Proceedings of the 5th World Conference on Media and Mass Communication, Vol. 5, Issue 1 2019, pp. 12-24

ISSN 2424-6778 online
DOI: https://doi.org/10.17501/24246778.2019.5102

RAPID AND EASY PROTOTYPING OF MULTIMEDIA TOOLS FOR EDUCATION

Stefano Federici*, Jacopo Molinas, Elisabetta Sergi, Riccardo Lussu and Elisabetta Gola

University of Cagliari, Italy

Abstract: Today the gap between students and teachers is growing. On the one side, students want to be engaged with methodologies that they recognize as their own, that is multimedia. On the other side, teachers very often have no deep understanding of new technologies. Some teachers can build creative presentations, but they very rarely can build engaging multimedia educational tools to teach and test their students' knowledge. To overcome the difficulties of the teachers in creating their own multimedia interactive educational tools, a simple programming environment, BloP, has been built. BloP has been tested in several educational tasks, such as teaching foreign languages or telling stories. New multimedia tools, that have been designed by using BloP, have been built in just a few weeks by students of a non-scientific degree. The tools have been very well accepted by users and have shown effective in improving their performances in the intended tasks. This experiment shows that even non-technical teachers could quickly and easily build their own multimedia educational tools by using specifically tailored environments and that those tools can be effective in improving the student's engagement and their performances.

Keywords: multimedia tools; block programming; rapid prototyping; educational tools

Introduction

Teaching in an effective way is getting today more and more difficult, as the gap between students and teachers enlarges. Whereas multimedia are the media of election for students that, in order to succeed, must be engaged with methodologies that they recognize as their own, teachers very rarely have a deep understanding of new technologies.

Teachers are often able to build creative presentations, but they can very rarely build engaging multimedia educational tools to teach and test their students' knowledge (TES and Nesta, 2014).

In order to overcome teachers' difficulties in creating multimedia educational interactive tools, a simple programming environment has been built. The new environment, based on the metaphor of building bricks, that is programming by assembling colored building blocks (Maloney *et al*, 2009), has been tested in several educational tasks, e.g. teaching foreign languages or telling stories, by adding new programming blocks to a block programming environment and by reshaping the environment so to make it closer to what the students would expect when they are learning a specific topic.

The media generation and the no media teacher

We clearly see that digital devices are a bigger and bigger part of the lives of our children. But at school, that is the place where they live a big part of their lives, the digital world has not the prominent role that it should have.

What is the reason? It is maybe that teachers are strongly against the usage of media and technology in education? Not at all. Recent surveys (British Telecommunications and IPSOS Mori, 2016, Schaffhauser and Nage, 2016) reveal that most of teachers think that technology could have a great and positive impact on education and that the usage of technology at school can have positive effects not only on tasks as problem

Corresponding Author E-mail: *sfederici@unica.it

solving and numeracy, something that we could maybe expect, but also on literacy, something that maybe we don't expect (Table 1).

Table 1:Recent surveys (Schaffhauser and Nage, 2016, British Telecommunications and IPSOS Mori, 2016)

Country	Year	Teachers think:	Percentage
UK	2014	not able to teach coding	50%
UK	2016	not prepared to equip kids for a digital world	75%
US	2017	no enough training to teach with technology	78%

The teacher student gap

Despite the clearly very positive attitude of teachers towards the usage of technology in education, a large part of teachers does not think that they are ready to use technology and coding in their classrooms. In a recent survey by TES and Nesta (2014), about 50% of the UK teachers interviewed think that they are not ready to teach their students how to code (Table 2, row 1). In another survey by the UK telecommunications company BT and by the UK market research company IPSOS Mori (British Telecommunications and IPSOS Mori, 2016), a large part of the teachers interviewed said that they see as part of their job equipping kids for a digital world. But only 25% of teachers agree that they feel prepared to it (Table 2, row 3). US teachers share the same opinion (Bolkan, 2017): 78% of them think that they haven't received enough training to teach in the classroom by using technology (Table 2, row 3).

Table 2: Recent surveys (TES and Nesta, 2014, British Telecommunications and IPSOS Mori, 2016, Bolkan, 2017).

Country	**Year**	**Teachers think that the use of technology has positive impact on:**	**Percentage**
US	2016	education	90%
UK	2016	problem solving	99%
UK	2016	numeracy	96%
UK	2016	literacy	69%

Block Programming for All

The fear of coding and technology witnessed by so many teachers is an ill-posed problem. Today every teacher can start creating their own multimedia supports by using very simple programming languages made of colored blocks that can be easily assembled in meaningful code (Figure 1). These languages have been created for kids and they are very easy to use. But being created for kids does not make them something that can be used just to play: they are also incredibly powerful in order to meet a lot of different expectations (Resnick and Silverman, 2005).

