so-interesting start, children have to concentrate on remembering syntax details, such as semicolons and parentheses, so the compiler does not get angry at them, instead of concentrating on learning the programming concepts (e.g., variables, conditions, loops, etc.) in addition to logic. It turns out that "students must become masters of syntax before solvers of problems" (Malan and Leitner, 2007).

Even more, when children come to a programming course, they come with broad expectations and questions like "When are we going to make our first game (or virus)?", and those who are patient enough to learn the basics of the language would soon get frustrated when they discover that they cannot do more than simple operations on meaningless data sets using a text-based interface.

To set it in a single sentence: "Computer programming has been introduced using programming languages that are difficult to use, with proposed activities that do not connect with young people's interests and in contexts where no one has enough expertise

to provide guidance" (Resnick *et al.*, 2003). These reasons make programming with the commonly-used text-based languages inappropriate for introduction to a wide audience of children, making it hard to discover potential young programmers early.

To solve this problem while preparing for SOI, we used *Scratch* to introduce programming through the training of different divisions, and as a part of the contests for the children under 15 years old.

3. Different Programming Experience with Scratch

Scratch is a new graphical programming language that makes it easy for children to create their own interactive stories, animations, games, and arts. Coding in Scratch is much easier than in traditional programming languages: to create a script, you simply snap together graphical blocks, much like LEGO bricks or puzzle pieces. Scratch is designed to help young people (ages 8 and up) to develop essential skills such as creative thinking, clear communication, systematic analysis, effective collaboration, iterative design, and continuous learning (Lifelong Kindergarten, MIT Media Lab, n.d.).

Scratch follows the principles of making a successful software tool for kids (Maloney *et al.*, 2004). Those principles include:

- making the value and possibilities of the tool clear from the beginning;
- respecting children interests;
- the ability to create complete meaningful projects that can be shown to others;
- supporting a wide range of different types of activities, giving the ability to aim kids with different backgrounds and interests;
- the ability to get started quickly and without external help;
- the ability to learn additional features over time, and use the tool in more complex ways.

For that, Scratch is described as offering a low floor (easy to get started), high ceiling (ability to create complex projects), and wide walls (support for a wide diversity of projects) (Lifelong Kindergarten, MIT Media Lab, n.d.).

With these principles in mind, Scratch was designed with core features that include (Resnick *et al.*, 2003):

- *Building-block programming*: Programming by snapping together graphical blocks that fit in only syntactically-correct ways. This approach eliminates syntax errors (which have proven to be a major obstacle for learning text-based programming languages), allowing youth to focus on the problems they want to solve, not the mechanics of programming.
- *Programmable manipulation of rich media*: Scratch programs manipulate images, animations, movies, and sound; which offers programming activities resonant with youth interests, providing them with an opportunity to start from their own comfort zone, but then reach out to learn new things.
- *Support for multiple languages*: The possibility of translating Scratch interface to many languages (Scratch is now available in more than 40 languages) and switching dynamically among them allows children to work and think with the language

most comfortable to them, and then to talk about the knowledge they are building more creatively, which develops a sense of mastership of the recently-gained knowledge. Through our work in SOI, we could decrease the minimum grade for accepting students in Scratch courses from the 4th grade to the 2nd grade after translating Scratch into Arabic. We could also find effective 3rd-grade Scratch programmers after the translation, compared to not having any below the 6th grade before the translation.

After getting it for free, students can start programming at once by dragging blocks from eight categories on the *blocks palette* and snapping them together (only if they are syntactically correct) on the *scripts area.* The result of running programs is shown immediately on the *stage* where sprites (programmable objects) interact with each other (Fig. 1). The available blocks support various programming concepts such as loops, conditions, Boolean expressions, variables and lists (arrays) in addition to parallel execution and events (Fig. 2).

Scratch is not the first programming language intended to be used for introducing programming, and is much inspired by the ideas behind Logo. It is also not the only one today with languages like NetLogo, StarLogo and Alice. However, Scratch seems to have several advantages over others which have for example a too restricted virtual world or a high learning curve (Malan and Leitner, 2007).

