

Analyze your Scratch projects with Dr. Scratch and assess your Computational Thinking skills

Jesús Moreno-León, jesus.moreno@programamos.es
Programamos.es, Sevilla, Spain

Gregorio Robles, grex@gsync.urjc.es
Universidad Rey Juan Carlos, Madrid, Spain

Abstract

In this paper we present the feature of Dr. Scratch that allows to automatically assessing the Computational Thinking skills of Scratch projects. The paper reviews similar initiatives, like Hairball, and investigates the literature with proposals for assessment of Scratch projects that we have studied and remixed in order to develop the Computational Thinking analysis. Then it introduces the various aspects that Dr. Scratch takes into consideration to compute a Computational Thinking score for a Scratch project and presents some preliminary findings of the analysis of over 100 investigated Scratch projects. Finally, future directions and limitations are presented and discussed.

Keywords

Computational thinking, learning, coding, Scratch

Introduction

Computational Thinking (CT) was defined by Wing as a skill that “involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science” (Wing, 2006). In the last years, governments and educational institutions around the world are trying to include the development of this competence in schools (European Schoolnet, 2014). In this regard, Lye and Koh, in their 'Review on teaching and learning of computational thinking through programming' (Lye, 2014), show that programming is a key instrument to develop this skill.

However, as explained in Section 2, assessing the development of CT is not a trivial issue, and several authors, like Resnick and Brennan, have proposed different approaches and frameworks to try to address the evaluation of this competence (Brennan, 2012). New tools are being developed to assist educators in the assessment of the CT of learners. One of the most relevant tools is Hairball (Boe, 2013), a static code analyzer for Scratch projects that detects programming errors in the scripts of the projects.

Dr. Scratch (Moreno, 2014) is a free/open source web tool, powered by Hairball, that analyzes Scratch projects to automatically assign a CT score in terms of abstraction and problem decomposition, parallelism, logical thinking, synchronization, flow control, user interactivity and data representation. Section 3 presents the algorithm used to assess the CT from Scratch code, which has been developed by remixing different proposals of educators and researchers who are using Scratch to teach Computer Science in primary and secondary schools. Section 4 shows the results of analyzing 100 projects we randomly downloaded from the Scratch web repository. Finally, in the conclusions of the paper we discuss the limitations of our approach, as some pillars

of CT, such as debugging or remixing skills, cannot be evaluated with this solution.

Background

The assessment of the development of CT is one of the most discussed topics by educators and researchers in educational conferences, seminars and workshops in the last years. Thus, if we focus in the Scratch programming language, it is possible to find several papers presenting different strategies to measure the development of CT of learners by studying their Scratch projects.

In the paper “New frameworks for studying and assessing the development of computational thinking” (Brennan, 2012), an approach in three phases is presented: project portfolio analysis using Scrape (Wolz, 2011), a tool which allows to visualize the blocks utilized by a particular user in the projects uploaded to the Scratch website, artifact-based interviews and design scenarios.

A different strategy is suggested by Wilson, Hainey and Connolly (Wilson, 2012), who presented a scheme to assess the level of programming competence demonstrated in a Scratch project. In order to do so, they analyze the use of some programming concepts (such as threads or conditional statements), study the organisation of the code (like variable and sprite names) and evaluate design and usability aspects (such as functionality or originality).

Aiming to study the variations in the development of CT among primary students of different ages, Seiter and Foreman developed the Progression of Early Computational Thinking Model, a model that “synthesizes measurable evidence from student work with broader, more abstract coding design patterns, which are then mapped onto computational thinking concepts” (Seiter, 2013).

Finally, Boe et al. developed Hairball (Boe, 2013), a tool that can be used by evaluators to assess the correctness of Scratch projects. Hairball, which is inspired by Lint (Johnson, 1977), performs static analysis of the programs of a project to detect different kinds of issues, such as dead code, wrong attribute initialization or incorrect message synchronization. In order to test the robustness of the tool, Hairball was used to assess Computer Science learning in a Scratch-based summer camp (Franklin, 2013).