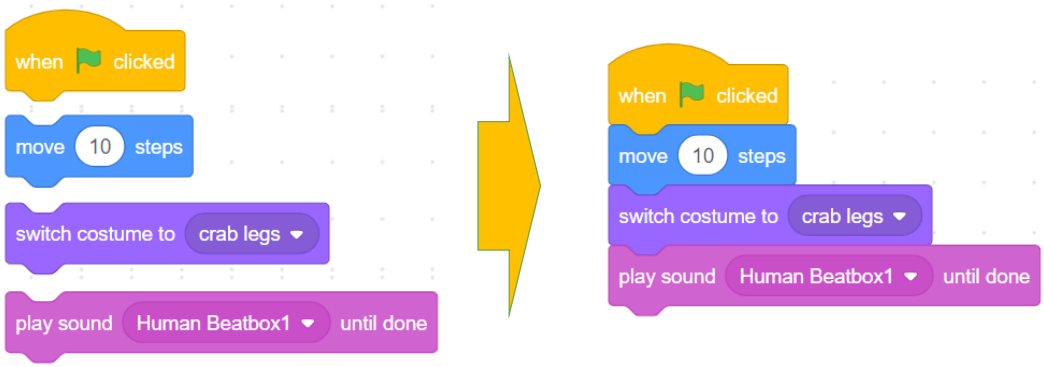


Figure 1 Colored blocks are assembled in meaningful code

One of these languages, the most well-known, is Scratch, created by the Media Lab of MIT (Massachusetts Institute of Technology) in Boston (Maloney *et al*, 2009). Everything in this tool is under the eyes of the newbie coder (Figure 2).

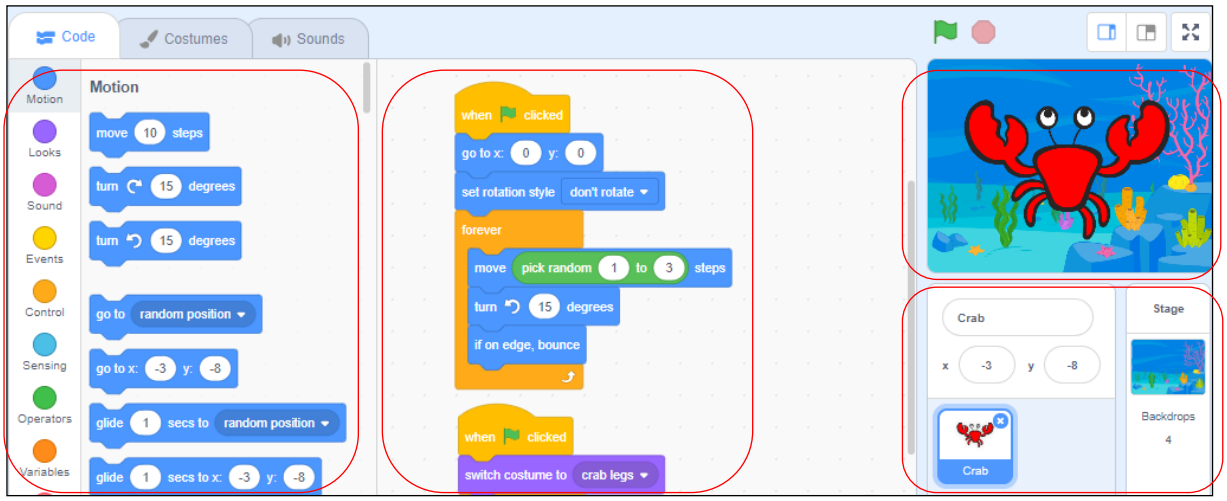


Figure 2 Elements of the MIT Scratch's interface and programming language: the blocks in the block area (left hand side), the scripts built by using blocks dragged from the block area (central area of the tool), the Stage where interactive objects act (top right area), the list of interactive objects (bottom right area)

In order to validate the hypothesis that these tools are intuitive and easy to use, we have run several experiments (Federici *et al.*, 2015) on groups of 10 to 20 teachers of both scientific and non-scientific subjects. All the teachers, with no exception, were able to discover the elements of this multimedia programming language by themselves. This discovery process, that made them able to easily create simple educational apps, took less than 2 hours.

Educational apps built by teachers improve the student performances (Chang *et al*, 2010, Beatty, 2013, Warschauer, 2013, Federici *et al.*, 2018), but it is when the students build the apps by themselves that they get a higher retention for the knowledge they acquired. Indeed, by building the app, they can manipulate the exact elements that are at the base of the topic they are learning, thus understanding the real structure of the topic (Udomon, 2013, Federici *et al.*, 2018).

Using Scratch to build an educational tool for each different topic requires a somewhat long time. So, we thought that a possibile solution to this problem could be making available to the students a simplified topic-specific block programming tool, with only a few programming blocks, where each command -represented by a programming block- corresponds to one of the underlying concepts of the topic. So, to give a few example, the new language would model operations and numbers to study difficult mathematical concepts; nouns, adjectives and verbs to study a foreign language; the concept of sequence of actions/events to study history, etc.