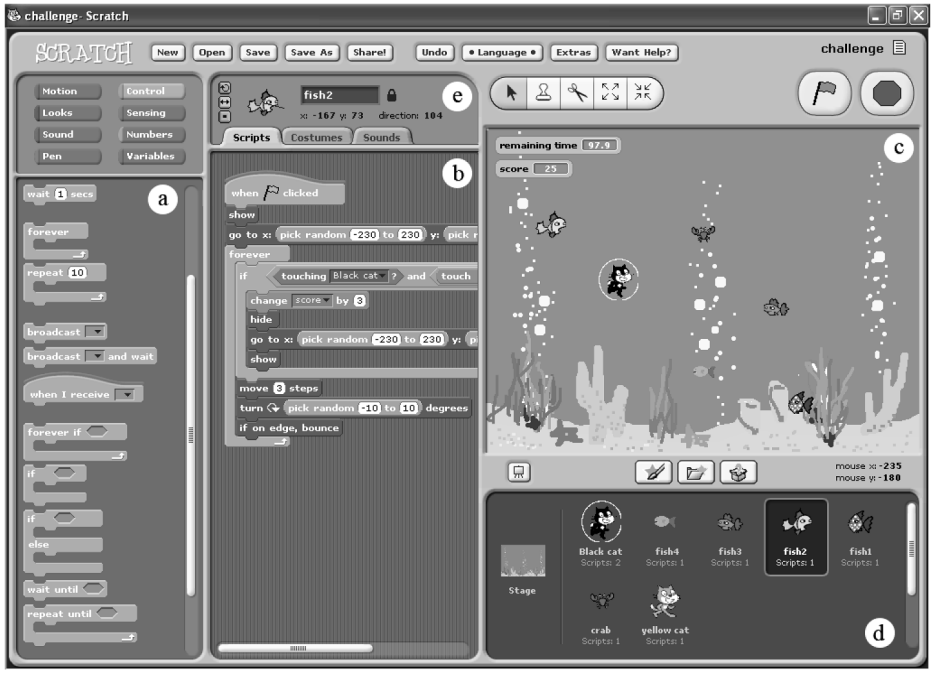


Fig. 1. Scratch interface (version 1.3.1) with the blocks palette (a), the scripts area (b), the stage (c), the sprites list – the objects to be programmed (d), and the current sprite's information (e). The shown project was one of the 1st division's tasks in SOI 2008.

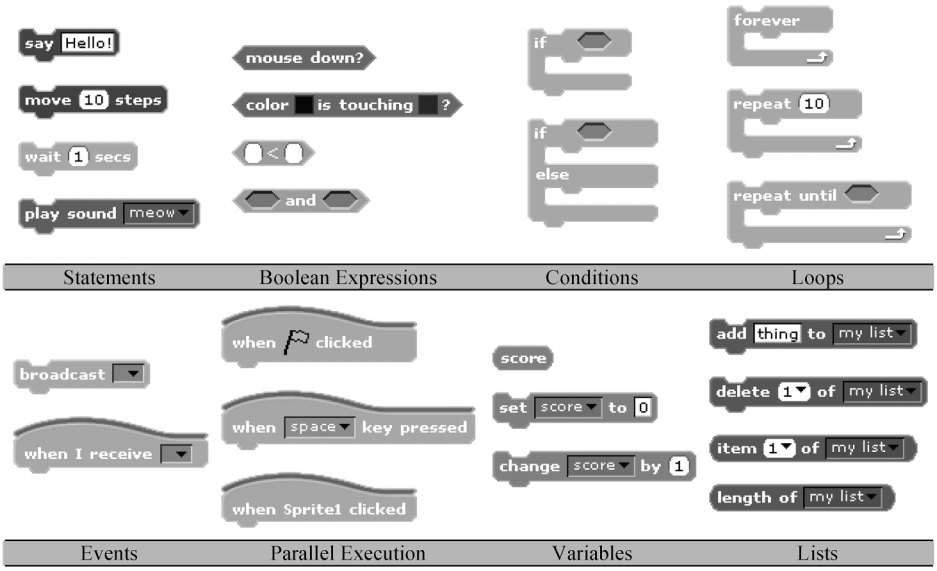


Fig. 2. Some of Scratch blocks showing supported programming concepts.

4. SOI Structure and the Utilization of Scratch

SOI consists today of three divisions depending on age: the 1st division for students under 12, the 2nd for students under 15, and the 3rd for students under 20. Scratch is used in SOI in two ways: to introduce programming before moving to C++ in all divisions, and as a part of the contest itself for the two younger divisions.