Methodology

Inspired by the work reviewed in Section 2, we developed a plug-in for the Hairball environment, *Mastery*¹, that analyzes a Scratch project to assign a CT score depending on the competence demonstrated by the developer on the following seven concepts: abstraction and problem decomposition, parallelism, logical thinking, synchronization, algorithmic notions of flow control, user interactivity and data representation. In order to evaluate the level of development on each of these concepts, the *Mastery* plug-in implements an algorithm based on the rules in Table 1.

Figure 1 can be used to illustrate the operation of the plug-in. Thus, following the rules in Table 1, the first script of the picture would be catalogued as *basic* in terms of logical thinking, as only *if* statements are used. The second script, however, would be considered to demonstrate a *developing* level, because an *if_else* block is utilized. Finally, the third script would prove a *proficient* level on this concept, as a logical operation, *or*, is being used.

¹ <https://github.com/jemole/hairball/blob/master/hairball/plugins/mastery.py>

CT Concept	Basic	Developing	Proficiency
Abstraction and problem decomposition	More than one script and more than one sprite	Definition of blocks	Use of clones
Parallelism	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, create clone, two scripts when %s is > %s, two scripts on when backdrop change to
Logical thinking	If	If else	Logic operations
Synchronization	Wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
Flow control	Sequence of blocks	Repeat, forever	Repeat until
User Interactivity	Green flag	Key pressed, sprite clicked, ask and wait, mouse blocks	When %s is >%s, video, audio
Data representation	Modifiers of sprites properties	Operations on variables	Operations on lists

Table 1. Level of development for each CT concept

The overall CT score is calculated by adding up the partial scores of each CT concept. Thus, projects with up to 7 points are considered to prove a *Basic* CT, while projects between 8 and 14 points are evaluated as *Developing*, and projects with more than 15 points are marked as *Proficient*.

With the aim of making it easier for users to analyze their projects, we have developed a web tool called Dr. Scratch² that allows users to analyze Scratch projects by either uploading a .sb2 file or just introducing the URL of the project. Figure 2 shows the output of the Dr. Scratch tool after performing the analysis on a Scratch project called Just Jump³. An overall CT score of 16 points is computed, informing the user about the partial count of the different CT concepts in a table below.

² <http://drscratch.org>

³ <http://scratch.mit.edu/projects/52452686/>

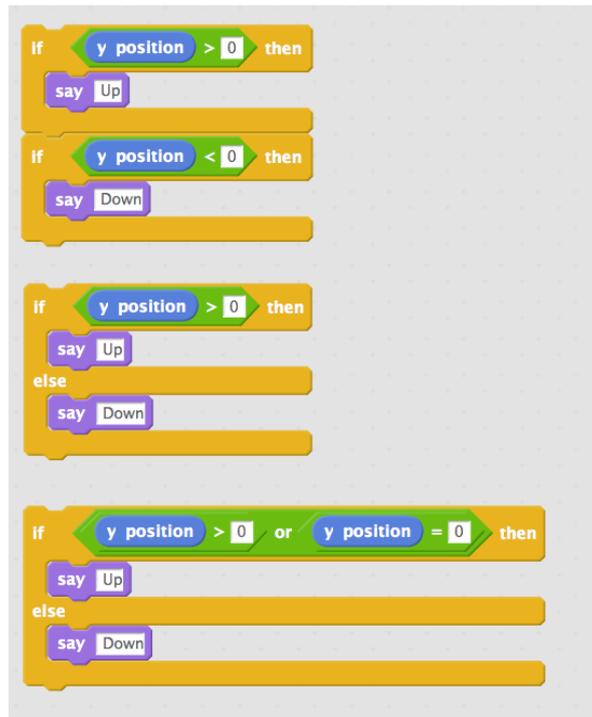


Figure 1. Different levels of development of logical thinking: basic (top), developing (center) and proficient (bottom).