Learning professional computer programming by block programming

A good support tool for this new way of teaching and learning -that is creating simplified, topic-specific programming languages- is a tool that allows even teachers of non-scientific subjects to easily build the necessary programming blocks of each new topic-specific programming language. Furthermore, the new language needs to be embedded in a safe programming environment that the students, when using the environment, cannot impair.

Such a tool, called BloP, had been already developed in order to overcome the difficulties of students in learning professional programming languages (Federici, 2011, Federici and Gola, 2014). The BloP tool (that stands for Block Programming tool) had been developed by adding advanced features to a Scratch clone called Snap (Harvey and Monig, 2010). Snap, being written in Javascript, could easily run on every modern PC without need for powerful hardware or installation. The first programming language developped with BloP was a block programming version of what was felt by engineering students as a very difficult programming language, the C/C++ programming language (Federici, 2011). Whereas showing a message on the screen is a trivial operation in Scratch or in Snap (Figure 3a), getting the same result in C requires writing a non trivial sequence of commands (Figure 3b).


```
#include <stdio.h>
int main(int argc, char** argv) {
      printf("Hello, World!\n");

}
```

(a) Snap (b) C/C++

Figure 3:Making the "Hello, World!" message show up on the screen in Snap and in C/C++

The new block version of C/C++, called blockC++, showed to be very successful (Federici and Stern, 2011) because all the student had to do was dragging around colored blocks (Figure 4a) instead of remembering cryptic words and symbols and typing letters (Figure 4b).


```
#include <stdio.h>
int main(int argc, char** argv) {
    printf("Hello, World!\n");
}
```

(a) blockC++ (b) C++

Figure 4:The same program written by dragging colored blocks and by typing letters on a keyboard

Even recent studies showed that non-experienced users prefer block versions to text versions of the same programming language (Homer and Noble, 2017) and that they get better results when they learn the basics of computer programming by using a block programming language than a textual language (Weintrop and Wilensky, 2017).

A meta block programming environment

The BloP tool is a metatool that allows teachers to easily and quickly develop simple-to-use versions of simple or even complex programming languages based on block programming, such as Logo (Figure 5), a simple educational language for kids, C++ (Figure 6), a complex language for expert programmers (Federici, 2011),

languages for the web such as PHP and MySQL (Figure 7), etc. The blockPHP and blockMySQL web programming languages (Figure 6) have been developed for example for students of a Communication Studies degree that had to learn how to build a simple dynamic web site by using a reduced set of HTML, PHP and MySQL.