Students in the two younger divisions start their training by following a 10-session course which is open to all interested children under 15. In the first session, children learn how to move an object on the screen, to draw while moving, and to create simple loops. They do that while learning the concepts of what Seymour Papert calls “turtle geometry”, making use of children’s knowledge about their body and how they move, to draw basic figures and combine them together (e.g., drawing a square and a triangle to create a house). Through this exercise they obtain their first debugging experience (Papert, 1980). Children end this session with experiments on drawing more sophisticated geometric shapes, with some of them using nested loops (Fig. 3).

Next, children are exposed every one or two sessions to a new project. Each project has certain programming and thinking skills to be learned, and the instructor’s duty is to point out these skills when they are needed. The sessions are titled “Hunting the Parrot” or “Racing Game” instead of “Dealing with Events” or “Creating Variables”, allowing the children to learn serious issues through “hard fun”. That is providing learners with challenging activities which are deeply connected with their interests and passions (Papert, 1993). This method relieves students of feeling strangers to programming or its related science topics.

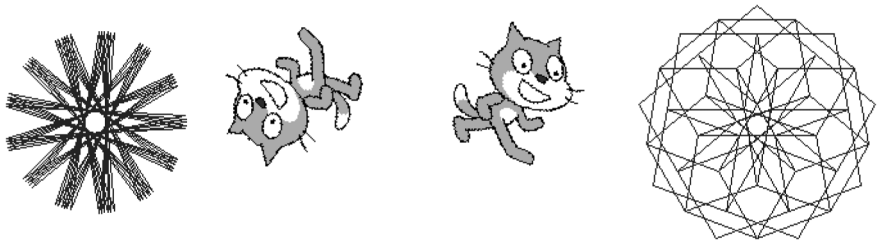


Fig. 3. Some of children creations by the end of their first session with Scratch.

Using Scratch at this stage gives children the potential to show their talents regarding computing. While instructors observe that some students are more interested in using Scratch as a design tool to create interactive media for example, others show interest in the programming process itself by using complicated and advanced programming structures and controls (e.g., using nested loops and conditions, familiarity with variables and using them in unexpected ways, etc.). By the end of the course, almost everyone has enjoyed working with Scratch, and the instructor can identify who enjoyed Scratch as a programming tool and nominate them to the next advanced course.

At the advanced course, children learn using more sophisticated techniques in Scratch, as searching and sorting lists, and make more extensive use of Boolean expressions. After that things get more formal with learning simplifying Boolean functions, expressing them as logical gates, and using truth tables. Then, programming with C++ is introduced depending on the child's previous knowledge of programming concepts using Scratch. While 1st division contestants are only required to comprehend simple programs and guess the outputs resulting from various inputs (with 15% of the total points of the contest), 2nd division contestants have additionally to complete IOI-style tasks (with all C++ tasks having 40% of the total points). Besides learning programming in C++, students learn more computer science skills and concepts such as estimating complexity and recursion, and are introduced to several basic data structures. In addition to testing those aspects of computer science implicitly through Scratch and C++ tasks, they are also theoretically tested through tasks similar to those used in the ACSL competitions (American Computer Science League, n.d.). The theoretical section takes 20% of the total points in each of the two younger divisions.

Scratch Tasks: In the contest, Scratch tasks have a considerable weight (65% of the total points for the 1st division, and 40% for the 2nd division). Contestants in both divisions face several Scratch tasks with various difficulties, and each is usually a game to be programmed. Contestants are provided with the stage and the sprites (the objects to be programmed) ready to be used so they do not waste time on painting the characters and objects of the game, and they are asked to add the behaviors (i.e., constructing the scripts) to accomplish specific missions. While a task description presents every detail about the required mission, students have also access to a working model of the mission as a Java applet, and by comparing it to their implementations they can make sure that they are on the right path.

Having a racing game for example as a task, kids are provided with the images of the car, the race route, and the obstacles; and with an explanation of the race rules: how the car behaves when driving on/outside the specified route or when it goes through obstacles, how it accelerates, how the score is calculated, and when the game ends. Contestants have to implement the scripts for doing that, with each part of the mission having a specific amount of points.

Submitted solutions are graded manually. Two graders check together each project and compare each partial behavior with the matching one from the working model. When the apparent behaviors are similar the corresponding amount of points is granted, otherwise, graders have to look for the scripts controlling the investigated behavior and estimate how far it was from attaining the desired results, and assign points according to their judgment.

