ToolboX.Academy: Coding & Artificial Intelligence made easy for kids, Big Data for educators

F Vico[1,2], J Masa[2], R Garcia[2]

[1] *D. Lenguajes y Ciencias de la Computación University of Malaga (SPAIN)*

[2] *ToolboX.Academy (Madrid, SPAIN)*

Abstract

Society has reached the point where there is considerable awareness about the need of teaching coding skills at the schools. Both, political leaders and educational systems start taking positions in this strategic new arena, but they lack from much of the expertise and necessary tools about how to teach this complex (but also intuitive to children and youngsters) field of knowledge. In this context, ToolboX Academy is a programming platform which subscribes a slightly new approach to attack the problem, in four differential aspects: 1) While most attention focuses on block-based approaches (Scratch and Code.org being the most popular initiatives), ToolboX embraces coding as it works for developers: text-based programming; 2) ToolboX embeds mainstream programming languages, which are widely used, are supported by big communities, are weakly typed, and closer to human natural language, like it is Javascript, for web applications, and GNU Octave, in engineering and scientific research; 3) it provides user-oriented problem-based learning, designed upon educational research results; and 4) it integrates curricular subjects in learning to code. In this paper, we review the design constraints taken into account to develop ToolboX Academy, and how it has been tested in the classroom. Some features are particularly stressed, like the use of open formats for content representation (definition of tasks or problems); the design of a simplified environment to represent most CS concepts, from basic loops to complex graph traversal strategies; and the use of learning analytics based on big data algorithms that can help educators and managers to learn from the interaction of their students with this environment.

Keywords: text-based programming, mainstream programming languages, problem-solving, curricular subjects, open formats, simplified coding environments, learning analytics, big data.

Preview of the version published in Proceedings of the EDULEARN19, 5173-5178, 2019.
ISBN: 978-84-09-12031-4. ISSN: 2340-1117

1 INTRODUCTION

Human civilizations have always progressed hand in hand with technology, and the current paradigm for social organization we are converging to is, by no means, different, since it assigns a central role to computers, communication systems and robotics. Be it the productive sectors, the institutions, or the information and communication needs among people, the networks of computers conform the substrate for making it happen. From automated manufacturing plants and warehouses, towards the ubiquitous mobile devices, going through self-driving cars and the large computer clusters that provide the Internet, to understand the architecture and functioning of these systems will be one of the most relevant aspects for the education of the human being in the digital era.

And something is moving in this direction. Mainly motivated by the demand of developers in tech companies, several initiatives (particularly the Hour of code[1]) promote teaching computer programming at the schools[2], and even the K-12 stars to integrate matters like digital literacy, computational thinking, and coding in their curricula. But changing the educational system is a very slow process, which demands political sensitivity and the involvement of many actors. On the other hand, even if handling electronic devices and programming computers seems to be intuitive for children, these computing stuff is promoted (and used) more as gaming platforms and information systems to access data, services and for social interaction. Indeed, despite the popularity of computing coding in the eighties, acquiring programming skills is considered as a tough endeavour. All together, institutional and market obstacles make it difficult for public education to adapt the curriculum and democratize computer coding.

There are also structural reasons. While Computer Science studies are well established at the university level, there is a fundamental lack of progress both in designing curricula for the K-12 grades, and also methodological, as it is unclear how to implement the initiatives (robotics vs. coding, certified educators vs. autonomous learning) and at what cost. Then, we arrive to this dilemma: we know how to teach children to read and write (even at premature ages), but how to teach them coding? There is little work done in this sense [3, 4], and, in general, experimentation is based on very limited samples and methodologies. In such a way, the efficacy of Scratch to teach coding is highly questionable in children from 8 to 16 years old[3] (acquisition of bad-habits have been reported[5] and it does not increase their ability to solve problems [6], while there are also some positive results, but on non-representative samples [5, 12]). Aligned with these problems, there are no standards for educators to assess coding skills. While they abound at the professional level, given its importance in recruiting programmers for tech companies[9, 10], for the pre-college the first proposals have not been validated and sometimes are associated to commercial products[11]. Finally (and this makes a difference with other subjects) teaching coding demands dedicated computers, and technical personnel for installation and maintenance, making it dependent on additional budgets.

ToolboX.Academy is a different approach to learning coding, in the sense that it adapts the technology to children (not the other way around). Making it simple, text-based (instead of blocky) and fun for students, with a minimal set of elements in the interface, and abundant content which is adapted to the capacity of each student to introduce all basic computational concepts, is a key for success in this matter. At the same time, collecting usage data and analyzing it with big data techniques for a better understanding of students' capacities and limitations, and how different groups develop with respect to mean curves of the same age, center and region, completes the set of tools that are proposed to implement learning to code.