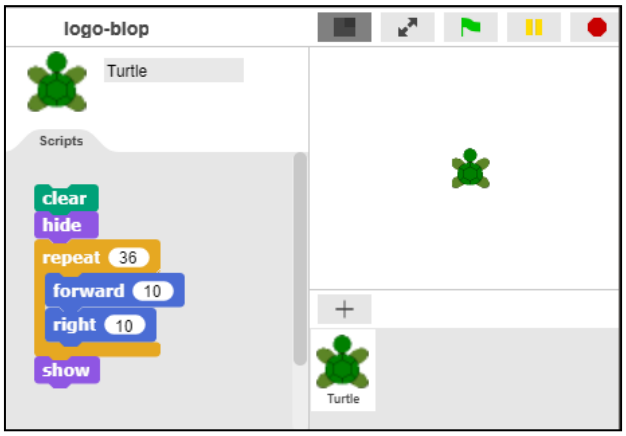


Figure 5 : blockLogo, block version of the Logo educational programming language

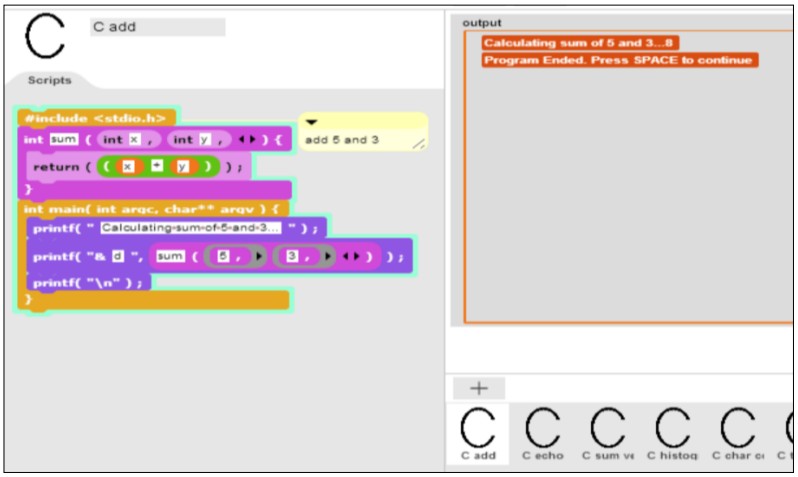


Figure 6 : blockC++, block version of the C++ professional programming language

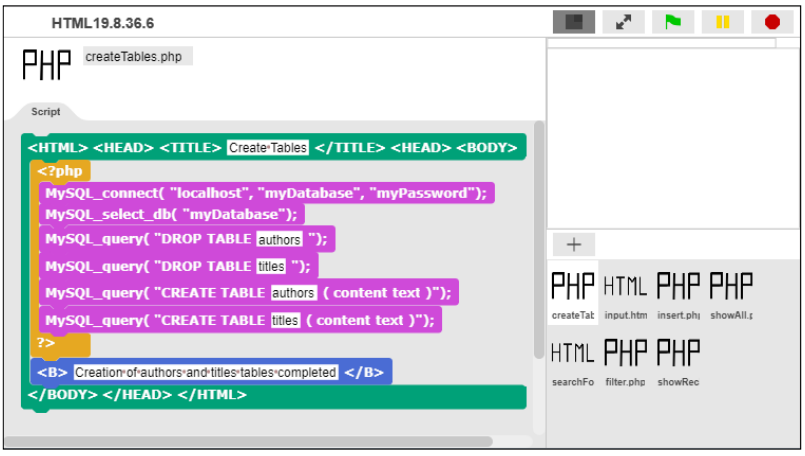


Figure 7 : blockPHP and blockMySQL web programming languages

Building a new language is quite simple, as the main building blocks of almost all computer languages are only apparently different and, more importantly, they are already there, ready to be adapted. See, for example, the

different syntax of the C++ and MySQL loops shown in Table 3. They are only superficially different. Internally, they are very similar. So, building a new loop by borrowing the mechanism of Snap or of a previously developed language is really straightforward. Even developping a language such as MySQL, a web language that handle database operations, starting from an apparently very different language such C++, is just a matter of a few hours of work.

Table 3 :C++ and MySQL loops.

C++ loop	MySQL loop
for (i=0; i < length; i++) { }	UPDATE *table_name* SET *column* = *value1*

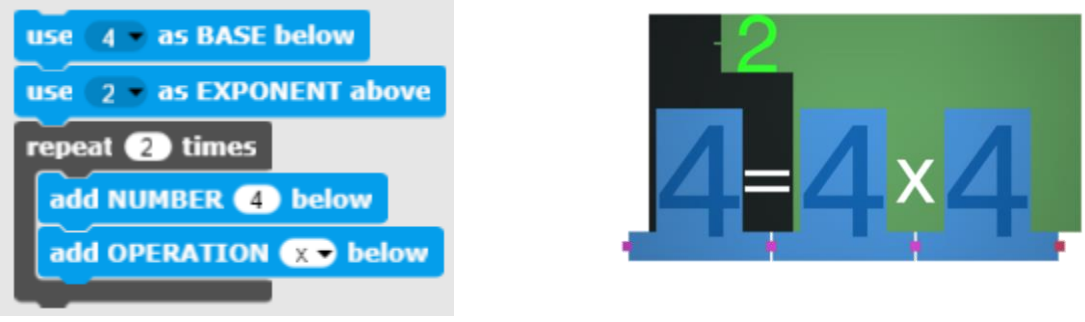
Studying school subjects in a block programming environment

In order to fruitfully use computer programming, students must understand very well how each single instruction of the programming language works and how they interact in order to get the desired result. The same happens when students learn every other school topic: they must learn which are the basic elements of the topic and understand which are their relevant interactions (Boulton, 2013). So, computer programming and general learning work in the same way. For example, in order to correctly use exponentiation, students must understand that exponentiation -like every other mathematical tool- is just the abbreviation of simpler operations, in this case the abbreviation of a sequence of multiplications where we *multiply a base number several times* (Table 4).

Table 4 :Exponentiation operation.

Exponentiation	Corresponding multiplication	Meaning
4^2	4 x 4	Multiply *base number* 4 *2 times*
4^3	4 x 4 x 4	Multiply *base number* 4 *3 times*
4^4	4 x 4 x 4 x 4	Multiply *base number* 4 4 *times*

This approach has shown effective when teaching to the students how to "program" the behaviour of these elements in Scratch (Federici *et al.*, 2018). The alternative approach proposed in this work is instead the building of a new simplified block programming language that will model the behaviour of the desired school subject, in this case the exponentiation operation.