Contestants in the 3rd division are prepared with IOI-requirements, and the national team is chosen from them. While they do not have Scratch tasks in their contest, they use Scratch through the preparation process which goes as follows: after selecting prospective students depending on their school records (especially in math) they are introduced to programming using Scratch for three or four sessions. At the beginning they learn fundamental control blocks (representing loops and conditions), and start then implementing a couple of projects. After being exposed to programming concepts in Scratch, they start learning C++ and other IOI-requirements. Here too, Scratch plays an important role in attracting teenagers to programming and helping instructors to distinguish prospective programmers through the way they use Scratch in.

5. Results: SOI, Scratch and the Community

As mentioned earlier, we decided to change to strategy in the SOI to aim at all children and give them the opportunity to get to know about programming in an interesting context, and Scratch was the right tool for that. Surveying students' opinions from various introductory courses showed that more than 90% enjoyed working with Scratch, though some said they felt board at some point when complicated concepts had to be explained by the instructor.

An interesting point was that about 60% of the surveyed students from both courses in the two younger divisions said they enjoyed more working with others, and they would prefer a contest where they solve tasks as a team rather than individuals. Most of the remaining 40% of the students were reported as male students.

As regards the general public, with the introduction of Scratch the national contest received more interest, with many schools asking us to train their students or to organize workshops for their instructors on preparing to the contest. Many university students were also attracted to the idea of teaching young children stuff they had not known about themselves before their university-level education.

The most important result was the change of SOI's contribution to the society. SOI is no longer a mere contest to recognize young students who are the most talented in

computer science. We think now about programming from a different perspective, an educational creative one: to help children develop themselves as creative thinkers.

When we talk to educators or parents we tell them that most students who come to SOI would not grow up to become professional programmers, but programming would still be important for everyone: it would allow them to express themselves more creatively and perfectly, help them to develop their logical thinking, and facilitate understanding the new technologies they are facing everywhere in their daily life. In other words, "the continual use of abstract thinking in programming can guide and discipline one's approach to problems in a way that has value well beyond the information technology-programming setting. In essence, programming becomes a laboratory for discussing and developing valuable life skills, as well as one element of the foundation for learning about other subjects" (National Research Council, 1999). This enhancement in role of the national contest would not have taken place without having a new tool that represents this philosophy, and this tool is Scratch.

Acknowledgements

I extend my thanks to Scratch team at the MIT Media Lab for their wonderful work. I am so grateful to my colleagues from the Computer & Automation Engineering Department at Damascus University who supported the adoption of Scratch in SOI. In particular, I would like to thank Beshr Al Nahas, Boushra Jbr, Waed Khwiess, Maya Taki and Kusay Tomeh for inspiring discussions and sharing results from their extensive experience with children and Scratch. I also thank my dear friend, Ahmad Baghdadi, for supporting me during this work and reviewing this paper.

References

American Computer Science League (n.d.). *Sample Problems*. `http://www.acsl.org/samples.htm`

Lifelong Kindergarten, MIT Media Lab. (n.d.). *About Scratch*. `http://info.scratch.mit.edu/About_Scratch`

Malan, D.J. and Leitner, H.H. (2007). Scratch for budding computer scientists. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*. ACM, 223–227.

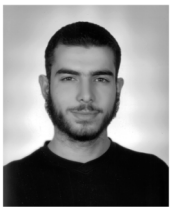
Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B. and Resnick, M. (2004). Scratch: A sneak preview. In *Proceedings of the Second International Conference on Creating, Connecting and Collaborating through Computing*. IEEE Computer Society, 104–109.

National Research Council (1999). *Being Fluent with Information Technology.* National Academies Press, Washington, DC.

Papert, S. (1980). *Mindstorms*: *Children, Computers, and Powerful Ideas.* Basic Books, Inc. New York, NY.

Papert, S. (1993). *The Children's Machine: Rethinking School in the Age of the Computer.* Basic Books, Inc. New York, NY.

Resnick, M., Kafai, Y. and Maeda, J. (2003). *A Networked, Media-Rich Programming Environment to Enhance Technological Fluency at After-School Centers in Economically-Disadvantaged Communities.* Proposal to National Science Foundation.



A. Idlbi is a fresh graduate from the Computer & Automation Engineering Department at Damascus University, and has been a scientific coordinator of Syrian Olympiad in Informatics since 2006. He was also the deputy leader of the Syrian team in IOI 2007. After participating in IOI 2004 and 2005, he has worked on introducing programming to the youth. His interests include promoting usage of new technologies to provide children with better learning opportunities.