A minimal, extensible, drag-and-drop implementation of the C programming language

Stefano Federici
Department of Education and Philosophy - University of Cagliari
Via Is Mirrionis, 1
09123 Cagliari, Italy
+39 070 675 7815
sfederici@unica.it

ABSTRACT
Block languages are visual programming languages based on the metaphor of programming bricks. Block languages such as Scratch, StarLogo and Alice, are becoming fundamental tools to get children interested in computer programming. These environments and environments derived from them, such as BYOB, have all the features needed to be strong candidates for introductory computer science courses and are starting to be used in some introductory university courses. Nonetheless, some computer science educators at college and university level feel that block languages are too toy-like to be used in regular computer science curricula. Standard programming languages, such as C or Java, are still thought of as more appropriate. In this work I will describe a third way to programming languages that can be used for introductory computer science: the visual implementations of relevant subsets of standard programming languages such as C. An initial evaluation showing excellent student acceptance is reported in this paper.

Categories and Subject Descriptors

General Terms
Algorithms, Experimentation, Languages, Theory, Verification.

Keywords
Programming languages, visual programming, block languages, Scratch, BYOB.

1. INTRODUCTION
From 2000 to 2007, Computer Science (CS) majors in the United States have dropped by more than 50% [1].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference’10, Month 1–2, 2010, City, State, Country.
Copyright 2010 ACM 1-58113-000-0/00/0010…$10.00.

The trend has improved a bit in the last few years, as "computational science – the use of advanced computing capabilities to understand and solve complex problems – has become critical to scientific leadership, economic competitiveness, and national security" [2]. Computational Science education started to get a lot of attention from the US government: the President’s Information Technology Advisory Committee [2], the Defense Advanced Research Projects Agency [3], the Association for Computing Machinery and the Computer Science Teachers Association [4] all insisted that a large effort must be put in increasing the strength of current educational practices in computer science.

But what should be strengthened? Students of the introductory computer science course of the Faculty of Engineering of the University of Cagliari think that learning how to write computer programs would be interesting “to create solutions on their own that could help them in their studies”, but they also feel that programming is a “long and complex task” and that the highest difficulty is “the number of tiny details in the syntax of computer languages”.

Do we need really to teach computational skills by using standard programming environments such as Turbo Pascal, Netbeans or Eclipse? New introductory tools for computation developed by outstanding research centres such as the Massachusetts Institute of Technology (Scratch, http://scratch.mit.edu), the Carnegie Mellon University (Alice, http://www.alice.org), and the University of Kent (Greenfoot, http://www.greenfoot.org), focus on the creative potential behind computation by “lowering the floor” for entry and by allowing students to get interesting results in a shorter time. Several introductory courses in Computer Science at university level have been reshaped to smooth the way into programming by taking advantage of visual programming environments [5, 6, 7]. But in these courses the adoption of the new environments is limited to the initial one or two weeks. The reason is that teachers think that these new tools are not suitably tailored to be used in a complete course about computer science. What is still missing? Teachers think that these tools do not provide the same potential that we find in regular programming languages.

In this paper I will describe a recent development of a new programming environment that has shown a good potential to lower the student effort required in advancing from introductory tools, such as Scratch, to regular programming languages, such as C. The initial outcomes of a study conducted on 40 students of the introductory computer science course at the Faculty of Engineering of the University of Cagliari will be reported.
2. BLOCK PROGRAMMING LANGUAGES

Scratch is a graphical environment, specifically designed to introduce children to programming [8]. Conceived in 2007 by Mitchel Resnick, director of the Lifelong Kindergarten Group of MIT, Scratch is the youngest and the most successful programming environment designed for people that have never successfully coped with computer programming.

Scratch is based on several interesting principles, such as an easy drag-and-drop mechanism based on the shape of embeddable or stackable blocks that makes impossible to create syntactically incorrect programs. Scratch is event-driven and concurrent.

Figure 1. The Scratch programming environment.