Future versions of Dr. Scratch are planned to provide more information on how to improve each of the aspects where there is room for improvement by the learner. In fact, brave readers can try the new features the development team is working on at the preproduction version of Dr. Scratch⁴, where the feedback report is completed with further information that learners can use to improve their coding skills. Nevertheless, at the moment of writing this paper, this version of the tool is unstable and therefore unreliable, as developers are constantly including and testing new enhancements.

Dashboard Overview

Dashboard

Welcome to Doctor Scratch. Project results

 **CT Score: 16/21**

Your level: **Master**

[Tweet](#) 0

CT Score in detail:

Concept	Points
Abstraction	2/3
Parallelization	1/3
Logic	3/3
Synchronization	3/3
FlowControl	3/3
UserInteractivity	2/3
DataRepresentation	2/3

Figure 2. Dr. Scratch shows the CT Score after analysing a Scratch project.

⁴ <http://drscratchpre.programamos.es>

Preliminary findings

In order to test the operation of the *Mastery* plug-in, we randomly downloaded and analysed 100 projects from the Scratch repository. The average CT score was 14.4 points, while the median was 16 and the mode 18. As can be seen in Figure 3, which presents the mean score for each of the CT components, the concepts in which higher results were obtained are flow control, abstraction, parallelism and synchronization, while user interactivity and data representation got slightly lower values.

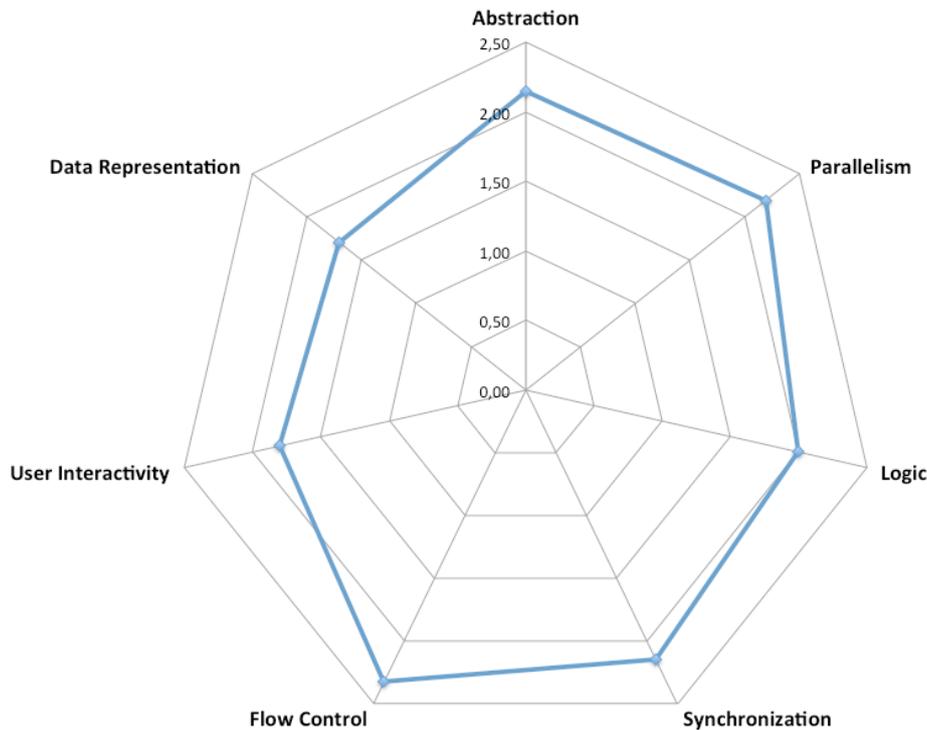


Figure 3. CT score average of 100 randomly downloaded project from the Scratch repository.