(a) 4^2 script (b) 4^2 output

Figure 8 :Building exponentiation operations by a topic-specific block programming language

As shown in Figure 8a, by "programming" the creation of, for example, 4^2, students have to define the value of the base ("use 4 as BASE below" block), define the value of the exponent ("use 2 as EXPONENT above"

block) define how many times ("repeat 2 times" block) a given value ("add NUMBER 4 below" block) must be used in a sequence of operations ("add OPERATION x below" block). By running the script, the final output is what is shown in Figure 8b, that is that 4^2 is the same as 4 x 4. If the student don't use the correct value for BASE, EXPONENT, TIMES, NUMBER or OPERATION, the output will be an error message explaining the kind of mismatch that has been detected. So, by programming the exponentiation operation with these blocks, the student learns that the value of "repeat 2 times" must match the value of the exponent, that the value of "add NUMBER 4 below" must match the value of the base, that the mathematical operation in "add OPERATION x below" must be the multiplication operation.

The students must then learn that the basic elements of the exponentiation operation are the *base* and the *exponent* and that the relationship $base^{exponent}$ means multiplying the base number a number of times equal to the exponent. Whereas using such relation several times when solving mathematical expressions can be done without paying too much attention to the exact elements of the relation, as they need just to replicate what has been done to solve the first expression, in this approach the student must select the correct block so that the correct element is "below" (base) or "above" (exponent), then must use the correct repetition and must select in the menu the correct operation, in this case "x" for multiplication. In this way they pay attention to what they are doing and understand what is the real meaning of the exponentiation operation. It has been shown (Federici *et al.*, 2018) that when students must build the operation programmatically, the retention, that is the performance when replicating the task six months after they learned it, is higher than when just using a multimedia tool to learn exponentiation and even higher when learning mathematical operations in the classic "blackboard and chulk" approach (Table 4, row 2). The correctness rate of this approach after six months is 78% against 67%.

Table 4 :Exponentiation operation: retention rate with different learning styles

	Blackboard and chalk	**Using a multimedia tool**	**Building a multimedia tool**
After 1 week	100%	99,83%	99,82%
After 6 months	67%	73%	78%

As computer programming and general learning work in a similar way, and computer programming can effectively support learning, we explored the possibility of using block programming to build specialized programming tools based on multimedia to teach non-scientific school subjects to young children. Using block programming to teach school subejcts is not completely new. Moreno-León and Robles (2015) and Costa (2016) used block programming to teach foreign languages. Lopez (2015) used Scratch itself to teach physics. But the creation of a specific block programming language for teaching a school subject is a significant change. The first tool, called BlockLang (Federici *et al*. 2019), has been built to teach foreign languages. The tool allows the students to assemble short structured phrases, such as for example "I have got seven lemons", and, when the phrase is correct, it gives as an output the drawing of what is stated in the phrase, in this case a boy holding a basket contaning 7 lemons (Figure 9).

Figure 9 :BlockLang: "Programming" English sentences ("I have got seven lemons" sentence)

The constrainings on how to build a correct phrase are embedded in the blocks created by the teacher. So, just to give a few example, to build a correct phrase the word "lemon" must be preceeded by an article or a number

and must end with the correct ending (Figure 10a), and the phrase "I have got" needs instead a final object (Figure 10b).

(a) Nouns and numbers (b) Verbs

Figure 10: BlockLang: simple sentences in the food domain

The second tool instead, called StoryBlock, has been built in order to teach history and literature by creating multimedia stories. The tool allows the students to assemble short sequences of "still scenes" where several characters, that can also say something in speech bubbles, are placed (Figure 11).

Figure 11: StoryBlock: "Programming" stories

All the characters have their own script. The tool has been built in two different versions (Figure 12).

(a) The only block of the simple version (b) Several blocks of the complex version

Figure 12: StoryBlock: simple and complex version

The first version was a simple version with just one block where for each scene the student can only describe if the character is on the Stage or not, which is the character's "costume" and the character's size and orientation (Figure 12a). The second version instead is a more complex version with a wider set of blocks (Figure 12b) that will allow the teacher to introduce the students to the basics of block programming too.

Effective and simple media tools for education

Can the BlockLang and StoryBlock tools be fruitfully used by teachers to teach and by students to learn? Are these simple tools effective? We ran several esperiments.

The first experiment (Federici *et al*, 2019) tested the performance improvement of students of a second grade class when learning the basic vocabulary of the food domain. As we wanted to test the hypothesis that this kind of tools can disclose the relationship between the elements of a given topic, the BlockLang tool was designed to investigate if the constrainings on the correct English word order, imposed by having to embed colored blocks one inside the other, could be easily learned at second grade, something that usually is not required to the students at this level. So, for example, to build the sentence "I have got seven lemons" (Figure 13a) the students had to correctly embed the "seven" numerical block inside the "ONE lemon" block and then embed the "seven lemons" block inside the "I have got ONE EGG" block (Figure 13b).