In Scratch (see Figure 1), projects (programs) can be built by dragging blocks (instructions) from the block palette (in which they are clearly organized in logic categories such as "Movement", "Control", "Sensing", etc) to the central Script area. Piles of blocks control the behavior of several sprites on the Stage.

Scratch is, at present, by far the most successful among all programming environments for real beginners. In 3 years, the number of open source projects freely downloadable from the Scratch website rose from about 70,000 to more than 1.8 million.

Despite its very introductory goal, Scratch has started to be successfully used in recent years in some university courses of introductory computer science [5, 6, 7]. In these courses, Scratch is not used as a programming tool to cover the full course of computer programming. Indeed, even though Scratch has acquired some interesting new features since its first appearance in 2007, such as strings and indexed lists, it still lacks some important computational concepts such as functions and procedures, recursion, and complex data structures.

But Scratch is open source, so the language and the tool can be extended (modded). The first and most well-known Scratch mod is BYOB (http://byob.berkeley.edu). BYOB is fully backward compatible with Scratch but it adds to Scratch, the possibility of defining new blocks and to handle higher-order functions [9], along with other new features.

BYOB looks almost identical to Scratch, the only noticeable difference being the presence of just 7 new blocks (see Figure 2).

By using these new blocks, BYOB programmers can assign blocks of code to variables and they can create and run scripts programatically. For example, a pile of blocks can be embedded inside the "the script" block and then can be run by using the "run" block. This new feature is so powerful that it allows everyone to develop their own programming language as an easy-to-use drag-and-drop programming environment.

3. A BLOCK VERSION OF THE C PROGRAMMING LANGUAGE

The facility to define new blocks and higher order functions in BYOB opens up new possibilities. New blocks can be implemented that will provide all the necessary elements to create a minimal block version of classic programming languages such as C or even a domain specific language, for example a language to build and understand search or sorting algorithms [10].

In this paper I will describe blockC, a minimal block implementation of the C programming language. BlockC is deliberately limited to integer, float and char values, and to mono- and bi-dimensional arrays. A full set of new blocks has been added to the standard categories already available in BYOB as a single BYOB program:

- output commands: printf in the Looks category
- input commands: scanf in the Sensing category
- control and conditional commands: include, main, if, if else, switch case, do while, for, repeat until, while in the Control category
- mathematical and logical operators: +, -, /, *, %, !=, ==, >=, <=, &&, || in the Operators category
- assignment command and definition of scalar and array variables: =, int, [ ] (array setting and access) in the Variables category
- definition of functions and procedures (in the Variables category) and call of functions and procedures (in the Sensing category):

As shown in Figure 3, the blocks of blockC are redundant. For example, we have three different types of printf blocks (Looks category). This is a feature that blockC shares with Scratch. Scratch blocks are also redundant, in order to make their usage by students as simple as possible and to avoid syntax errors as much as possible. Students are required to fill in a minimum of information, that is the information that is really relevant to the command being used. They can concentrate on the logic behind building a correct computer program. So, in the case of printf, they don't have to remember exactly which escape sequence ("\n") must be used to print a newline character. This is conveniently embedded in one of the redundant forms of the printf block.

---

1 In this paper we used version 3.0.9 of BYOB.
In this implementation the blocks to define and to call functions and procedures have been split in two different categories (definition in Sensing, call in Variables). This has been done to fit the limited set of BYOB categories and to allow for a better visual differentiation.

The new blocks, once defined, are available in the same left pane as the basic BYOB blocks, at the basic BYOB blocks, at the bottom of the basic block list, separated by a thin grey line (see Figure 4, left).

By dragging and dropping these blocks in the central script area (see Figure 5) it is possible to build simple to medium complexity C programs (see Figure 4, right, and Figure 8).

To run a blockC program it is sufficient to click once on the stack of C blocks. BlockC programs behave exactly as expected, using the BYOB’s Stage to ask for their inputs (see Figure 6) and to write their output to a visible BYOB list (see Figure 7).