2 DESCRIPTION OF THE ENVIRONMENT

ToolboX.Academy is a platform that integrates an intuitive programming environment for children, and content, in the form of sets of tasks to be solved by writing scripts. As a difference with other approaches to teach coding to children, here the tasks database is big enough to cover a full course of computer programming, for all grades of primary and secondary education, focusing on conventional coding skills in primary education, and advanced programming and Artificial Intelligence in Secondary education. This follows an innovative coding curriculum designed based on the results of an experiment over a representative sample of more than a thousand children of all grades[12]. Another important improvement is in the tracking of students' progress, since the environment stores events as the interaction with the interface takes place, generating a huge amount of metadata, that can be interpreted with Big Data methods, to better understand the learning process, and to detect disorders and special capacities in the students[13].

2.1 Front-end: the student's interface

As shown in Fig. 1, the environment is divided into three main areas: statement window, script editor and graphic interface. The statement defines the task, is what the script has to do, and it is shown in the top-left window. Typically, it is related to a concrete mission that a robot avatar is intended to perform, like picking up objects, finding something or going somewhere. On the right window the student can write the script in Javascript to solve the task. This script editor is based on plain text and, like conventional programming environments, it provides tools that help in development, like line numbering, and pointing out warnings and errors in the code. Syntax highlighting is a very convenient feature, since it helps in identifying (and learn) the nature of the different elements in the code, like reserved words, commands, numbers or operators. Finally, the graphic window (bottom-left) allows the user to interpret the elements of the task, their interrelationships, and whether the script does solve the problem. In native programming tasks, a simulated world is shown, where robots and other elements show around. In non-native coding, this window shows graphics, simulation of physical or chemical processes, or even objects to be counted for kids.

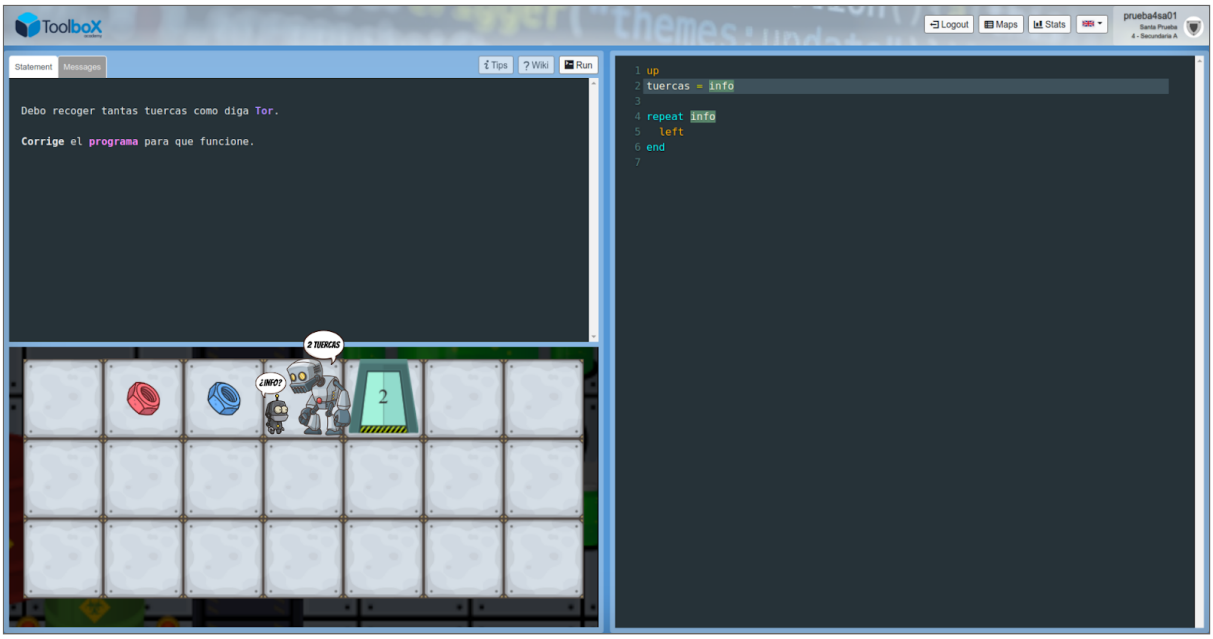

Figure 1. Distribution of windows and elements in the interface of ToolboX.Academy: Statement window (top-left), script editor (right) and graphic window (bottom-left).