(a) The only block of the simple version (b) Several blocks of the complex version

Figure 13:Embedding in the correct order the elements of the "I have got seven lemons" sentence

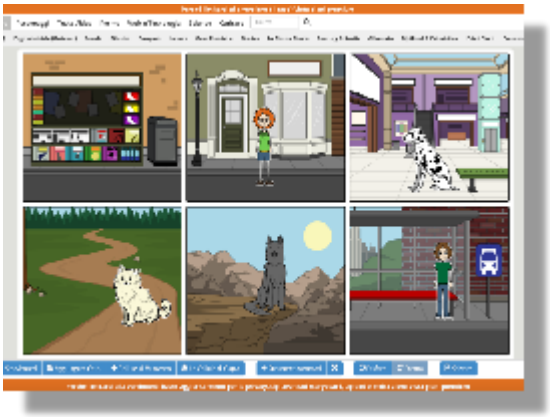
We tested two different second grade classes in an elementary school, one class using the Blocklang tool and one control class using the classic "blackboard and chulk" approach. As expected, we found out that the tool was very successful when the students had to remember the correct English word order in simple phrases (Table 5).

Table 5:Usage ot correct word order in simple sentences on the food domain

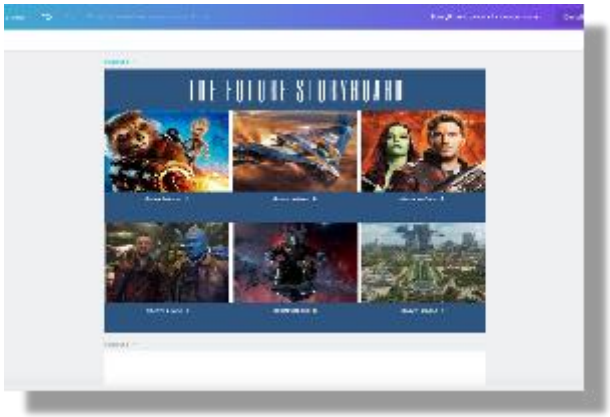
	Test Class	**Control Class**
Percentage of sentences with correct word order	73%	60%

What is interesting to note is that, in order not to give to the students of the test class an advantage over the students of the control class, the test was done on paper in both classes. Even if they were not used to rebuild the correct word order by writing full sentences, the relative performance of the students that had learned this task by using a multimedia tool specifically created for them was 20% better than the one of the students in the control class.

As to the task of allowing teachers and students to build their multimedia stories by using images and sounds with the StoryBlock tool, we ran a small experiment by allowing several people of different ages to test the StoryBlocks tool and comparing its usage to two simple free tools for storytelling, the storybordthat.con and canva.com online tools (Figure 14).



storyboardthat.com



canva.com

Figure 14: Free alternatives to StoryBlock

In this qualitative test we found out that, among a group of about 20 people that tested the tool and that for the most part had no previous experience on graphically building stories either by hand or by using digital tools, 94% of them thought that the StoryBlock tool was easy to use with an average score of 7 out of 10. Moreover, 83% of them said that they preferred using this tool with respect to designing their storyboards by using paper and pencil and 87% of them said that they preferred this homemade tool to the two other free tools specifically designed to easily create storytelling presentations.

Table 5 Testing and comparing simple tools for storyboarding

	StoryBlock	Paper and pencil	Similar tools
Easy to use	94%	Nd	Nd
Easier to use than paper and pencil	83%	17%	Nd
Easier to use than other free tools	87%	Nd	13%

Easy prototyping of multimedia tools for education

How easy is to build this kind of tools? The BlockLang and StoryBlock tools were built by two students of a Communication Studies degree. Those students had just taken one course on block programming based on Scratch lasting just 4 months, with no prior knowledge of computer programming.

Table 6 : development of BlockLang and StoryBlock

	BlockLang	StoryBlock
Prior knowledge of the developer	4-months computer programming course (based on Scratch)	4-months computer programming course (based on Scratch)
Development time	3 weeks	4 weeks
Number of versions	1	2
Easily updatable by user	Yes	No

The BlockLang tool was built in 3 weeks, and part of this time was used to draw all the images related to the food domain words there were present in the tool.

Even if the teacher could have no time to acquire the knowledge necessary to build the tool, with a minimal knowledge on how to drag colored blocks they could easily update the words present in the tool (nouns, adjectives, phrases) with no knowledge at all of computer programming. Indeed, “Power” users can unlock the

tool and use general blocks that are not available to the students. For example, using the "ARTICLE FOOD" block the teacher can add a new food item by just filling in the relevant information (Figure 15).

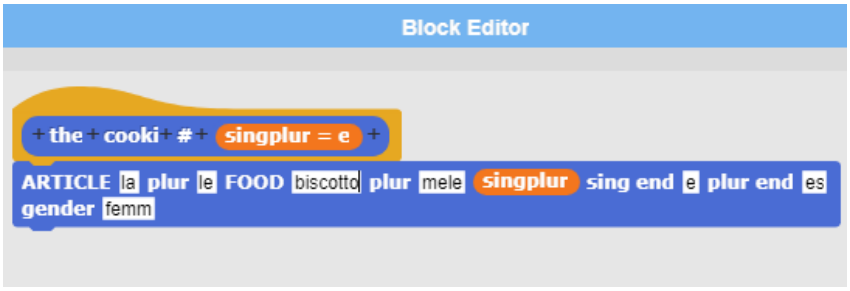


Figure 15:Adding a new food block to the tool

The corresponding drawing can be simply downloaded from the web, and imported in the tool by drag and drop (Figure 16). Once the tool has been updated, it can be locked again, so that the elements of the original Snap tool are not visible to the student.