2 Using Scratch/BYOB category names is not a real limitation. Indeed, by leveraging on the multilanguage nature of Scratch, a specific language file can be added to the locale folder so that we can change category names to “input”, “output”, “definition”, etc. References to specific language files cannot be directly embedded in BYOB projects. So, if users want to get advantage of this more clear separation of blocks, they have to explicitly switch to the desired “C language” every time they load their blockC project. Even though BYOB source code can be modified, this has not been changed so far, in order to make blockC a regular BYOB project instead of a separate tool (see section 5).
3.1 A minimal and extensible C

Several important features of C (such as pointers, structures, file handling) are not implemented in blockC. BlockC is deliberately a minimal implementation of the C programming language that shows how a drag-and-drop version of a standard programming language can be easily built in BYOB. Indeed, we don’t think that a full implementation of the C programming language developed in a drag-and-drop environment would be a better solution to the problem of having students get interested in computer programming.

The main goal behind the development of blockC is indeed to allow students a smooth transition from a simple and intuitive language such as Scratch or BYOB to more mature programming environments such as Netbeans or Eclipse for standard languages such as C or Java. The goal of blockC is indeed relieve students of the necessity of struggling with syntax during their very first approach to programming, since the syntax of programming languages is what is felt by them to be the biggest obstacle to their introduction to computer programming (see section 4).

Even in such a minimal C, complex and interesting programs for an introductory course on programming can still be implemented. In Figure 8 a full implementation of the Basic Sort algorithm in blockC is shown.

A teacher might be interested in further specific features of C that they use in their own courses. In fact, the very first prototype of blockC (miniC++, http://byob.sitibs.com/miniC++v2.0.ypr, [11]) could only handle integer scalar values. Float and char types have been added to allow students of the introductory computer science course at the Faculty of engineering of the University of Cagliari to implement the same algorithms that were presented to them in the previous years.

As we do not intend to force everyone to use our specific blockC, but we want to make available a strong and positive experience in developing block languages, blockC is open source and it can be easily extended by adding new blocks by means of the simple mechanism provided by BYOB to create custom blocks.

Just to give a few examples, in Figure 9 we see the original implementation of the "int i ;" block that declares integer variables. Adding a similar block to handle floating point and character variables is just a matter of duplicating it and making very small changes, namely changing the type associated with the variable (see Figure 10).
Even adding new loops, for example a do while loop (see Figure 12), is really straightforward when we see how simple is to build a for loop (see Figure 11).

![Figure 11. Implementation of for loop.](image)

![Figure 12. Implementation of do while loop.](image)

4. EXPERIMENTING WITH BLOCK LANGUAGES

What are the real benefits of using a block language instead of a standard programming environment? There is preliminary evidence indicating that using Scratch in an introductory computer-science subject at Harvard led to a sharp reduction in the number of students dropping or failing the subject [12].

A quantitative analysis of the impact of blockC on the performance of engineering students in building C programs is currently under investigation. We have already gathered subjective evidence that students like blockC and that they think it can help them in computer programming, especially in avoiding syntax errors and in remembering well the template of the single C commands. A parallel experiment has been also run by using ASSL (Animated Search and Sort Library) [10] a domain specific programming language inspired by AIA [13]. ASSL is again based on BYOB and allows students to build search and sorting algorithms by expressing them in an almost natural language drag-and-drop formalism based on metal balls and magnets. Students can then visually follow how the algorithm works by seeing on the Stage the result of each "command" (see Figure 13).

![Figure 13. Animated Search and Sort Library.](image)

With ASSL students also said that the tool helped them in really understanding how the algorithms work compared to the study of the corresponding C program. They also appreciated the natural language wording of the "commands" of the library and said that ASSL "is much more useful with respect to the C algorithm. With ASSL everyone can build a sorting algorithm just by reasoning, without having any knowledge of a programming language".

Another interesting result is that during the blockC/ASSL evaluation, students from the blockC/ASSL group were more consistent in attending the seminars compared with students from the control group who were using the Dev-C++ IDE.

Finally, students appreciated a lot the fact that all blockC blocks were available in the block palette. So they didn't have to remember which are the C commands, but could concentrate on the structure of the program.