In a regular session, and depending on age and progression curve, the user is presented with a number of tasks, one after another, and it is intended that she solves them interactively. By obtaining information from the *Tips* and *Wiki* buttons, the user writes a script that solves the task posted in the statement. While the *Tips* button informs about how the solution is to be planned (e.g. giving hints on how to start, or how to approach a tricky part of it), the Wiki button provides information on computational structures, like how do they operate or their syntax.

Once the script has been written, it is executed by the student and it generates an outcome, either as an alphanumeric computation, or by moving characters in the simulated world. In this way, the system informs about the result, be it a compilation error generated by the script, a normal execution with a wrong result, or a successful execution. In the first two cases, the user is informed of what went wrong for code debugging or rewriting, and in the latter case, if the task has been solved, a take-home-message is shown, comprising what has been learned in solving the task.

2.2 Back-end: the center's interface

As it has been stated above, one of the main features of ToolboX.Academy is the fact that it stores all the information generating from the student-interface interaction, allowing to further explore student and groups progress (Fig. 2), as well as detecting abnormal capacities or disorders (curves labelled with blue and red points in Fig. 2).

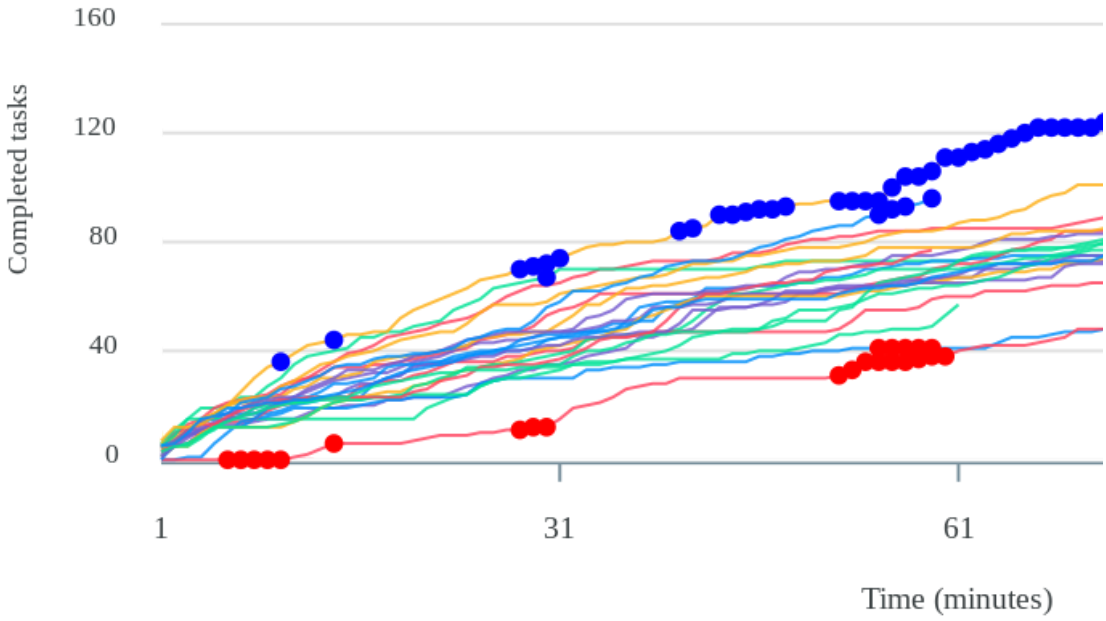

Figure 2. Backend: learning curves of the students in a classroom during a coding session. Variable slopes in the mean curve correspond to different degrees of difficulty in the tasks. Curves highlighted with blue points identify atypically fast progress with respect to the rest of the group (screening of potential gifted students), while the red points identify the periods when the inferior curve was progressing atypically slow (potential ADHD cases).

As it has been shown in a controlled experiment where most metadata were recorded[13], the administration of the educational centers, educators and parents, can benefit from the outcome of big data analysis.

3 CONCLUSIONS

In the last decades, personal computers have evolved. Much in the way that microprocessors have attached to Moore's law (doubling the number of transistors every two years), storage, integration, input/output interfaces, monitors and other devices have little to do with the first computers designed for domestic use in the eighties. And so it happened to software and programming languages, which now are ready for building the complex applications that institutions, companies and Internet users demand. But during all this time, children did not gain access to this more and more sophisticated technology, and Computer Science studies developed and was taught exclusively at universities and research centers, not schools. And while the Internet is helping in providing new environments to fill this gap, most efforts end up with no more than introductions to the matter, eluding the responsibility of bringing children a reliable tool that provides the main computational concepts, and, at the same time, reveals data about how this learning process takes place. For example, one outstanding conclusion that directly derived from the use of ToolboX.Academy is that the gender gap in the STEM area is not provoked by an innate predisposition of male students to acquire coding skills, since a rigorous statistical study showed[14] that both genres perform similarly when they learn with this tool.