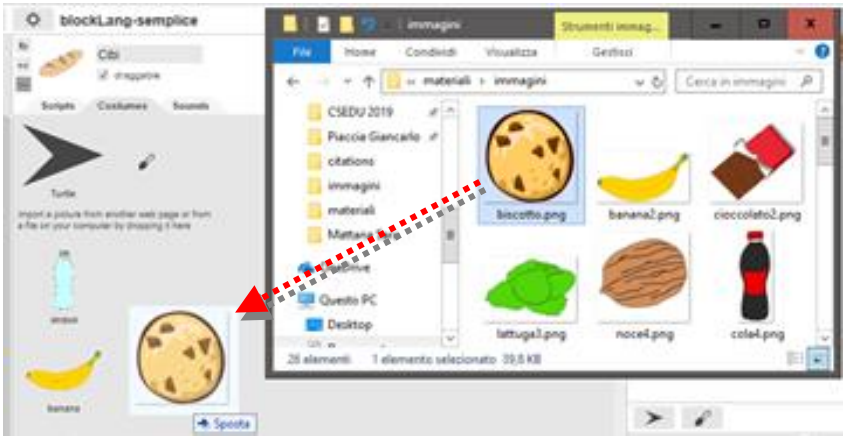


Figure 16: Adding a new food picture to the tool

The development of the StoryBlock tool took 4 weeks as, as we said, it was built in two versions (Figure 12). The StoryBlock tool, contrary to the BlockLang tool, has no "update" mode, as all the elements of a storyboarding tool are already available.

Conclusion

Today it is possible, even for non-technical teachers, to quickly and easily build multimedia educational tools by using specifically tailored programming environments such as BloP. The tools can be created in what is felt like a very short time when compared to time necessary to acquire the deep knowledge necessary to build multimedia interactive tools similar to BlockLang or StoryBlock. The tools have shown to be effective both in quantitative and qualitative tests, by improving the student's engagement and their performances and by making available simple tools for every school topic that are easier to use when compared to other similar tools. The two tools described in this paper were very well accepted by the users and have shown effective to improve their performances in the intended task (Federici *et al*. 2018).

Tools built by using BloP are not limited to teach to students at the elementary school level. As we said, BloP was initially built to teach programming languages in a simpler way (blockC, blockPHP, etc). Furthermore it has been used to build blockASSN, a simplified graphical versions of a language to easily develop searching and sorting algorithms (Federici and Stern, 2011) a computer science problem that it is felt particularly difficult by computer science students (Mendoza *et al*, 2015).

We feel that every school topic that has a linguistic structure, such the language structure of BlockLang, or a storytelling structure, such as history and literature learned by using StoryBlock, or a scientific structure such as the exponentiation operation or the searching and sorting blockASSN language, can be reshaped and reimplemented in BloP. This, in our view, makes BloP a promising tool to make every school topic easier to teach and to learn in a way that is closer to the mindset of the new digital students.

Acknowledgement

Stefano Federici and Elisabetta Gola express their gratitude for the support of Fondazione Banco di Sardegna within the project ``Science and its Logics: The Representation's Dilemma'', Cagliari, number: F72F16003220002.

All authors would like to thank the Istituto Comprensivo n°2 school in Sinnai and the teachers Debora and Donatella for their help in designing and administering the experiment.

References

Beatty, K, 2013, Teaching & researching: Computer-assisted language learning, Routledge, Taylor and Francis Group, New York

Bolkan, J., 2017, Most Teachers Say Classroom Tech Helps Students, but Teachers Need More Training, The JOURNAL – Transforming Education Through Technology. Date of access: 09/05/2019. https://thejournal.com/articles/2017/10/26/most-teachers-say-classroom-tech-helps-students-but-teachers-need-more-training.aspx.

Boulton, K., 2013, Why is it that students always seem to understand, but then never remember?, in toTheReal. Date of access: 09/05/2019. https://tothereal.wordpress.com/2013/05/06/why-is-it-that-students-always-seem-to-understand-but-then-never-remember/ (last retrieved on 29/10/2017).