As for teachers, whereas Scratch and BYOB were not successfully accepted in the past years as tools that could replace a standard programming environment, blockC has been accepted this year as a support tool for the introductory course of computer science at the Faculty of Engineering. Furthermore, during the presentation of miniC at CSEDU 2011, several computer science teachers at the university level who were concerned about the very high dropout in their C courses became interested in this approach.

5. LIMITATIONS OF THE APPROACH

Although block languages are very well accepted by students, since they represent a step towards an easier introduction to programming, they suffer several limitations due to inherent limitations of the BYOB environment. Although it is possible to modify the BYOB code, the implementation reported here was designed to be a standard BYOB project, so to investigate until which point BYOB can be suitable to implement block languages.

Our opinion is that BYOB is incredibly good at this task, but there are still some important limitations that make it not fully suitable as an environment to design programming environments. The main problem is that BYOB cannot "hide" some of the elements of its environment. It would be really useful if developers could hide

---

3 Two groups of 20 students of the introductory course in computer science at the Faculty Engineering have been asked to write an identical set of C algorithms by using either blockC or a standard C/C++ programming environment (Dev-C++, http://www.bloodshed.net/dev/devcpp.html). At the end of the experiment, blockC has also been presented to the students of the Dev-C++ group and their opinions have been collected.

4 Just to give an example, the "j = i + 1" instruction very frequently present in a standard search and sorting algorithms become "move the right magnet where the left magnet is" and "move the right magnet one position to the right".
several sprites and scripts so to be able to run scripts that are not available to the user at load time. For example, as of now, to setup the blockC environment students have to unnaturally click the green flag button at the top right corner of the BYOB interface. Furthermore, students risk damaging the environment by removing start-up scripts or blocks that implement blockC.

Students involved in the experiment complained about the excessive number of blocks that were visible in the block palette (BYOB basic blocks at the top and blockC blocks at the bottom of the palette, see Figure 4, left). In this first stage of the development of block languages we decided not to modify the BYOB’s file format so as not to impact on the possibility of opening blockC as a regular BYOB project. So we added a new “hide/show Basic Blocks” option in the Edit menu of BYOB to improve the experience of the students involved in the experiment by even such a simple change made students much more comfortable at using the tool.

![Figure 14. Usage of the variable block.](image)

Another of the problems reported by students is that it would be useful to have menus to select values in argument slots. Although this kind of selection is available in regular BYOB blocks, BYOB does not allow menu arguments to be defined for custom blocks. Furthermore, even though BYOB provides an extremely elegant and safe way of defining and use variables, by providing a menu in which all available variable names are listed, BYOB doesn't allow a user to programmatically create or access variables by their name. For this very reason we created a rounded "variable" block in which students must type a variable name (see Figure 14). This new block is fairly consistent with a syntactically constrained way of writing blockC programs. Indeed, because of BYOB’s feature of limiting the values typed inside the arguments of new block to numbers/letters, students are only allowed to type numerical values inside arguments of blockC blocks. So, if they have to type a variable name inside an argument, the only way they have is to use the "variable" block.

6. FUTURE WORK

As we think that BYOB is a great environment in which block languages can be fully implemented, the next phase of this work will be enhancing BYOB so that it will be possible to selectively hide sprites, blocks, and scripts to give to the users a cleaner and safer environment as soon as the blockC project is loaded.

7. CONCLUSIONS

BlockC looks a promising tool to help reconcile the desire of teachers to use what they think are sound methods of teaching computer science (C, Java, etc) with the desire of students to work with easier environments in which the focus is not on learning the tiny details of the syntax of a specific programming language but instead learning the logic behind the usage of the mechanisms of a programming language (variables, arrays, functions, etc). This is an important outcome of the experiment as students think they will less likely abandon the computer science course.

8. SOURCE CODE

A streamlined release of BYOB with the preloaded source code of the blockC project is available for free download at http://byob sitiohs.com/blockC.zip.

9. REFERENCES