Teaching Programming Indirectly with "Paint"

Zsuzsanna Szalayné Tahy

Eötvös Loránd University, Faculty of Informatics
Budapest, Hungary
sztzs@caesar.elte.hu

Abstract. In many cases IT literacy is inadequate: users do not understand the concepts of software, and consequently using applications creates more problems. Professionals suggest learning programming to improve computational thinking [1]; but this way is impractical for many. There is another efficient method to teach computational thinking and prepare for programming. By using an application such as Paint software the teaching programming can be embedded into the teaching of application usage. So the way to the high level computational thinking and programming comes through exploring the used application.

Keywords: application, programming, basics, teaching method

1 Introduction

The main task is the improvement in digital literacy for members of the digital society. The most important stage is to understand how IT tools work. As they are based on written programs, the starting point of improvement is to teach programming. This is the idea behind several national curriculums [2], where programming is set as key competence. There are students, who had learnt programming using Scratch in elementary school. Many of them learn program writing later but felt the gap: "I am not able to learn programming". As they cannot transfer the visual solution to code, they cannot transfer the learnt algorithm to everyday practice. The suggested approach is indirect teaching: students explore the well-known software, then understand the idea behind the tools. In fact, the teachers are teaching programming skills but students do not realise this. This method will be demonstrated through the example of the Paint program.

2 Embedded Programming – Using Paint

Paint is an example of one layer handled pixel-graphics software, saving the product into bitmap (bmp) format. In the following the concept of 'take apart, look inside, explore who, what, how, why...' is demonstrated. Students like to paint but in this case they start by exploring the software. Maybe the first question is "Where are the colour tools?" but the next question is about the model of colours and the data structures: "How many different colours are defined in the software?" "How are the codes for colours stored in the memory?"

Through teaching Paint we can teach several concepts of programming such as *set of objects*, *property* and *method*. Students should imagine an instance of array abstraction, so they can understand and practice indexing. Exploring colour schemes creates several questions for students about number systems, data representation, picture size, file size, etc. Practical solutions involve the concept of lossy and lossless compression. By working with painting examples students are able to explore methods and algorithms: students should guess "what does the little man in the computer do" when they resize the image. Depending on the level of teaching and students' precognition, we can model the method or we can write pseudo code. The main points are:

- data structures, models and objects are explored;
- students understand the relation between view and binary sign;
- students state algorithms as hypotheses and try to prove them with tests.

The next stage, after exploring paint and other applications could be "Try to hack an image". This task involves modifying files by writing code. The best practice for teachers is to give a frame code which reads and writes files. The frame program includes the implementation of data structure, the class of *Pixel* with properties like Red, Green and Blue components and the array of pixels. Although the goal is to modify an image, the first step is to explore the code to find relations between codes and learnt concepts. The next step is to learn how to debug and run codes. The last step is the start of explicit learning of programming putting these skills and knowledge all together: modifying an image by coding [3].

3 Summary

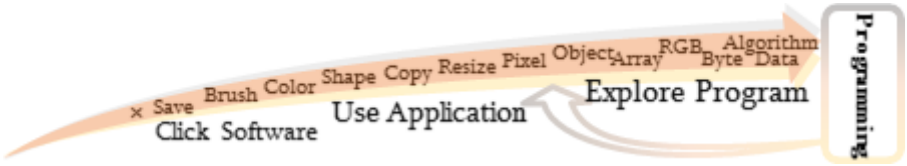

Fig. 1: . . . look inside, explore who, what, how, why. . .

References

1. J.M. Wing: Computational Thinking and CS@CMU Carnegie Mellon University, 2006.
2. Department for Education, GOV.UK: National curriculum [Online] https://www.gov.uk/government/collections/national-curriculum, 16. July 2014.
3. David J. Malan: BMP Puzzles Harvard University, [Online] http://nifty.stanford.edu/2011/malan-bmp-puzzles/, 2011.