British Telecommunications, IPSOS Mori, 2016, A NEW CORNERSTONE OF MODERN PRIMARY SCHOOL EDUCATION. Date of access: 09/05/2019. https://digilearnscot.files.wordpress.com/2016/12/bt-ipsos-tech-literacy-full-report.pdf

Chang, C.W., Lee, J.H., Chao, P.Y., Wang, C.Y., and Chen, G.D., 2010, Exploring the Possibility of Using Humanoid Robots as Instructional Tools for Teaching a Second Language in Primary School, in Journal of Educational Technology & Society, Vol. 13, No. 2, pp. 13-24

Costa, S., Gomes, A., Pessoa, T., 2016, Using Scratch to Teach and Learn English as a Foreign Language in Elementary School, in International Journal of Education and Learning Systems, Vol. 1.

Federici, S., 2011, A Minimal, Extensible, Drag-and-Drop Implementation of the C Programming Language, in proceedings of SIGITE'11, October 20–22, 2011, West Point, New York, USA.

Federici S., Stern, L., 2011, A Constructionist Approach to Computer Science. In Barton S, Hedberg J, Suzuki K, Proceedings of Global Learn Asia Pacific 2011, pp. 1352-1361, Chesapeake, VA, Association for the Advancement of Computing in Education, ISBN: 1-880094-85-1.

Federici, S., Gola, E., 2014, BloP: easy creation of Online Integrated Environments to learn custom and standard Programming Languages, in Proceedings of SIREM-SIEL 2014, 1st Joint SIREM-SIel conference. The Innovative LEDI publishing Company.

Federici, S., Gola, E., Brau, D., Zuncheddu, A., 2015, Are educators ready for coding? From students back to teacher: introducing the class to coding the other way round, in Proceedings of CSEDU 2015 - Proceedings of the 7th International Conference on Computer Supported Education, Vol. 1, 2015, pp. 494-500, Lisbon, Portugal

Federici, S., Medas, C., Gola, E., 2018, Who learns better: Achieving long-term knowledge retention by programming-based learning, in Proceedings of CSEDU 2018 - Proceedings of the 10th International Conference on Computer Supported Education, Vol. 2, 2018, pp. 124-133, Funchal, Madeira; Portugal

Federici, S., Sergi E., Gola, E., 2019, Easy Prototyping of Multimedia Interactive Educational Tools for Language Learning based on Block Programming, (to appear) in Proceedings of CSEDU 2019 - Proceedings of the 11th International Conference on Computer Supported Education, Heraklion, Greece.

Han, J, 2012, Emerging Technologies, Robot Assisted Language Learning in Language Learning & Technology, Vol. 16, No. 3 pp. 1-9

Harvey, B, Monig, J, 2010, Bringing "no ceiling" to scratch: Can one language serve kids and computer scientists, in Proceedings of Constructionism 2010, Paris

Homer, M., Noble, J., 2017, Lessons in combining block-based and textual programming, in Journal of Visual Languages and Sentient Systems, 3(1), pp. 22–39. https://doi.org/10.18293/vlss2017-007

Lopez, V., and Hernandez, M. I., 2015, Scratch as a computational modelling tool for teaching physics, in Physics Education, Vol. 50, No. 3, IOP Publishing Ltd,

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E., 2010, The scratch programming language and environment, in ACM Transactions on Computing Education. volume 10, n. 4, doi=10.1145/1868358.1868363.

Mendoza, A., Stern, L., Carroll, J., 2015, "Learnability" as a Positive Influence on Technology Use, in 4th International Multi-Conference on Society, Cybernetics and Informatics, Vol. 1, Editors: Carrasquero JV, Holmqvist M, McEachron D, Tremante A, Welsch F, International Institute of Informatics and Systemics

Moreno-León, J., Robles, G., 2015, Computer programming as an educational tool in the English classroom, in Proceedings of IEEE Global Engineering Education Conference (EDUCON 2015), p. 962.

Nesta, TES, 2014, Teachers feel unprepared for September's new computing curriculum. Date of access: 09/05/2019. http://www.nesta.org.uk/news/teachers-feel-unprepared-september's-new-computing-curriculum

Resnick, M., Silverman, B., 2005, Some reflections on designing construction kits for kids. In Proceedings of IDC 05, the 2005 conference on Interaction design and children, pp. 117-122, Boulder, Colorado.

Schaffhauser, D., Nage, D., 2016, Teaching with Tech: A Love (and hate) Story, The JOURNAL – Transforming Education Through Technology, Vol. 43, N. 5, pp. 6-15, Public Sector Media Group, Woodland Hills, CA, US.

Udomon, I., Xiong, C., Berns, R., Best, K., Vike, N., 2013, Visual, Audio, and Kinesthetic Effects on Memory Retention and Recall, in Journal of Advanced Student Science (JASS), issue 1

Warschauer, M., Healey, D., 1998, Computers and language learning: An overview, in Language Teaching, 31, 57-71.

Weintrop, D., Wilensky, U., 2017, Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms, in ACM Transactions in Computing Education, 18, 1, Article 3. https://doi.org/10.1145/3089799