And there are also risks. The current scenario has provoked that learning computer programming is considered more like an extracurricular activity, than a fundamental competence in primary and secondary education. In a similar way that learning music practice is recommended for its benefits on brain development and academic performance[15, 16], computational thinking is on the verge of being considered complementary to the standard curriculum. That would be a wrong way in solving the problem of digital illiteracy, not only for being an elitist solution, but also for the infravaluated role that coding would assume in a society which is more and more dependent on software, in all aspects of life.

ACKNOWLEDGEMENTS

The authors are indebted to the educational centers (both in the public system: Intelhorce and La Biznaga schools, and Emilio Prados high school, and private system: El Pinar and San Patricio) for helping in the initial steps of the design and further beta testing.

REFERENCES

[1] Code.org (2015). Hour of code. Retrieved from https://hourofcode.com/ on May 28, 2019.

[2] F.J. García-Peñalvo, D. Reimann, M. Tuul, A. Rees & I. Jormanainen. An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers. 2016. TACCLE3 Consortium.

[3] C.T. Hsin, M.C. Li & C.C. Tsai. The influence of young children's use of technology on their learning: A review. *Educational Technology & Society*, **17** (4), 85–99. 2014.

[4] G. Fessakis, E. Gouli & E. Mavroudi. Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, **63**, 87-97. 2013.

[5] J. Moreno, & G. Robles. Automatic detection of bad programming habits in Scratch: A preliminary study. *Frontiers in Education Conference*, IEEE, 1-4. 2014.

[6] F. Kalelioglu & Y. Gülbahar. The Effects of teaching programming via Scratch on problem solving skills: A Discussion from learners' perspective. *Informatics in Education*, **13**(1), 33-50. 2014.

[7] D. Franklin, P. Conrad, B. Boe, K. Nilsen et al. Assessment of computer science learning in a Scratch-based outreach program. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 371–376. 2013.

[8] J. Maloney, K. Peppler, Y. Kafai, M. Resnick & N. Rusk. Programming by choice: urban youth learning programming with Scratch. *ACM SIGCSE Bulletin*, **40**(1):367–371. 2008.

[9] S. Joseph. Programmer competency matrix. .Net blog of Sijin Joseph. Retrieved from http://www.starling-software.com/employment/programmer-competency-matrix.html on May 28, 2019. 2008.

[10] R. Poss. How good are you at programming? A CEFR-like approach to measure programming proficiency. Retrieved from http://science.raphael.poss.name/programming-levels.html on May 28, 2019. 2014.

[11] Kodable. Computer science standards. SurfScore, Inc. [Checked March 15, 2017]. Retrieved from http://resources.kodable.com/public/cs_standards_2017.pdf. 2017.

[12] F. Vico, M Molina, D. Orden, J. Ortiz, R. Garcia & J. Masa. A coding curriculum for K-12 education: the evidence-based approach, in *Proceedings of the 11th annual International Conference on Education and New Learning Technologies* (EDULEARN19), 7102-7106. 2019.

[13] F. Vico, M. Molina, D. Orden, F. Rivas-Ruiz, R. Garcia & J. Masa. Managing e-learning metadata within ToolboX.Academy: knowledge acquisition with Big Data algorithms at the student and group levels, in *Proceedings of the 11th annual International Conference on Education and New Learning Technologies* (EDULEARN19), 7030-7035. 2019.

[14] F. Vico, M. Molina, D. Orden, F. Rivas-Ruiz, R. Garcia & J. Masa. Coding skills are acquired gender-independently in the K-12 system: the ToolboX.Academy experience, in *Proceedings of the 11th annual International Conference on Education and New Learning Technologies* (EDULEARN19), 7076-7079. 2019.

[15] S.M. Carpentier, S. Moreno & A.R. McIntosh. Short-term music training enhances complex, distributed neural communication during music and linguistic tasks. *Journal of Cognitive Neuroscience*, **28**(10):1603-12. 2016.

[16] K.L. Hyde, J: Lerch, A. Norton, M. Forgeard, E. Winner, A.C: Evans & G. Schlaug. Musical training shapes structural brain development. *Journal of Neuroscience*, **29**(10):3019-25. 2009.