Conclusions and future work

In this paper we present the procedure used by the Dr. Scratch tool in order to assess the CT skills demonstrated by a Scratch project. The tool assigns a CT score, which is calculated by adding up the partial counts of each CT concept: abstraction, logical thinking, synchronization, parallelism, flow control, user interactivity and data representation.

This approach has several limitations. On the one hand, the examination of a single project might not be as accurate or complete as the analysis of the collection of projects of the user; in this regard, the new feature of Dr. Scratch that will allow scratchers to create an account to store the record of multiple analyses may alleviate this issue. Furthermore, the use of a particular block or groups of blocks is not enough to confirm fluency on a certain CT concept; other plug-ins like *Dead code*, *Attribute initialization*, *Sprite naming* or *Repeated code* have been incorporated to the Dr. Scratch tool aiming to detect if the blocks are being used correctly (Moreno, 2014), although the inner working of these plug-ins and other new features is out of the scope of this paper. Nevertheless, the biggest limitation of this approach is the fact that some key CT competences

cannot be measured by analysing the code of a project, such as the debugging or remixing skills.

Therefore, this solution must be used by students as a tool to receive feedback that might help them to discover areas in which to focus to keep on improving their coding skills, or by teachers as a tool that might assist them in the assessing of Scratch projects, but not as a replacement of the evaluators work. A simple project with the appropriate blocks could get a high CT score although its functionality might be useless. In addition, important aspects on educational environments, such as originality or creativity are not evaluated, so teachers should not rely exclusively on the score assigned by Dr. Scratch.

In the near future we plan to carry out new research to test the effectiveness of the procedure presented in this paper as a means to assess the CT by comparing the results obtained with other tools and solutions that have already been tested and with a panel of experienced teachers and evaluators.

Acknowledgements

The work of Jesús Moreno-Leon and Gregorio Robles has been funded in part by the Region of Madrid under project “eMadrid - Investigacion y Desarrollo de tecnologías para el e-learning en la Comunidad de Madrid” (S2013/ICE-2715). The authors are very thankful to Eva Hu Garres and Mari Luz Aguado for their technical support with Dr. Scratch.

References

- Boe, B. a. (2013). *Hairball: Lint-inspired static analysis of scratch projects*. (ACM) Proceeding of the 44th ACM technical symposium on Computer science education.
- Brennan, K. a. (2012). *New frameworks for studying and assessing the development of computational thinking*. Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada.
- European Schoolnet. (2014). *Computing our future. Computer programming and coding – priorities, school curricula and initiatives across Europe*. European Schoolnet.
- Franklin, D. a.-T. (2013). *Assessment of computer science learning in a scratch-based outreach program*. (ACM) Proceeding of the 44th ACM technical symposium on Computer science education.
- Johnson, S. C. (1977). *Lint, a C program checker*. Citeseer.
- Lye, S. Y. (2014). *Review on teaching and learning of computational thinking through programming: What is next for K-12?* (Vol. 41). (Elsevier) Computers in Human Behavior.
- Moreno, J. a. (2014). *Automatic Detection of Bad Programming Habits in Scratch: A Preliminary Study*. (IEEE) Frontiers in Education Conference, 2014 IEEE.
- Seiter, L. a. (2013). *Modeling the learning progressions of computational thinking of primary grade students*. (ACM) Proceedings of the ninth annual international ACM conference on International computing education research.
- Wilson, A. a. (2012). *Evaluation of computer games developed by primary school children to gauge understanding of programming concepts*. 6th European Conference on Games-based Learning (ECGBL).
- Wing, J. M. (2006). *Computational thinking* (Vol. 49). Communications of the ACM.
- Wolz, U. a. (2011). *Scrape: A tool for visualizing the code of Scratch programs*. Poster presented at the 42nd ACM Technical Symposium on Computer Science Education, Dallas, TX.

Copyright

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International license (CC BY-NC-ND 4.0). To view a copy of this licence, visit <https://creativecommons.org/licenses/by-nc-nd/4.0/>